Series 32000®

GENIX V.3™ Administrator's Guide

Customer Order Number 424510771-110
NSC Publication Number 424510771-110A
December 1986

Customer Order Number 424510771-110
NSC Publication Number 424510771-110A
December 1986

Series 32000®

GENIX V.3™ Administrator's Guide

© 1986 National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, California 95052-8090

REVISION RECORD

REVISION	RELEASE DATE	SUMMARY OF CHANGES
A	12/86	First Release. *Series 32000*® GENIX V.3™ Administrator's Guide NSC Publication Number 424510771-110A.

PREFACE

This guide describes procedures used in the administration of a *Series 32000®* computer running the GENIX V.3™ operating system. It provides instructions on how to perform the admistrative tasks of a GENIX V.3 system, and gives background information about when and why these tasks are desirable. This guide may also serve as a quick reference to administrative procedures.

Knowledge of GENIX V.3 fundamentals such as the directory structure and the shell is assumed.

The information contained in this manual is for reference only and is subject to change without notice.

No part of this document may be reproduced in any form or by any means without the prior written consent of National Semiconductor Corporation.

GENIX V.3, ISE, ISE16, ISE32, and SYS32 are trademarks of National Semiconductor Corporation.

Series 32000 is a registered trademark of National Semiconductor Corporation.

The GENIX V.3 Operating System is derived from AT&T's UNIX System V.3 Operating System. Portions of the documentation for the GENIX V.3 Operating System are derived from AT&T coprighted UNIX V.3 Operating System documentation and reproduced under License from AT&T.

UNIX is a registered trademark of AT&T.

CONTENTS

Chapter 1 INTRODUCTION 1-1

1.1 WHAT IS A SYSTEM ADMINISTRATOR? 1-1

1.2 SOME ASSUMPTIONS ABOUT YOUR EXPERIENCE 1-1

1.3 HOW THIS GUIDE IS ORGANIZED 1-2

1.4 ABOUT THE PROCEDURES 1-2

1.5 SYSTEM ADMINISTRATION COMMANDS 1-3
1.5.1 System States 1-3
1.5.2 Logins 1-3
1.5.3 Passwords 1-3

1.6 DOCUMENTATION CONVENTIONS 1-4
1.6.1 Command References 1-4
1.6.2 Information in the Examples 1-5

Chapter 2 SYSTEM SECURITY 2-1

2.1 INTRODUCTION 2-1

2.2 IMPORTANT SECURITY GUIDELINES 2-1

2.3 LOGINS AND PASSWORDS 2-2
2.3.1 Password Aging 2-2
2.3.2 Sample /etc/passwd Entries 2-4
2.3.3 Locking Unused Logins 2-5
2.3.4 Special Administrative Passwords 2-5

2.4 SET-UID AND SET-GID 2-8
2.4.1 Check Set-UIDs Owned by Root 2-8
2.4.2 Check Set-UIDs in the Root File System 2-9
2.4.3 Check Set-UIDs in Other File Systems 2-10

Chapter 3 USER SERVICES 3-1

3.1 INTRODUCTION 3-1

3.2 LOGIN ADMINISTRATION 3-1
3.2.1 Adding Users 3-1
3.2.2 Changing or Deleting Password Entries 3-2
3.2.3 Group IDs 3-3

3.3 THE USER'S ENVIRONMENT 3-4
3.3.1 Environment Variables 3-4
3.3.2 Umask 3-6
3.3.3 Default Shell and Restricted Shell 3-7

3.4 USER COMMUNICATIONS SERVICES 3-7
3.4.1 Message of the Day 3-7

3.4.2 News 3-8
3.4.3 Write to All Users 3-9
3.4.4 Mail and Mailx 3-9

3.5 ANTICIPATING USER REQUESTS 3-9
3.5.1 Trouble Reporting 3-9

Chapter 4 PROCESSOR OPERATIONS 4-1

4.1 INTRODUCTION 4-1
4.1.1 General Operating Policy 4-1
4.1.2 Maintaining a System Log 4-2
4.1.3 Administrative Directories and Files 4-2
4.1.4 Root Directories 4-2
4.1.5 Important System Files 4-3

4.2 OPERATING LEVELS 4-5
4.2.1 General 4-5
4.2.2 How init Controls the System State 4-6
4.2.3 A Look at Entering the Multi-User State 4-11
4.2.4 A Look at the System Life Cycle 4-13

4.3 FIRMWARE MONITOR INFORMATION 4-17
4.3.1 Invocation 4-17
4.3.2 Monitor Memory Organization 4-18
4.3.3 Print Commands 4-19
4.3.4 Change Commands 4-20
4.3.5 MMU and Configuration Register Commands 4-20
4.3.6 Data Commands 4-21
4.3.7 Program Execution Commands 4-22
4.3.8 Test Commands 4-22

4.4 DIAGNOSTIC INFORMATION 4-25
4.4.1 Stand-Alone Diagnostics 4-25

Chapter 5 DISK MANAGEMENT 5-1

5.1 INTRODUCTION 5-1

5.2 DEVICE TYPES 5-1
5.2.1 Hard Disk Devices 5-1
5.2.2 Diskette Drives 5-2

5.3 IDENTIFYING DEVICES TO THE OPERATING SYSTEM 5-2
5.3.1 Block and Character Devices 5-4
5.3.2 Defining a New Special File 5-4

5.4 FORMATTING AND PARTITIONING 5-5
5.4.1 Formatting Disks 5-5
5.4.2 Hard Disk Partitioning 5-5
5.4.3 Planning to Change Hard Disk Partitions 5-5
5.4.4 Changing Partitions to Increase Swap Space 5-6

5.5 OTHER DISK OPERATIONS 5-6
5.5.1 Duplicating Disks 5-6

5.5.2 Verifying Usability 5-7
5.6 THE BAD BLOCK HANDLING FEATURE 5-7
5.6.1 When Is a Block Bad? 5-8
5.6.2 When Are Bad Blocks Detected? 5-9
5.6.3 Bad Block Handling: Normal Operation 5-9
5.6.4 A Bad Block Handling Scenario 5-10
5.6.5 Errors in Single-User Mode 5-12
5.6.6 Dealing With Data Loss 5-12
Chapter 6 FILE SYSTEM ADMINISTRATION 6-1
6.1 INTRODUCTION 6-1
6.1.1 How the File System is Organized 6-1
6.1.2 Block 0 6-3
6.1.3 Block 1: the Super-Block 6-3
6.1.4 I-Nodes 6-4
6.1.5 Storage Blocks 6-7
6.1.6 Free Blocks 6-7
6.1.7 Summary 6-7
6.2 THE RELATIONSHIP BETWEEN THE FILE SYSTEM AND THE STORAGE DEVICE 6-7
6.2.1 Disk Format 6-7
6.2.2 Partitions 6-8
6.2.3 Size Limitations 6-9
6.3 HOW THE FILE SYSTEM WORKS 6-10
6.3.1 Tables in Memory 6-10
6.3.2 System Steps in Accessing a File 6-11
6.3.3 Synchronization 6-14
6.3.4 Search Time 6-14
6.3.5 Holes in Files 6-15
6.3.6 Summary 6-15
6.4 ADMINISTERING THE FILE SYSTEM 6-15
6.4.1 Creating a File System and Making it Available 6-15
6.4.2 Using mkfs 6-15
6.4.3 Relating the File System Device to a File System Name . . 6-16
6.4.4 Mounting and Unmounting File Systems 6-17
6.4.5 Summary 6-18
6.5 MAINTAINING A FILE SYSTEM 6-18
6.5.1 The Need for Policies 6-19
6.5.2 Shell Scripts for File System Administration 6-19
6.5.3 Checking for File System Consistency 6-19
6.5.4 Monitoring Disk Usage 6-20
6.5.5 Monitoring Percent of Disk Space Used 6-20
6.5.6 Monitoring Files and Directories that Grow 6-21
6.5.7 Identifying and Removing Inactive Files 6-21
6.5.8 Identifying Large Space Users 6-22
6.5.9 File System Backup and Restore 6-23

6.5.10 Complete Backup 6-23
6.5.11 Incremental Backup 6-24
6.5.12 Selective Backup 6-25
6.5.13 Backup Schedule Reminder 6-25
6.5.14 Restoring a File System from Backup 6-26

6.6 WHAT CAN GO WRONG WITH A FILE SYSTEM 6-26
6.6.1 Hardware Failure 6-26
6.6.2 Program Interrupts 6-27
6.6.3 Human Error 6-27

6.7 HOW TO CHECK A FILE SYSTEM FOR CONSISTENCY 6-27
6.7.1 The Fsck Utility 6-28
6.7.2 The Fsck Command 6-28
6.7.3 Sample Command Use 6-29
6.7.4 File System Components Checked by Fsck 6-30
6.7.5 Running fsck 6-35

Chapter 7 PERFORMANCE MANAGEMENT 7-1

7.1 INTRODUCTION 7-1

7.2 GENERAL APPROACH TO PERFORMANCE MANAGEMENT . . 7-1

7.3 IMPROVING PERFORMANCE 7-3
7.3.1 Modifying the Tunable Configuration Parameters 7-3
7.3.2 Improving Disk Utilization 7-3
7.3.3 Defining Best System Usage Patterns 7-6

7.4 SAMPLES OF GENERAL PROCEDURES 7-7
7.4.1 Sample Procedure for Investigating Performance Problems 7-7
7.4.2 Sample System Reconfiguration 7-10

7.5 PERFORMANCE TOOLS 7-11
7.5.1 Sar 7-12
7.5.2 Sag 7-24
7.5.3 Timex 7-25
7.5.4 Sadp 7-26

7.6 TUNABLE PARAMETERS 7-28
7.6.1 Kernel Parameters 7-33
7.6.2 Paging Parameters 7-36
7.6.3 Streams Parameters 7-37
7.6.4 Log Driver Parameters 7-39
7.6.5 Message Parameters 7-40
7.6.6 Semaphore Parameters 7-41
7.6.7 Shared Memory Parameters 7-42
7.6.8 Remote File Sharing Parameters 7-42

Chapter 8 LINE PRINTER ADMINISTRATION 8-1

8.1 INTRODUCTION 8-1
8.1.1 How the LP Spooling System Works 8-1

8.2 ADMINISTRATIVE COMMANDS 8-2
8.2.1 Command Descriptions and Examples 8-2
8.3 PRINTER INTERFACE PROGRAMS 8-8
8.3.1 Model Interface Programs 8-8
8.3.2 Writing Interface Programs 8-8
8.4 FILES AND DIRECTORIES 8-11
8.4.1 /usr/spool/lp/FIFO 8-11
8.4.2 /usr/spool/lp/default 8-11
8.4.3 /usr/spool/lp/log 8-11
8.4.4 /usr/spool/lp/oldlog 8-11
8.4.5 /usr/spool/lp/outputq 8-11
8.4.6 /usr/spool/lp/pstatus 8-12
8.4.7 /usr/spool/lp/qstatus 8-12
8.4.8 /usr/spool/lp/seqfile 8-12
8.4.9 /usr/spool/lp/class 8-12
8.4.10 /usr/spool/lp/interface 8-12
8.4.11 /usr/spool/lp/member 8-12
8.4.12 /usr/spool/lp/model 8-13
8.4.13 /usr/spool/lp/request 8-13
8.4.14 Lock Files 8-13
8.4.15 Cleaning Out Log Files 8-14
Chapter 9 TTY MANAGEMENT 9-1
9.1 INTRODUCTION 9-1
9.1.1 Definition of Terms 9-1
9.2 THE TTY SYSTEM 9-2
9.2.1 How the TTY System Works 9-2
9.2.2 How to Tell What Line Settings Are Defined 9-2
9.2.3 How to Create New Line Settings and Hunt Sequences . . 9-3
9.2.4 How to Modify TTY Line Characteristics 9-4
9.2.5 How to Set Terminal Options 9-5
Chapter 10 BASIC NETWORKING 10-1
10.1 INTRODUCTION 10-1
10.2 NETWORKING HARDWARE 10-1
10.3 NETWORKING COMMANDS 10-2
10.3.1 User Programs 10-2
10.3.2 Administrative Programs 10-3
10.4 DAEMONS 10-3
10.4.1 Internal Programs 10-4
10.5 SUPPORTING DATA BASE 10-4
10.5.1 Devices File 10-5
10.5.2 Dialers File 10-9
10.5.3 Systems File 10-11
10.5.4 Dialcodes File 10-15

10.5.5 Permissions File 10-15
10.5.6 Poll File 10-21
10.5.7 Devconfig File 10-22
10.5.8 Sysfiles File 10-22
10.5.9 Other Networking Files 10-23

10.6 ADMINISTRATIVE FILES 10-23

10.7 DIRECT LINKS 10-25
10.7.1 General 10-25
10.7.2 BNU Software and Direct Links 10-26

Chapter 11 REMOTE FILE SHARING 11-1

11.1 OVERVIEW 11-1
11.1.1 Resource Sharing 11-1
11.1.2 Domains 11-3
11.1.3 Administration 11-4
11.1.4 Security 11-5
11.1.5 Connect Security 11-5

11.2 SHARING RESOURCES 11-7
11.2.1 Advertise Resources 11-8
11.2.2 Mount Remote Resources 11-12

11.3 MAPPING REMOTE USERS 11-17
11.3.1 How Mapping Works 11-17
11.3.2 Mapping Components 11-18
11.3.3 Choosing Mapping Strategy 11-21

11.4 MAINTENANCE 11-25
11.4.1 Introduction 11-25
11.4.2 Host Maintenance 11-27
11.4.3 Domain Maintenance 11-32
11.4.4 Remote File Sharing Mode (init 3) 11-34
11.4.5 Recovery 11-37
11.4.6 Monitoring 11-40
11.4.7 Parameter Tuning 11-46

Procedure 1 SYSTEM IDENTIFICATION AND SECURITY PROCEDURES 1-1

1.1 CHECK CONSOLE TERMINAL CONFIGURATION (Procedure 1.1) 1-1

1.2 SET TIME AND DATE (Procedure 1.2) 1-2

1.3 ESTABLISH OR CHANGE SYSTEM AND NODE NAMES (Procedure 1.3) 1-4
1.3.1 System Administration Menu—nodename 1-4
1.3.2 Command—uname 1-5

1.4 ASSIGN SPECIAL ADMINISTRATIVE PASSWORDS (Procedure 1.4) 1-6

Procedure 2 USER SERVICES PROCEDURES 2-1

2.1 ADD USERS OR GROUPS (Procedure 2.1) 2-1
2.2 MODIFY USER INFORMATION (Procedure 2.2) 2-4
2.3 DELETE USERS OR GROUPS (Procedure 2.3) 2-7
2.4 LIST USERS OR GROUPS (Procedure 2.4) 2-9
2.5 WRITE TO ALL USERS (Procedure 2.5) 2-11
Procedure 3 PROCESSOR OPERATIONS PROCEDURES 3-1
3.1 POWERUP (Procedure 3.1) 3-1
3.2 POWERDOWN (Procedure 3.2) 3-2
3.2.1 From Multi-User 3-2
3.2.2 From Single-User 3-4
3.3 SHUTDOWN TO SINGLE-USER (Procedure 3.3) 3-5
3.4 RETURN TO MULTI-USER (Procedure 3.4) 3-6
3.4.1 From Single-User 3-6
3.5 RECOVERY FROM SYSTEM TROUBLE (Procedure 3.5) 3-7
3.5.1 Perform a System Dump 3-8
3.5.2 Example of sysdump 3-8
Procedure 4 DISK MANAGEMENT PROCEDURES 4-1
4.1 FORMAT DISKETTES (Procedure 4.1) 4-1
4.2 DUPLICATE DISKETTES (Procedure 4.2) 4-3
4.3 CHECK FOR HARD-DISK ERRORS (Procedure 4.3) 4-5
Procedure 5 FILE SYSTEM ADMINISTRATION PROCEDURES 5-1
5.1 CREATE FILE SYSTEMS ON DISKETTE (Procedure 5.1) 5-2
5.2 CREATE FILE SYSTEMS ON HARD DISK (Procedure 5.2) 5-5
5.3 MAINTAIN FILE SYSTEMS (Procedure 5.3) 5-9
5.3.1 File System Checking 5-9
5.3.2 Monitoring Disk Usage 5-11
5.4 FILE SYSTEM BACKUP AND RESTORE (Procedure 5.4) 5-13
5.4.1 Complete Backup 5-14
5.4.2 Incremental Backup 5-16
5.4.3 Selective Backup 5-18
5.4.4 Restore 5-20
5.4.5 Backup Schedule Reminders 5-21
Procedure 6 SYSTEM RECONFIGURATION PROCEDURES 6-1
6.1 RECONFIGURE THE SYSTEM (Procedure 6.1) 6-1
6.1.1 Modifying the Tunable Parameters 6-2
6.1.2 Rebuilding the Operating System 6-2
6.2 UNBOOTABLE OPERATING SYSTEM RECOVERY (Procedure 6.2) 6-4

Procedure 7 LP SPOOLER ADMINISTRATION PROCEDURES 7-1
7.1 INSTALL THE LP SPOOLER (Procedure 7.1) 7-1
7.2 SHUT DOWN FOR MAINTENANCE (Procedure 7.2) 7-2
7.3 RESTART THE SCHEDULER (Procedure 7.3) 7-2
7.4 ADD ANOTHER LP PRINTER (Procedure 7.4) 7-3
Procedure 8 TTY MANAGEMENT PROCEDURES 8-1
8.1 CHECK TTY LINE SETTINGS (Procedure 8.1) 8-1
8.2 MAKE TTY LINE SETTINGS (Procedure 8.2) 8-4
8.3 MODIFY TTY LINE CHARACTERISTICS (Procedure 8.3) 8-6
Procedure 9 BASIC NETWORK PROCEDURES 9-1
9.1 INSTALL BASIC NETWORKING SOFTWARE (Procedure 9.1) . . 9-1
9.1.1 Run Sysadm Installpkg 9-2
9.2 SET UP BASIC NETWORKING FILES (Procedure 9.2) 9-3
9.2.1 Set Up Devices File - devicemgmt 9-4
9.2.2 Set Up /etc/inittab - portmgmt 9-6
9.2.3 Set Up Systems File - systemmgmt 9-7
9.2.4 Set Up Poll File - pollmgmt 9-8
9.2.5 Set Up Permissions File 9-9
9.2.6 Set Up Devconfig File 9-9
9.2.7 Set Up Sysfiles File 9-10
9.2.8 Add Uucp logins 9-10
9.2.9 Other Networking Files 9-11
9.3 BASIC NETWORKING MAINTENANCE (Procedure 9.3) 9-11
9.3.1 Automated Networking Maintenance (cron) 9-12
9.3.2 Uudemon.poll 9-12
9.3.3 Uudemon.hour 9-12
9.3.4 Uudemon.admin 9-13
9.3.5 Uudemon.cleanup 9-13
9.3.6 Manual Maintenance 9-13
9.4 BASIC NETWORKING DEBUGGING (Procedure 9.4) 9-14
9.4.1 Check for Faulty ACU/Modem 9-14
9.4.2 Check Systems File 9-14
9.4.3 Debug Transmissions 9-14
9.4.4 Check Error Messages 9-15
9.4.5 Check Basic Information 9-15
9.5 REMOVE BNU SOFTWARE (Procedure 9.5) 9-16
9.5.1 Prerequisites 9-16
9.5.2 Single-User Mode 9-17
9.5.3 Run sysadm removepkg 9-17
9.5.4 Return to Multi-User Mode 9-17
9.6 SET UP BNU NPACK NETWORK (Basic) [Procedure 9.6] 9-18

9.6.1 Prerequisites 9-18
9.6.2 Create NPACK Systems Entry 9-18
9.6.3 Create NPACK Devices Entry 9-19
9.6.4 Create NPACK Devconfig Entries 9-20
9.6.5 Set Up NPACK NETWORK Listener 9-20
9.6.6 NPACK Dialers Entry 9-21

9.7 SET UP BNU NPACK NETWORK (Special) [Procedure 9.7] 9-21

Procedure 10 REMOTE FILE SHARING PROCEDURES 10-1

10.1 DOMAIN SETUP (primary) [Procedure 10.1] 10-1
10.1.1 Basic Domain Setup 10-2
10.1.2 Optional Domain Setup 10-6

10.2 HOST SETUP (Procedure 10.2) 10-8
10.2.1 Basic Host Setup 10-8
10.2.2 Optional Host Setup 10-11

10.3 ADVERTISE RESOURCES (Procedure 10.3) 10-14

10.4 MOUNT REMOTE RESOURCES (Procedure 10.4) 10-16

Appendix A DEVICE NAMES AND DESIGNATORS A-1

A.1 INTRODUCTION A-1
A.1.1 System V Release 2.0 and Later A-1

A.2 HARD DISK DEFAULT PARTITIONS A-3

A.3 DISKETTE PARTITIONS A-3

Appendix B DIRECTORIES AND FILES B-1

B.1 INTRODUCTION B-1

B.2 DIRECTORIES B-1

B.3 FILES B-1
B.3.1 /etc/checklist B-3
B.3.2 /etc/fstab B-3
B.3.3 /etc/gettydefs B-4
B.3.4 /etc/group B-4
B.3.5 /etc/init.d Directory B-6
B.3.6 /etc/inittab B-6
B.3.7 /etc/master.d Directory B-8
B.3.8 /etc/motd B-8
B.3.9 /etc/passwd B-8
B.3.10 /etc/profile B-9
B.3.11 /etc/rc0 . . . . . B-9
B.3.12 /etc/rc0.d Directory B-14
B.3.13 /etc/rc2 . . . . . B-14
B.3.14 /etc/rc2.d Directory B-14
B.3.15 /etc/rc.d Directory B-17
B.3.16 /etc/rc3 . . . . . B-17
B.3.17 /etc/rc3.d Directory B-17

B.3.18 /etc/rstab B-17
B.3.19 /etc/save.d Directory B-17
B.3.20 /etc/shutdown B-17
B.3.21 /etc/shutdown.d Directory B-22
B.3.22 /etc/TIMEZONE B-22
B.3.23 /etc/utmp B-23
B.3.24 /etc/wtmp B-23
B.3.25 /usr/adm/sulog B-23
B.3.26 /usr/lib/cron/log B-23
B.3.27 /usr/lib/help/HELPLOG B-25
B.3.28 /usr/lib/spell/spellhist B-26
B.3.29 /usr/news B-26
B.3.30 /usr/options Directory B-26
B.3.31 /usr/spool/cron/crontabs B-29

Appendix C GENIX V.3 System Error Messages C-1

C.1 INTRODUCTION C-1

C.2 GENIX V.3 SYSTEM NOTICE MESSAGES C-1

C.3 GENIX V.3 SYSTEM WARNING MESSAGES C-4

C.4 GENIX V.3 SYSTEM PANIC MESSAGES C-5

FIGURES

Figure 3-1. A Default /etc/profile 3-5
Figure 3-2. Environment Array for a Typical User 3-6
Figure 3-3. Sample Trouble Report 3-10
Figure 4-1. A Look at System Initialization 4-8
Figure 4-2. A Look at the System Life Cycle 4-14
Figure 5-1. Directory Listing Extracts: Regular and Device Files 5-3
Figure 6-1. A GENIX V.3 File System 6-1
Figure 6-2. Adding the /usr File System 6-2
Figure 6-3. The GENIX V.3 View of a File System 6-3
Figure 6-4. The Super-Block 6-5
Figure 6-5. The File System Address Chain 6-6
Figure 6-6. The System I-Node Table 6-11
Figure 6-7. File System Tables and Their Pointers 6-12
Figure 6-8. Sample /etc/save.d/except File 6-24
Figure 6-9. Error Message Abbreviations in fsck 6-36
Figure 7-1. Outline of Typical Troubleshooting Procedure 7-8

Figure 7-2. Example of sag Output 7-24
Figure 7-3. Output from Sadp: Cylinder Access Histogram 7-27
Figure 7-4. Output from Sadp: Seek Distance Histogram 7-29
Figure 8-1. Dumb Line Printer Interface Program 8-9
Figure 9-1. Gettydefs Entries 9-3
Figure 9-2. Getty Entries from /etc/inittab 9-5
Figure 11-1. Example - Sharing Resources 11-2
Figure 11-2. Example - Remote File Sharing Security 11-6
Figure 11-3. Output from Sar Dc 11-41
Figure 11-4. Output from Sar Du 11-43
Figure 11-5. Output from Sar S 11-44
Figure 11-6. Output from Fusage 11-45
Figure 11-7. Output from Df 11-46
Figure B-1. Typical /etc/checklist File B-3
Figure B-2. Typical /etc/fstab File B-4
Figure B-3. Typical Gettydefs File B-5
Figure B-4. Typical /etc/group File B-6
Figure B-5. Typical /etc/inittab File B-7
Figure B-6. Typical /etc/passwd File B-10
Figure B-7. Standard /etc/profile File B-11
Figure B-8. Typical /etc/rc0 File B-12
Figure B-9. Typical /etc/rc2 File B-15
Figure B-10. Typical /etc/shutdown File B-18
Figure B-11. Typical /etc/TIMEZONE File B-22
Figure B-12. Typical /usr/adm/sulog File B-24
Figure B-13. Typical /usr/lib/cron/log File B-25
Figure B-14. Typical /usr/lib/help/HELPLOG File B-26
Figure B-15. Typical /usr/options Directory B-27
Figure B-16. Typical /usr/options File Contents B-28
Figure B-17. Typical /usr/spool/cron/crontabs/root File B-30

TABLES

TABLE 2-1. PASSWORD AGING CHARACTER CODES 2-3

TABLE 4-1. SYSTEM STATES 4-7

TABLE 4-2. MEMORY TEST PATTERNS IN HEXADECIMAL 4-23

TABLE 4-3. MEMORY TESTS 4-23

TABLE 6-1. DISK PARTITIONS, 30-MEGABYTE DRIVE 6-9

TABLE 6-2. INTERRECORD GAP AND BLOCKS/CYLINDER RECOMMENDATIONS 6-16

TABLE 7-1. SUGGESTED PARAMETER VALUES: GENIX V.3 SYSTEMS . . . 7-30

TABLE 8-1. USER COMMANDS FOR THE LP SPOOLING SYSTEM 8-2

TABLE 8-2. ADMINISTRATIVE COMMANDS FOR THE LP SPOOLING SYSTEM 8-3

TABLE 11-1. RFS TUNABLE PARAMETER SETTINGS 11-49

TABLE A-1. EXAMPLES OF TYPICAL DEVICE PARTITIONS A-2

TABLE A-2. SINGLE HARD DISK DEFAULT PARTITIONING A-4

TABLE A-3. FLOPPY DISK DRIVE A-4

GLOSSARY

INDEX

Chapter 1

INTRODUCTION

This guide describes procedures used in the administration of a *Series 32000®* Computer running the GENIX V.3™ operating system. It is designed to accomplish the following objectives:

- provide clear instructions on how to perform the administrative tasks of a GENIX V.3 system
- give you background information about when and why these tasks are desirable
- serve as a quick reference to administrative procedures

1.1 WHAT IS A SYSTEM ADMINISTRATOR?

A System Administrator performs two main tasks:

- decides what rules are needed to govern the use of the computer system
- implements those rules so as to provide the maximum amount of computing service for the system's users, consistent with the physical limitations of the machine

Obviously, if you are the only user of your *Series 32000* Computer, these tasks consist simply of those things you do to keep the machine running and your programs and data from disappearing permanently. If, on the other hand, your *Series 32000* Computer will be used by a number of other people, the tasks become more complex, and you are required to be aware of the needs of your whole user community.

1.2 SOME ASSUMPTIONS ABOUT YOUR EXPERIENCE

This guide assumes that you know the mechanics of using a computer terminal to enter commands, and that you have an awareness of such GENIX V.3 system fundamentals as the directory structure and the shell. We also expect that you feel comfortable using the *Series 32000* Computer; you know how to turn it on and how to use such things as the diskette drive. (If you feel unsure of your knowledge on these points, you might find it helpful to refer to the *Documentation Roadmap*. The *Documentation Roadmap* contains titles and descriptions of other GENIX V.3 system manuals you may want to look at before starting on this one.

1.3 HOW THIS GUIDE IS ORGANIZED

There are two main parts to this guide:

Part 1 - Support Information	Part 1 contains ten chapters of detailed information about the GENIX V.3 operating system. Each chapter is related to a general topic, such as User Services or Processor Operations.
Part 2 - Procedures	Part 2 contains ten step-by-step procedures that correspond to the chapters in Part 1.

Three appendices, a glossary, and an index are located in the back of the book.

1.4 ABOUT THE PROCEDURES

A table at the beginning of each procedure gives you information in capsule form. Table entries appear only when the information is relevant. The tables follow this style:

Purpose	Summary of what the procedure is used for.
When Performed	When you should schedule the procedure.
Starting Conditions	The state the computer should be in when you begin the procedure. Any special login requirements.
sysadm menu	The part of the System Administration Menu package that contains the subcommands to perform the procedure.
Commands	The commands used to perform the procedure.
Firmware Programs	The names of programs to be run in firmware mode.
Bootable Programs	The names of programs used to boot the system.
Media	Diskettes or tapes used in the procedure.
Time	Approximately how long the procedure takes.
Caution	Special instructions you must follow before or during the procedure to ensure the integrity of your system software and user files.

Reference The chapter and section in Part 2 where this topic is more fully discussed.

Other GENIX V.3 system manuals where additional information can be found.

1.5 SYSTEM ADMINISTRATION COMMANDS

The majority of the procedures are based on menus of the System Administration Menu package. This package consists of a hierarchical arrangement of interactive screens that lead you through system administration tasks.

The procedures shown in this guide by-pass the higher level System Administration Menus and take you directly to the subcommands. Subcommands are the equivalent of menu selections from lower level menus. If you prefer, you may start at the main menu with the unadorned command

```
$ sysadm
```

You can get to the submenu level with a command like

```
$ sysadm filemgmt
```

1.5.1 System States

In some procedures, we state that a particular system state is required. In most cases this means that the system must be in either the single- or multi-user state. The single-user state corresponds to run level 1, while the multi-user state corresponds to run levels 2 or 3. Procedures for bringing the system to different system states are found in Part 1 under Processor Operations Procedures. See Section 9.2 for more information on system states.

1.5.2 Logins

In some procedures, we state that a particular login is required. This frequently means that you must be logged in as **root** to do the procedure. The phrase "an authorized login" is also used. The standard meaning of this term is that you must log in using a special administrative or system login name to do the procedure (see Chapter 2 for a list of these logins).

1.5.3 Passwords

It is strongly recommended that you set up and use passwords for administrative and system logins (see Procedure 1.4 for information on how to do this). In the procedures, we assume that such password protection has been established. When you enter a **sysadm** command as an ordinary user, therefore, you are prompted for a password. We show this with an entry like:

```
$ sysadm backup
Password:
```

At this point, to go ahead with the procedure, you are required to enter an acceptable password. As is always the case in the GENIX V.3 system, the password is not echoed to your screen.

In procedures that require you to be logged in as **root** (that is, the super-user), you are not prompted for the **sysadm** password. Also, the pound sign (#) prompt is used for the **root** login. Here's an example:

```
# sysadm backup
```

1.6 DOCUMENTATION CONVENTIONS

Whenever the text includes examples of output from the computer and/or commands entered by you, we follow the standard notation scheme that is common throughout GENIX V.3 system documentation:

- Text that you type in from your terminal is shown in **bold** type.
- Text that is printed on your terminal by the computer is shown in Roman type.
- Comments and explanations within a display are shown in *italic* type and are indented to separate them from the text that represents computer output or input.

  Italics are also used to show substitutable values, such as *file*, when the format of a command is shown.
- There is an implied **RETURN** at the end of each command and menu response you enter.
- Where you may be expected to enter only a **RETURN** (as in the case where you are accepting a menu default), the symbol **<CR>** is used.
- The dollar sign ($) and pound sign (#) symbols are the standard default prompt signs for an ordinary user and **root**, respectively.

  $ — means you are logged in as an ordinary user.
  # — means you are logged in as **root**
- When the # prompt is used in an example, it means the command illustrated may be used only by **root**.
- When the full path name of a command is shown in an example (like **/etc/fsck**) the command must be entered that way.

1.6.1 Command References

When commands are mentioned in a section of the text for the first time, a reference to the manual section where the command is described is included in parentheses: **command**(section). The numbered sections are located in the following manuals:

Sections (1), (6)	*User's Reference Manual*
Sections (1), (1M), (7), (8)	*Administrator's Reference Manual*
Sections (1), (2), (3), (4), (5)	*Programmer's Reference Manual*

1.6.2 Information in the Examples

While every effort has been made to present displays of information just as they appear on your terminal, it is possible that your system will produce slightly different output. Some displays depend on a particular machine configuration that may differ from yours. Changes between releases of the GENIX V.3 system software may cause small differences in what appears on your terminal.

Chapter 2

SYSTEM SECURITY

2.1 INTRODUCTION

This chapter deals with maintaining the security of your *Series 32000* Computer system.

- Important security guidelines

  Guidelines for setting passwords and permissions; protecting the system from unauthorized access.

- Logins and passwords

  Aging passwords to control the amount of time passwords may be kept for logins.

  Locking logins to prevent their being used.

  Protecting administrative commands and logins with passwords.

- Set-UID and Set-GID

  Preventing unauthorized use of programs conditioned to execute via an administrative login.

2.2 IMPORTANT SECURITY GUIDELINES

The security of the system is ultimately the responsibility of all who have access to the system. No system is totally secure. The system is not tamperproof. Some of the items to consider are as follows.

- Especially with a small computer, physical access to the machine must be considered. Anyone with physical access to the machine can literally walk off with it.
- Set the access permissions to directories and files to allow only the necessary permissions for owner, group, and others.
- All logins should have passwords. Change passwords regularly. Do not pick obvious passwords. Six-to-eight character nonsense strings using letters and numbers are recommended over standard names. Logins that are not needed should be either removed or blocked.
- Dial-up ports that do not have logins usually cause trouble.
- Any system with dial-up ports is not really secure. Top secret information should not be kept on a system with dial-up ports.
- Users who make frequent use of the **su** command can compromise the security of your system by accessing files belonging to other users without the other users' knowledge. This command is also dangerous since you must know another user's login password to use it. The more people who know a given login and password, the less secure access is to the system. For this reason, a log is kept on the use of the command. Check the file

/usr/adm/sulog to monitor use of the **su** command. The format of **/usr/adm/sulog** is described in Appendix B: Administrative Directories and Files.

- Login directories, **.profile** files, and files in **/bin, /usr/bin, /lbin,** and **/etc** that are writable by others are security give-aways.
- Encrypt sensitive data files. The **crypt**(1) command together with the encryption capabilities of the editors (**ed** and **vi**) provide protection for sensitive information. This requires the Security Administration Utilities package (domestic customers only).
- Log off the system if you must be away from the data terminal. Do not leave a logged-in terminal unattended, especially if you are logged in as **root.**

2.3 LOGINS AND PASSWORDS

The discussion of logins and passwords covers

- password aging
- sample **/etc/passwd** entries
- locking unused logins
- special administrative logins

2.3.1 Password Aging

The password aging mechanism forces users to change their password on a periodic basis. Provisions are made to prevent a user from changing a new password before a specified time interval. Password aging is selectively applied to logins by editing the **/etc/passwd** file. Realistically, password aging forces a user to adopt at least two passwords for a login. If you require more access control than what is provided by password aging, you can change **/etc/profile** to require a second access code as part of the login process (see Procedures 2.1, 2.2, and 2.3).

The password aging information is appended to the encrypted password field in the **/etc/passwd** file. The password aging information consists of a comma and up to four bytes (characters) in the format:

,Mmww

The meaning of these fields is as follows:

, The delimiter between the password itself and the aging information.

M The Maximum duration of the password (in weeks) following the notation in Table 2-1.

m The minimum time interval before the existing password can be changed by the user (in weeks) following the notation in Table 2-1.

ww The week (counted from the beginning of 1970) when the password was last changed; two characters, *ww*, are used. You do not enter this information. The system automatically adds these characters to the password aging information.

TABLE 2-1. PASSWORD AGING CHARACTER CODES

CHARACTER	NUMBER OF WEEKS
. (period)	0 (zero)
/ (slash)	1
0 through 9	2 through 11
A through Z	12 through 37
a through z	38 through 63

All times are specified in weeks (0 through 63) by a 64 character alphabet. Table 2-1 shows the relationship between the numerical values and character codes. Any of the character codes may be used in the four fields of the password aging information.

Two special cases apply for the character codes:

- If *M* and *m* are equal to zero (the code, ..), the user is forced to change the password at the next login. No further password aging is then applied to that login.
- If *m* is greater than *M* (for example, the code, ./), only **root** is able to change the password for that login.

2.3.2 Sample /etc/passwd Entries

Password administration can be set up in a variety of ways to meet the needs of different organizations. Some examples are discussed in the following sections.

Change Every Two Weeks

The following shows the password aging information required to establish a new password every 2 weeks (0) and to deny changing the new password for 1 week (/).

1. Here is a typical login/password entry in the **/etc/passwd** file for the typical user **jqu:**

   jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:

2. To cause **jqu** to change the password at least every 2 weeks, but keep it at least for 1 week, you should use the code **0/.** After you edit the **/etc/passwd** file, adding **,0/** to the password field, the entry looks like this:

   jqu:RTKESmMOE2m.E,0/:100:1:J. Q. Username:/usr/jqu:

   After the password entry is changed, **jqu** will have to change the password at the next login and every 2 weeks thereafter.

3. After **jqu's** first login following the change, the system automatically adds the two-character, "last-time-changed" information to the password field.

   jqu:RTKESmMOE2m.E,0/W9:100:1:J. Q. Username:/usr/jqu:

   In this example, **jqu** changed the password in week **W9.**

Change One Time Only

The following shows the password aging information required to establish for one time only a new password when the user next logs in (using the .. code).

1. Here is the typical login/password entry for user **jqu** again:

   jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:

2. To cause **jqu** to change the password at the next login (and to cause this only once), you should use the code **...** After you edit the **/etc/passwd** file, adding **,..** to the password field, the entry looks like this:

   jqu:RTKESmMOE2m.E,..:100:1:J. Q. Username:/usr/jqu:

   After the password entry is changed, **jqu** will have to change the password at the next login only.

3. After **jqu** supplies the new password, the system automatically removes the aging code (,..) from the password field:

   jqu:EFDNLqsFUj.fs:100:1:J. Q. Username:/usr/jqu:

   Note that the encrypted password information has changed.

Change by root Only

The following shows the password aging information required to establish a password for a login so that only the **root** login can change the password (using the **./** code).

1. Here is the typical login/password entry for user **jqu** again:

   jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:

2. To prevent **jqu** from changing the password, you should use the code **./.** After you edit the **/etc/passwd** file, adding **,./** to the password field, the entry looks like this:

   jqu:RTKESmMOE2m.E,./:100:1:J. Q. Username:/usr/jqu:

   Now only **root** can change the password for the **jqu** login. If **jqu** tries to change the password, a permission denied message is displayed.

2.3.3 Locking Unused Logins

If a login is not used or needed, you should do one of two things:

- remove the entry from **/etc/passwd**
- disable (lock) the login

A login is locked by editing the **/etc/passwd** file and changing the encrypted password field to contain one or more characters that are not used by the encryption process. One way to do this is to use the expression **Locked;**. In this expression, the semicolon (;) is an unused encryption character. The text, however, only serves to remind you that the login is locked. Another expression, **not valid**, could also be used to prevent the use of a login because a space is another unused encryption character.

The following line entry from the **/etc/passwd** file shows the "locked" **bin** login.

bin:Locked;:2:2:0000-Admin(0000):/bin:

2.3.4 Special Administrative Passwords

There are two familiar ways to access the system: either via a conventional user login or the **root** login. If these were the only two ways to access the system, however, effective use of the system would have to be curtailed (because **root** would own many directories) or many users would have to know the **root** password (a bad security risk) or the system would be wide open (because **root** would own few directories). All of these conditions are undesirable.

The solution to a good mix of system use and system security is available to you through the use of special system logins and administrative commands that can be password-protected (see Procedure 1.4 for information on doing this). The commands, which are also logins, that can be password-protected perform functions that might be needed by a number of users on your *Series 32000* Computer:

FUNCTION	USE
setup	This command is used to set up the *Series 32000* Computer. Once the machine has been set up, you do not want anyone doing it again without your knowledge.
sysadm	This command allows access to many useful administrative functions that do not require a user to log in as root.
powerdown	This command powers the computer down.
checkfsys	This command initiates a file system check on the specified file systems.
makefsys	This command makes a new file system on the specified media.
mountfsys	This command mounts the specified file system for use.
umountfsys	This command unmounts the specified, previously mounted file system.

The commands above allow access to selected directories and system functions. They may be used as login names at the **login** prompt as well as commands. If you log in to the system with one of these names, the system will execute the command after login and exit to the **login** prompt once you quit or complete the function performed by the command.

Most of these system functions allow a user access to critical portions of the operating system. For this reason, it is recommended that you assign passwords to the commands above. Once you assign passwords to these commands, any user attempting to log on to your *Series 32000* Computer using one of these commands as a login (and any user attempting to execute one of these commands from the shell) is prompted for the password.

It is recommended that the passwords to these commands/logins be known to only a few users.

LOGIN	USE
root	This login has no restrictions on it and it overrides all other logins, protections, and permissions. It allows the user access to the entire operating system. The password for the **root** login should be very carefully protected.
sys	This login has the power of a normal user login over the files it owns, which are in **/usr/src.**
bin	This login has the power of a normal user login over the files it owns, which are in **/bin.**
adm	This login has the power of a normal user login over the object files it owns, which are in **/usr/adm.**
uucp	This login owns the object and spooled data files in **/usr/lib/uucp.**
nuucp	This login is used by remote machines to log into the system and initiate file transfers via **/usr/lib/uucp/uucico.**
rje	This login has an entry in **/etc/passwd.** There are no files currently associated with this login.
daemon	This is the login of the system daemon, which controls background processing.
trouble	This login has the power of a normal user login over the files it owns, which are in **/usr/lib/trouble.**
lp	This login owns the object and spooled data files in **/usr/spool/lp.**

2.4 SET-UID AND SET-GID

The set-user identification (set-UID) and set-group identification (set-GID) bits must be used very carefully if any security is to be maintained. These bits are set through the **chmod**(1) command and can be specified for any executable file. When any user runs an executable file that has either of these bits set, the system gives the user the permissions of the owner of the executable.

System security can be compromised if a user copies another program onto a file with **-rwsrwxrwx** permissions. For example, if the super user (**su**) command has the write access permission allowed for others, anyone can copy the shell onto it and get a password-free version of **su**. The following paragraphs provide a few examples of command lines that can be used to identify the files with a set-UID.

For more information about the set-UID and set-GID bits, see **chmod**(1) and **chmod**(2).

2.4.1 Check Set-UIDs Owned by Root

The following command line lists all set-UID programs owned by **root**. The results are mailed to **root**. All mounted paths are checked by this command starting at **/**. Any surprises in **root's** mail should be investigated.

```
# find / —user root —perm —4100 —exec ls —l { } \; | mail root
you have mail
# mail
From root Mon Aug 27 07:20 EDT 1984

-r- sr-xr-x   1 root    bin     38836          Aug 10 16:16 /usr/bin/at
-r- sr-xr-x   1 root    bin     19812          Aug 10 16:16 /usr/bin/crontab
-r- sr-xr-x   1 root    bin     27748          Aug 10 16:16 /usr/bin/shl
--- s--x--x   1 root    sys     46040          Aug 10 15:18 /usr/bin/ct
-r- sr-xr-x   1 root    sys     12092          Aug 11 01:29 /usr/lib/mv_dir
-r- sr-s r-x  1 root    bin     33208          Aug 10 15:55 /usr/lib/lpadmin
-r- sr-s r-x  1 root    bin     38696          Aug 10 15:55 /usr/lib/lpsched
--- s--x---   1 root    rar     45376          Aug 18 15:11 /usr/rar/bin/sh
-r- sr-xr-x   1 root    sys     11416          Aug 11 01:26 /bin/mkdir
-r- sr-xr-x   1 root    sys     11804          Aug 11 01:26 /bin/rmdir
-r- sr-xr-x   1 root    bin     12524          Aug 11 01:27 /bin/df
-rwsr-xr-x    1 root    sys     21780          Aug 11 01:27 /bin/newgrp
-r- sr-s r-x  1 root    sys     23000          Aug 11 01:27 /bin/passwd
-r- sr-xr-x   1 root    sys     23824          Aug 11 01:27 /bin/su

? q (quit)
#
```

In this example, an unauthorized user (**rar**) has made a personal copy of **/bin/sh** and has made it set-UID to **root**. This means that **rar** can execute **/usr/rar/bin/sh** and become the super user.

2.4.2 Check Set-UIDs in the Root File System

The following command line reports all files with a set-UID for the root file system. The **ncheck**(1M) command, by itself, can be used on a mounted or unmounted file system. The normal output of the **ncheck** —**s** command includes special files. Here, the **grep** command is used to remove device files from the output. The filtering done in this example to remove the device files is applicable only for the root file system (**/dev/dsk/c0d0s0**). The output of the modified **ncheck** is used as an argument to the **ls** command. The use of the **ls** command is possible only if the file system is mounted.

```
# ls —l `ncheck —s /dev/dsk/c0d0s0 | cut —f2 | grep —v dev`

-r- sr-xr-x   1 root   bin     12524   Aug   11   01:27   /bin/df
-rwxr-sr-x    1 root   sys     32272   Aug   10   15:53   /bin/ipcs
-r- xr-sr-x   2 bin    mail    32852   Aug   11   01:28   /bin/mail
-r- sr-xr-x   1 root   sys     11416   Aug   11   01:26   /bin/mkdir
-rws r-xr-x   1 root   sys     21780   Aug   11   01:27   /bin/newgrp
-r- sr-sr-x   1 root   sys     23000   Aug   11   01:27   /bin/passwd
-r- xr-sr-x   1 bin    sys     27964   Aug   11   01:28   /bin/ps
-r- xr-sr-x   2 bin    mail    32852   Aug   11   01:28   /bin/rmail
-r- sr-xr-x   1 root   sys     11804   Aug   11   01:26   /bin/rmdir
-r- sr-xr-x   1 root   sys     23824   Aug   11   01:27   /bin/su
-r- xr-sr-x   1 bin    sys     21212   Aug   10   16:08   /etc/whodo

#
```

In this example, nothing looks suspicious.

2.4.3 Check Set-UIDs in Other File Systems

The following command line entry shows the use of the **ncheck** command to examine the **usr** file system (**/dev/dsk/c0d0s2**, assuming a single-disk system with default partitioning) for files with a set-UID. In this example, the complete path names for the files start with **/usr**. **/usr** is not part of the **ncheck** output.

```
# ncheck —s /dev/dsk/c0d0s2 | cut —f2
/dev/dsk/c0d0s2:
/bin/at
/bin/crontab
/bin/shl
/bin/sadp
/bin/timex
/bin/cancel
/bin/disable
/bin/enable
/bin/lp
/bin/lpstat
/bin/ct
/bin/cu
/bin/uucp
/bin/uuname
/bin/uustat
/bin/uux
/lib/mv_dir
/lib/expreserve
/lib/exrecover
/lib/accept
/lib/lpadmin
/lib/lpmove
/lib/lpsched
/lib/lpshut
/lib/reject
/lib/mailx/rmmail
/lib/sa/sadc
/lib/uucp/uucico
/lib/uucp/uusched
/lib/uucp/uuxqt
/rar/bin/sh
#
```

In this example, the **/usr/rar/bin/sh** should be investigated.

Chapter 3

USER SERVICES

3.1 INTRODUCTION

This chapter deals with providing a variety of services to the users of your *Series 32000* Computer system.

- Login administration

  Assigning user and group IDs to persons authorized to be users of your system. Maintaining the **/etc/passwd** and **/etc/group** files.

- The users' environments

  Setting up a master profile and helping users develop individual profiles. Establishing environment variables.

- User communications services

  Establishing and maintaining such services as message-of-the-day, news, mail.

- Anticipating user requests

  Developing an organized plan for responding to user problems.

3.2 LOGIN ADMINISTRATION

3.2.1 Adding Users

Before users are permitted to log in to your system, they must be listed in the **/etc/passwd** file. The **adduser** selection from the **sysadm usermgmt**(1) menu leads you through a series of prompts that create an entry in the **/etc/passwd** file (see Procedure 2.1). If you prefer, you can make the necessary changes to the **passwd** file yourself using an editor. You need to login as **root** to do this; **/etc/passwd** is generally installed as a read-only file. An entry in the **passwd** file consists of a single line with seven colon-separated fields.

abc:gOQ3xsv05bWuM,9/TA:103:123:Allen B. Cipher:/usr/abc:

The fields are:

login name	A valid name for logging onto the system. A login name is from three to six characters; the first character must be alphabetic. It is usually chosen by the user.
password	The encrypted form of the password, if any, associated with the login name in the first field. All encrypted passwords occupy 13 bytes. The actual password can be a maximum of 8 characters. At least one character must be numeric. This is to discourage users from choosing ordinary words as passwords. When you add a user to the file you may use a default

password, such as **passwd9**, and instruct the user to change it at the first login. Following the encrypted password, separated by a comma, there may be a field that controls password aging. See Section 2.3.1.

user ID — The user ID number (**uid**) is between 0 and 50,000. The number must not include a comma. Numbers below 100 are reserved. User ID 0 is reserved for the super-user. The System Administration menu package does not permit you to specify a number below 100 when adding a user.

group ID — The same conditions apply to the group ID (**gid**) number as to the **uid**, except that the group ID 1 is reserved for the "other" group.

account — The account is the name of and optional additional information about the user. There is no required format for this field.

home directory — The home directory is where the user is placed upon logging in. The name is usually the same as the login name, preceded by a parent directory such as **/usr**. The **modadduser** menu of the System Administration package allows you to specify the default parent directory. The home directory is the origination point of the user's directory tree.

program — This is the name of a program invoked at the time the user logs in. If the field is empty, the default program is **/bin/sh**. This field is most commonly used to invoke a special shell, such as **/bin/rsh** (restricted shell).

As noted above, the password field may contain a subfield that controls the aging of passwords. A description of how the process works can be found in Chapter 2 and in **passwd**(4) in the *Programmer's Reference Manual.* The effect is to force users periodically to select a new password. If password aging is not implemented, a user can keep the same password indefinitely.

If you inspect the **/etc/passwd** file on your *Series 32000* Computer you will see several commands listed among the user login names. These are commands, such as **makefsys**(1M) or **sysadm**(1), that can have passwords assigned to them.

3.2.2 Changing or Deleting Password Entries

As with adding users, there are **sysadm usermgmt** menu selections for changing or deleting user entries from the **/etc/passwd** file (see Procedures 2.2 and 2.3). You have the option, however, of using an editor to make the changes.

Every so often a user will forget his or her password. When that happens (let's say to user **abc**), you can login as **root** and enter the command:

```
# passwd abc                          (The # prompt shows you are root.)
New password: passwd9                 (The password entered is not echoed.)
Re-enter new password: passwd9
```

Since you did this as the super-user (**root**), you were not prompted for the old password. The command changes **abc**'s password to **passwd9**. You should make sure the user changes the password immediately.

When you delete a login from **/etc/passwd** using the **sysadm usermgmt** menu, all of the user's files and directories are removed. If you remove an entry from the **passwd** file using an editor, you have only removed the entry. The user's files remain.

3.2.3 Group IDs

Group IDs are a means of establishing another level of ownership of and access to files and directories. Users with some community of interest can be identified as members of the same group. Any file created by a member of the group carries the group ID as a secondary identification. By manipulating the permissions field of the file, the owner (or someone with the effective user ID of the owner) can grant read, write, or execute privileges to other group members.

Information about groups is kept in the **/etc/group** file. A sample entry from this file is shown and explained below:

```
prog::123:jqp,abc
```

Each entry is one line; each line has the following fields:

group name	The group name can be from 3 to 8 characters, the first of which must be alphabetic.
password	The password field should not be used.
group ID	The group ID is a number from 0 to 50,000. The number must not include a comma. Numbers below 100 are reserved.
login names	The login names of group members are in a comma-separated list. A login name should be a member of no more than one group. There is nothing to prevent a user from having more than one login name, however, as long as each is unique within the system.

3.3 THE USER'S ENVIRONMENT

The key element in establishing an environment in which users can successfully communicate with the computer is the profile. Profiles are of two types:

1. The system profile.

   This is an ASCII text file, **/etc/profile**, that contains commands, shell procedures, and environment variables. Whenever a user logs in, the **login** process executes this file.

2. An individual user's profile.

   This is an executable commands file, **.profile**, that may reside in a user's home directory. The individual profile can contain additional commands and variables that further customize a user's environment. If one exists, it too is executed at login time, after the execution of **/etc/profile**.

A sample **/etc/profile** is shown in Figure 3-1.

Several interesting things are contained in the profile:

- Some environment variables are exported (see Section 3.3.1).
- A file named **/etc/motd** is **cat**-ted (see Section 3.4.1).
- If the user is not **root**, the names of news items are displayed (**news —n**; see Section 3.4.2).
- If the user has mail (**mail —e**), a message about it is displayed (see Section 3.4.4).

For information on the shell programming commands used in Figure 3-1, see **sh**(1) in the *User's Reference Manual.*

3.3.1 Environment Variables

An array of strings called the environment is made available by **exec**(2) when a process begins. Since **login** is a process, the array of environment strings is made available to it. An example of a typical array of strings is shown in Figure 3-2.

The environment variables shown in Figure 3-2 give values to 12 names for user **abc**. Other programs make use of the information. For example, the user's terminal is defined as a 5420 (TERM=5420). When the user invokes the editor **vi**(1), **vi** checks the file referenced by TERMINFO (**/usr/lib/terminfo**) where it learns the characteristics of a 5420 terminal (such as the 24-line screen). New strings can be defined at any time. By convention they are defined with the variable in uppercase, followed by an equal sign, followed by the value. Once defined, an environment variable can be made global for the user through the **export** statement. The individual **.profile** file can contain whatever the user wants.

```
# The profile that all logins get before using their own .profile.
trap "" 2 3
export LOGNAME
# Login and —su shells get /etc/profile services.
# —rsh is given its environment in its .profile.
case "$0" in
—su)
        export PATH
        ;;
—sh)
        export PATH
        # Allow the user to break the Message of the Day only.
        trap "trap " 2" 2
        cat —s /etc/motd
        trap "" 2
        if mail —e
        then
        echo "you have mail"
        fi
        if [ ${LOGNAME} != root ]
        then
        news —n
        fi
        ;;
esac
umask 022
trap 2 3
```

Figure 3-1. A Default /etc/profile

```
PS1=$
LOGNAME=abc
PWD=/usr/abc
HOME=/usr/abc
PATH=:/bin:/usr/bin:/usr/lbin
SHELL=/bin/sh
MAIL=/usr/mail/abc
TERM=5420
PAGER=pg
TZ=EST5EDT
TERMINFO=/usr/lib/terminfo
EDITOR=vi
```

Figure 3-2. Environment Array for a Typical User

3.3.2 Umask

A system default controls the permissions mode of any files or directories created by a user. The *Series 32000* Computer has default values of 666 for files and 777 for directories. That means that for files everyone automatically gets read and write permission. For directories, everyone gets read, write, and execute permission. (Execute permission on a directory means the ability to **cd** to the directory and to copy files from it.)

Users frequently set up a user mask in their **.profile** by means of the **umask**(1) command. **umask** alters the default permission levels by a specified amount. For example,

umask 027

leaves the permission level for owner unchanged, lowers the permission level for group by 2, and reduces the permissions for others to zero. The system default was 666; this user mask changes it to 640, which translates into read and write permission for the owner, read permission for the group, and no permission for others.

There may be a **umask** command in **/etc/profile.** If there is, it does not change a user's ability to put one in **.profile.**

3.3.3 Default Shell and Restricted Shell

Generally, when a user logs in the default program that is started is **/bin/sh.** There may be cases, however, where a user needs to be given a restricted shell.

A restricted shell is one where the user is not allowed to

- change directories
- change the value of $PATH
- specify pathnames or command names containing a slash (/). That is, the user of a restricted shell may not access files or directories other than the present working directory or those included in $PATH
- redirect output

The restrictions are enforced after **.profile** has been executed.

The administrator can use a restricted shell strategy to limit certain users to the execution of a small number of commands or programs. By setting up a special directory for executables (**/usr/rbin,** for example), and controlling PATH so it only references that directory, the administrator can restrict the user's activity in whatever way is appropriate.

3.4 USER COMMUNICATIONS SERVICES

Several ways of communicating with and among users are available in the GENIX V.3 system. Some of the most frequently used are described in this section.

3.4.1 Message of the Day

Items of broad interest that you want to make available to all users can be put in the **/etc/motd** file. The contents of **/etc/motd** are displayed on the user's terminal as part of the login process. The login process executes a file called **/etc/profile,** which is an executable shell script that, among other things, commonly contains the command

cat /etc/motd

Any text contained in **/etc/motd** is displayed for each user each time the user logs in. For this information to have any impact on users, you must take pains to use it sparingly and to clean out outdated announcements. A typical use for the Message of the Day facility might be

5/30: The system will be unavailable from 6-11pm Thursday, 5/30 - preventive maintenance.

Part of the preventive maintenance should be to remove the notice from **/etc/motd.**

3.4.2 News

Another electronic bulletin board facility is the **/usr/news** directory and the **news**(1) command. The directory is used to store announcements in text files, the names of which are usually used to provide a clue to the content of the news item. The **news** command is used to print the items on your terminal.

The **/etc/profile** file is also used to inform users about news items. A typical **/etc/profile** contains the line

news —n

The **—n** argument causes the names of files in the **/usr/news** directory to be printed on a user's terminal as the user logs in. Item names are displayed only for current items, that is, items added to the **/usr/news** directory since the user last looked at the news. The idea of currency is implemented like this: when you read a news item an empty file named **.news_time** is written in your login directory. As with any other file, **.news_time** carries a time stamp indicating the date and time the file was created. When you log in, a comparison is made between the time stamp of your **.news_time** file and time stamp of items in **/usr/news.**

Unlike the Message of the Day where users have no ability to turn the message off, with **news** users have a choice of several possible actions:

read everything	If the user enters the command, **news** with no arguments, all news items posted since the last time the user typed in the command are printed on the user's terminal.
select some items	If the **news** command is entered with the names of one or more items as arguments, only those items selected are printed.
read and delete	After the **news** command has been entered, the user can stop any item from printing by pressing the DELETE key. Pressing the DELETE key twice in a row stops the program.
ignore everything	If the user is too busy to read announcements at the moment, they can safely be ignored. Items remain in **/usr/news** until removed. The item names will continue to be displayed each time the user logs in.
flush all items	If the user simply wants to eliminate the display of item names without looking at the items, a couple of techniques will work: **$ touch .news_time** updates the time-accessed and time-modified fields of the **$ news > /dev/null** prints the news items on the null device.

3.4.3 Write to All Users

The ability to write to all logged-in users, via the **wall**(1M) command, is an extension of the **write**(1) command. It is fully effective only when used by the super-user. While **wall** is a useful device for getting urgent information out quickly, users tend to find it annoying to have messages print out on their terminal right in the middle of whatever else is going on. The effect is not destructive, but is somewhat irritating. Many users guard against this distraction by including the command

mesg n

in their **.profile.** This blocks other ordinary users from interjecting a message into your **stdout.** The **wall** command, when used by the super-user, overrides the **mesg n** command. It is best to reserve this for those times when you as the system administrator need to ask users to get off the system. The use of the **wall** command is described in Procedure 2.5.

3.4.4 Mail and Mailx

The GENIX V.3 system offers two electronic mail utilities through which users can communicate among themselves. If your system is connected to others by networking facilities, **mail**(1) and **mailx**(1) can be used to communicate with persons on other systems.

Mail is the basic utility for sending messages. **Mailx** uses **mail** to send and receive messages, but adds to it a multitude of extras that are useful for organizing messages into storage files, adding headers, and many other functions.

When **mailx** is used, a set-up file is helpful. You can find a description of how to use a **.mailrc** set-up file in the **mailx**(1) pages of the *User's Reference Manual.*

3.5 ANTICIPATING USER REQUESTS

As the system administrator you can expect users to look to you to help solve any number of problems. In addition to the system log described in Chapter 4, you will find it helpful to keep a user trouble log. The problems that users run into fall into patterns. If you keep a record of how problems were resolved, you will not have to start from scratch when a problem recurs.

3.5.1 Trouble Reporting

Another technique that is strongly recommended is an organized way for users to report problems. Figure 3-3 shows a sample Trouble Report that can be used to record and keep track of system problems.

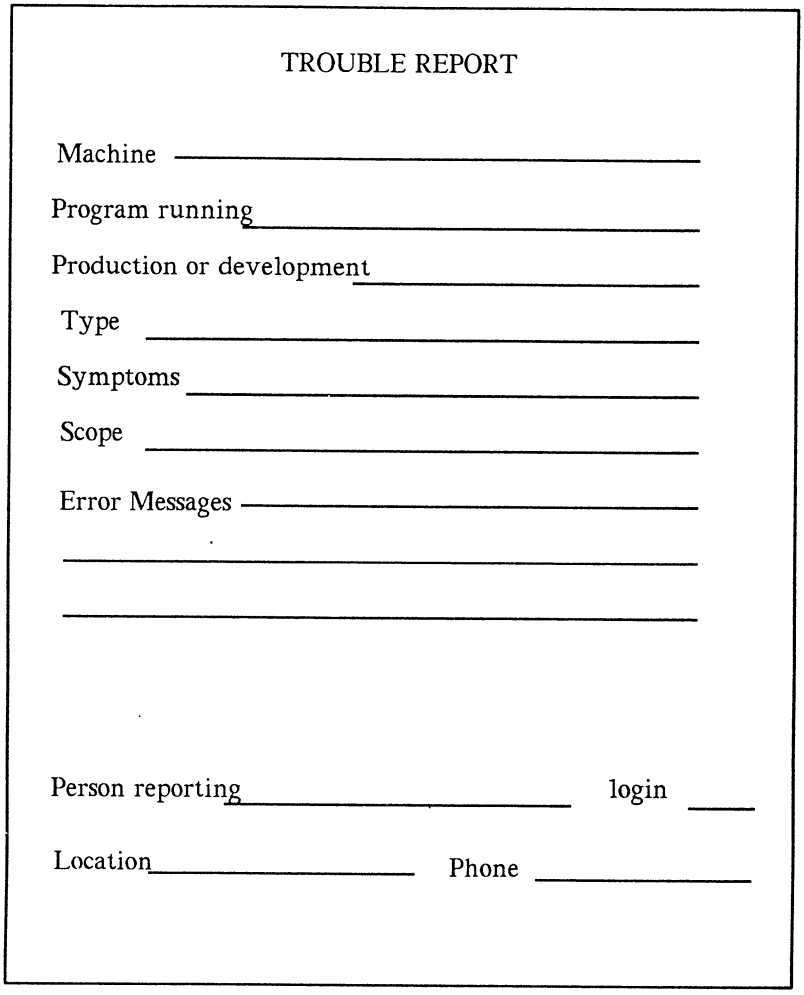


Figure 3-3. Sample Trouble Report

Chapter 4

PROCESSOR OPERATIONS

4.1 INTRODUCTION

This chapter deals with the day-to-day operations of your *Series 32000* Computer system.

- General operating policy

  Guidelines for balancing the needs of system maintenance and the interests of your user community; suggestions for record keeping; lists of important administrative directories and files.

- Operating Levels

  Definition of the operating levels of the system; how they are controlled.

- Running firmware programs

  Running firmware-resident and bootable programs from the system firmware.

- Hardware diagnostic information

  Using the hardware diagnostic monitor to locate system troubles.

4.1.1 General Operating Policy

Many administrative tasks require the system to be shut down to a run level other than the multi-user state (see the discussion on Operating Levels below). This means that conventional users cannot access the system. When the machine is taken out of the multi-user state, the users on the machine at the time are requested to log off. You should do these types of tasks when they will interfere the least with the activities of the user community.

Sometimes situations arise that require the system to be taken down with little or no notice provided to the users. Try to provide the user community as much notice as possible about events affecting the use of the machine. When the system must be taken out of service, also tell the users when to expect the system to be available. Use the news (**/etc/news/***headline*) and the Message of the Day (**/etc/motd**) to keep users informed about changes in hardware, software, policies, and procedures.

At your discretion, the following items should be done as prerequisites for any task that requires the system to leave the multi-user state.

1. When possible, schedule service-affecting tasks to be done during periods of low system use. For scheduled actions, use the Message of the Day (**/etc/motd**) to inform users of future actions.

2. Check to see who is logged in before taking any actions that would affect a logged-in user. The **/etc/whodo** and **/bin/who** commands can be used to see who is on the system.

3. If the system is in use, provide the users advanced warning about changes in system states or pending maintenance actions. For immediate actions, use the **/etc/wall** command to send a broadcast message announcing that the system will be taken down at a given time. Give the users a reasonable amount of time to terminate their activities and log off before taking the system down.

4.1.2 Maintaining a System Log

In a multi-user environment it is strongly recommended that a complete set of records be maintained. A system log book can be a valuable tool when trouble shooting transient problems or when trying to establish system operating characteristics over a period of time. Some of the things that you should consider entering into the log book are:

- maintenance records (dates and actions)
- printouts of error messages and diagnostic phases
- equipment and system configuration changes (dates and actions)

The format of the system log and the types of items noted in the log should follow a logical structure. Think of the log as a diary that you update on a periodic basis. To a large measure, how you use your system will dictate the form and importance of maintaining a system log.

4.1.3 Administrative Directories and Files

This section briefly describes the directories and files that are frequently used by a system administrator. For more detail about the purpose and contents of these directories and files, see Appendix B. For additional information on the formats of the system files, refer to Section 4 of the GENIX V.3 manual pages in the *Programmer's Reference Manual.*

4.1.4 Root Directories

The directories of the **root** file system (/) are as follows.

bck	Directory used to mount a backup file system for restoring files.
bin	Directory that contains public commands.
dev	Directory containing special files that define all of the devices on the system.
etc	Directory that contains administrative programs and tables.
install	Directory used by the System Administration Menu package to mount utilities packages for installation and removal (**/install** file system).
lib	Directory that contains public libraries.
lost+found	Directory used by **fsck**(1M) to save disconnected files.
mnt	Directory used to temporarily mount file systems during the restore of the operating system from floppy disks.

save	Directory used by the System Administration Menu package for saving data on diskettes.
stand	Directory used for stand-alone utilities.
tmp	Directory used for temporary files.
usr	Directory used to mount the **/usr** file system. (See Chapter 6 for a description of this file system.)

4.1.5 Important System Files

The following files and directories are important in the administration of the *Series 32000* Computer.

/etc/checklist	File used to define a default list of file system devices to be checked by **/etc/fsck.**
/etc/fstab	File used to specify the file system(s) to be mounted by **/etc/mountall** and remote file systems to be mounted by **/etc/rmountall.**
/etc/gettydefs	File containing information used by **/etc/getty** to set the speed and terminal settings for a line.
/etc/group	File describing each group to the system.
/etc/init.d	Directory containing executable files used in upward and downward transitions to all system run levels. These files are linked to files beginning with **S** (start) or **K** (stop) in **/etc/rc***n***.d,** where *n* is replaced by the appropriate run level.
/etc/inittab	File containing the instructions to define the processes created or terminated by **/etc/init** for each initialization state.
/etc/master.d	Directory containing files that define the configuration of hardware devices, software drivers, system parameters, and aliases.
/etc/motd	File containing a brief Message of the Day, output by **/etc/profile.**
/etc/passwd	File identifying each user to the system.
/etc/profile	File containing the standard (default) environment for all users.
/etc/rc0	File executed by **/etc/shutdown** that executes shell scripts in **/etc/rc0.d** and **/etc/shutdown.d** directories for transitions to system run-levels 0, 5, and 6.
/etc/rc0.d	Directory containing files executed by **/etc/rc0** for transitions to system run-levels 0, 5, and 6. Files in this directory are linked from files in the **/etc/init.d** directory and begin with either a **K** or an **S. K** indicates processes that are stopped, and **S** indicates processes that are started when entering run-levels 0, 5, or 6.
/etc/rc2	File executed by **/etc/init** that executes shell scripts in **/etc/rc2.d** and **/etc/rc.d** on transitions to system run-level 2.

/etc/rc2.d Directory containing files executed by **/etc/rc2** for transitions to system run-levels 2 and 3. Files in this directory are linked from files in the **/etc/init.d** directory and begin with either a **K** or an **S**. **K** indicates processes that should be stopped and **S** indicates processes that should be started when entering run-levels 2 or 3.

/etc/rc.d Directory containing executable files that do the various functions needed to initialize the system to run-level 2; they are executed when **/etc/rc2** is run. (Files contained in this directory prior to the GENIX V.3 System were moved to **/etc/rc2.d**. This directory is only maintained for compatibility.)

/etc/rc3 File executed by **/etc/init** that executes shell scripts in **/etc/rc3.d** on transitions to system run-level 3 (Remote File Sharing state).

/etc/rc3.d Directory containing files executed by **/etc/rc3** for transitions to system run-level 3 (Remote File Sharing mode). Files in this directory are linked from the **/etc/init.d** directory and begin with either a **K** or an **S**. **K** indicates processes that should be stopped, and **S** indicates processes that should be started when entering run-level 3.

/etc/rstab File used to specify the Remote File Sharing resources from your machine that are automatically offered to remote machines upon entering system run level 3 (Remote File Sharing state).

/etc/save.d Directory containing files that are used by the System Administration Menu commands associated with backing-up data on floppy diskettes.

/etc/shutdown File containing a shell script that gracefully shuts down the system in preparation for system backup or for scheduled downtime.

/etc/shutdown.d Directory containing executable files that do the various functions needed to transition the system to the single-user state (run-levels **1**, **s**, or **S**). (Files contained in this directory prior to the GENIX V.3 System were moved to **/etc/rc0.d**. This directory is only maintained for compatibility.)

/etc/TIMEZONE File used to set the time zone shell variable TZ.

/etc/utmp File containing the information on the current run-state of the system.

/etc/wtmp File containing a history of system logins.

/usr/adm/sulog File containing a history of **su** command usage. This file should be checked periodically for size.

/usr/lib/cron/log File containing a history of all the actions taken by **/etc/cron**. This file should be checked periodically for size.

/usr/lib/help/HELPLOG
File containing a history of all the actions taken by the **/usr/bin/help** (if it is enabled on the system).

/usr/lib/spell/spellhist
File containing a history of all words that **spell** fails to match (if the Spell Utilities are installed on the system).

/usr/news Directory containing news files. This directory should be checked periodically, and old files should be discarded.

/usr/options Directory containing files that identify the utilities that are installed on the system.

/usr/spool/cron/crontabs
Directory containing crontab files for the **adm, root,** and **sys** logins and ordinary users listed in **cron.allow.**

Each of these files is described in more detail in Appendix B.

4.2 OPERATING LEVELS

4.2.1 General

After you have set up your *Series 32000* Computer for the first time (plugging it in, hooking all the hardware together, installing the system software, booting it, and running the setup programs) as documented in the *Series 32000* Computer system manual, you and other users can use the system. Whenever you turn the computer on (including the first time), the system comes up in the monitor. To boot the GENIX V.3 system, type **k** and return. This monitor command brings the system up in the multi-user environment in which

- The file systems are mounted.
- The **cron** daemon is started for scheduled tasks.
- The basic networking functions of **uucp** (if added to the system) are available for use.
- The spooling and scheduling functions of the LP package (if added to the system) are available for use.
- Users can log in. The **gettys** are spawned on all connected terminal lines listed in **/etc/inittab** to have **gettys** respawned. (**Gettys** are not on when the system is installed. You have to turn them on yourself.)

This is defined as the multi-user state. It is also referred to as **init** state 2 because all of the activities of initializing the system are under the control of the **init** process. The "2" refers to entries in the special table **/etc/inittab** used by **init** to initialize the system to the multi-user state.

Not all activities, however, can be performed in the multi-user state. For example, if you were able to unmount a file system while users were accessing it, you would cause a lot of data to be lost. Hence, for unmounting and other system administration tasks, there is a need for another state, the single-user state.

The single-user state is an environment in which only the console has access to the system and the root file system alone is mounted. You are free to do tasks that affect the file systems and the system configuration because you are the only one on the system.

There are other system states (see Table 4-1), but first a note of clarification. One of the more confusing things about the discussion of system states is that there are many terms used to identify the same thing: the particular operating level of the system.

Here is a list of frequently encountered synonyms:

- run state
  run level
  run mode
- init state
  system state

Likewise, each system state may be referred to in a number of ways, for example:

- single-user
- single-user mode
- run level 1, and so on

In any case, each state or run level clearly defines the operation of the computer. Table 4-1 defines each of them as they pertain to the *Series 32000* Computer.

4.2.2 How init Controls the System State

GENIX V.3 systems always run in one state or another. The actions that cause the various states to exist are under the control of the **init** process, which is the first general process created by the system at boot time. It reads the file **/etc/inittab**, which defines exactly which processes exist for which run level.

In the case of the multi-user state (run level 2), **init** scans the file for entries that have a tag for the run level (the tag is a 2) and executes everything after the last colon (:) on the line containing the tag. These tags represent the run levels in the table in Figure 4-1.

TABLE 4-1. SYSTEM STATES

RUN LEVEL	DESCRIPTION
0	Power-down state.
1, s, or S	Single-user mode is used to install/remove software utilities, run file system backups/restores, and to check file systems. Though **s** and **1** are both used to go to single user state, **s** only kills processes spawned by **init** and does not unmount file systems. State **1** unmounts everything except root and kills all user processes, except those that relate to the console.
2	Multi-user mode is the normal operating mode for the system. The default is that the **root** (**/**) and user (**/usr**) file systems are mounted in this mode. When the system is powered up it is put in multi-user mode.
3	Multi-user/Remote File Sharing mode is used to start Remote File Sharing (RFS), connect your computer to an RFS network, mount remote resources, and offer your resources automatically. (See Chapter 11.)
4	User defined run state.
5	Firmware mode is used to run programs in the **root** file system. An example is executing **/unix** to reboot the system.
6	Halt and reboot the operating system. The system comes up in multi-user mode (run-level 2).

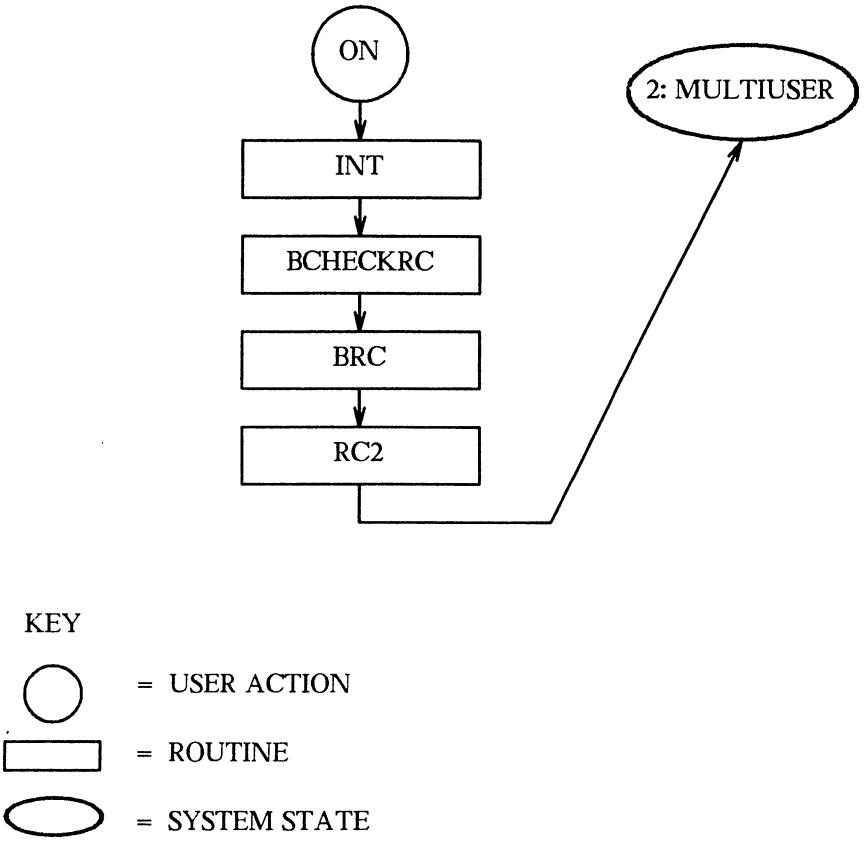


Figure 4-1. A Look at System Initialization

If you look at your **/etc/inittab**, you'll see something that looks like Figure 3-1. (It is most unlikely that yours will look exactly like this one; **/etc/inittab** changes from one configuration to another.)

NOTE: If **/etc/inittab** was removed by mistake and is missing during shutdown, **init** will enter the single-user state (**init s**). While entering single-user state, **/usr** will remain mounted and processes not spawned by **init** will continue to run. You should replace **/etc/inittab** before changing states again.

```
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
nu::sysinit:/etc/newunix </dev/console >/dev/console 2>&1
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
is:2:initdefault:
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
RB:6:wait:echo "0he system is being restarted." >/dev/console 2>&1
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
ct:234:respawn:/etc/getty contty contty
he:234:respawn:sh -c 'sleep 20 ; exec /etc/hdelogger >/dev/console 2>&1'
10:234:respawn:/etc/getty /dev/tty10 9600
11:234:off:/etc/getty tty11 9600
12:234:off:/etc/getty tty12 9600
13:234:off:/etc/getty tty13 9600
14:234:off:/etc/getty tty14 9600
15:234:off:/etc/getty tty15 9600
16:234:off:/etc/getty tty16 9600
17:234:off:/etc/getty /dev/tty17 9600 >/dev/console 2>&1
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
ct:234:respawn:/etc/getty contty contty
he:234:respawn:sh -c 'sleep 20 ; exec /etc/hdelogger >/dev/console 2>&1'
10:234:respawn:/etc/getty /dev/tty10 9600
11:234:off:/etc/getty tty11 9600
12:234:off:/etc/getty tty12 9600
13:234:off:/etc/getty tty13 9600
```

The format of each line is

id:level:action:process

- *id* is one or two characters that uniquely identify an entry.
- *level* is zero or more numbers and letters (**0** through **6**, **s**, **a**, **b**, and **c**) that determines what *level*(s) *action* is to take place in. If *level* is null, the *action* is valid in all levels.
- *action* can be one of the following:

sysinit	run *process* before **init** sends anything to the system console (Console Login:).
bootwait	start *process* the first time **init** goes from single-user to multi-user state after the system is booted. (If **initdefault** is set to **2**, the process will run right after the boot.) *init* starts the process, waits for its termination and, when it dies, does not restart the process.
wait	when going to *level*, start *process* and wait until it's finished.
initdefault	when **init** starts, it will enter *level*; the *process* field for this *action* has no meaning.
once	run *process* once and don't start it again if it finishes.
powerfail	tells **init** to run *process* whenever a direct powerdown of the computer is requested.
respawn	if process does not exist, start it, wait for it to finish, and then start another.
ondemand	synonymous with **respawn**, but used only with *level* **a**, **b**, or **c**.
off	when in *level*, kill process or ignore it.

- *process* is any executable program, including shell procedures.
- # can be used to add a comment to the end of a line. Everything after a # on a line will be ignored by init.

When changing levels, **init** kills all processes not specified for that level. We'll go through more manageable pieces of this table below to get a clearer idea of how the system is controlled by **init**.

4.2.3 A Look at Entering the Multi-User State

4.2.3.1 Powering Up

When you power up your system, it will enter the system monitor by default. Then you must boot the operating system which will bring you up in the multi-user environment. In effect, going to the multi-user state follows these broad lines (see Figure 4-1):

1. You turn on the computer.
2. The system monitor is loaded.
3. You boot the GENIX V.3 operating system by entering **k** and returning.
4. The operating system is loaded and the early system initializations are started by **init.**
5. The run level change is prepared by the **/etc/rc2** procedure.
6. Finally the system is made public via the spawning of **gettys** along the terminal lines.

4.2.3.2 Early Initialization

Just after the operating system is first loaded into core the **init** process is created. It immediately scans **/etc/inittab** for entries of the type **sysinit:**

```
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
nu::sysinit:/etc/newunix </dev/console >/dev/console 2>&1
```

They are executed in sequence and perform the necessary early initializations of the system. Note that each entry indicates a standard input/output relationship with **/dev/console.** This is the way communication is established with the system console before the system has been brought to the multi-user state.

4.2.3.3 Preparing the Run Level Change

Now the system must be placed in a particular run level. First, **init** scans the table to find an entry that specifies an *action* of the type *initdefault*. If it finds one, it uses the run level of that entry as the tag it will use to select the next entries to be executed. In our sample **/etc/inittab**, the **initdefault** entry specifies run level 2 (the multi-user state) as the level to select and execute other entries:

```
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
is:2:initdefault:
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
ct:234:respawn:/etc/getty contty contty
he:234:respawn:sh -c 'sleep 20 ; exec /etc/hdelogger >/dev/console 2>&1'
10:234:respawn:/etc/getty /dev/tty10 9600
co:234:respawn:/etc/getty console console
ct:234:respawn:/etc/getty contty contty
he:234:respawn:sh -c 'sleep 20 ; exec /etc/hdelogger >/dev/console 2>&1'
10:234:respawn:/etc/getty /dev/tty10 9600
```

The other entries shown above specify the actions necessary to prepare the system to change to the multi-user run level. First, **/etc/rc2** is executed. It executes all files in **/etc/rc2.d** that begin with the letter **S**. It then executes all the files in the **/etc/rc.d** directory, accomplishing (among other things) the following:

- sets up and mounts the file systems
- starts the **cron** daemon
- displays the current system hardware configuration
- makes **uucp** available for use, if installed
- makes line printer (lp) system available for use, if installed
- starts a **getty** for the Console
- starts the hard disk error logging daemon
- starts **getty** on the lines connected to the ports
- indicates (ttys)

At this moment, the full multi-user environment is established, and your system is available for users to log in (see Procedures 3.1 and 3.4).

4.2.4 A Look at the System Life Cycle

4.2.4.1 Changing Run Levels

In effect, changing run levels follows these broad lines (see Figure 4-2):

1. The system administrator enters a command that directs **init** to execute entries in **/etc/inittab** for a new run level.
2. Key procedures, such as **/etc/shutdown, /etc/rc0, /etc/rc2,** and **/etc/rc3,** are run to initialize the new state.
3. The new state is reached. If it is either state 1 or 5, the system administrator can proceed.

4.2.4.2 Run Level Directories

Run levels 0, 2, and 3 each have a directory of files that are executed in transitions to and from that level. These directories are **rc0.d, rc2.d,** and **rc3.d,** respectively. All files in these directories are linked to files in **/etc/init.d.** The run-level filenames look like this:

S*00name*

or

K*00name*

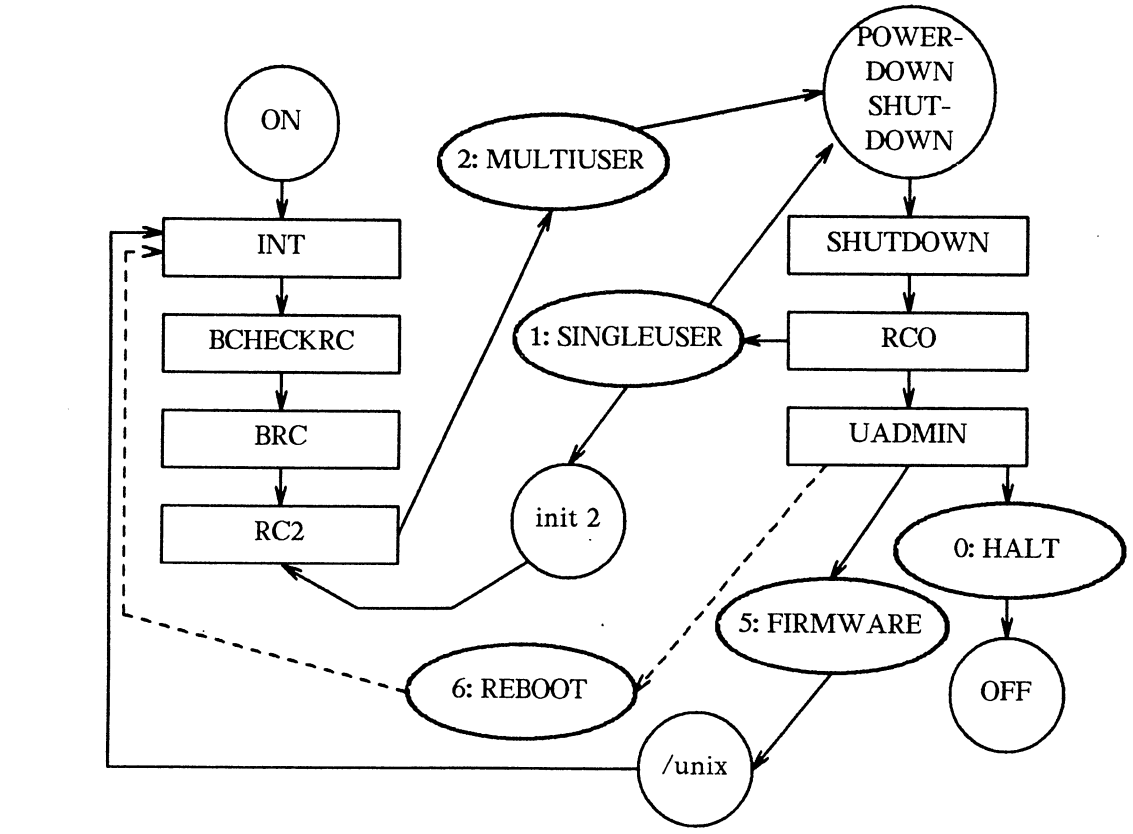
The filenames can be split into three parts:

S or **K**	The first letter defines whether the process should be started (**S**) or stopped (**K**) upon entering the new run level.
00	The next two characters are a number from 00 to 99. They indicate the order in which the files will be started (S00, S01, S02, etc.) or stopped (K00, K01, K02, etc).
name	The rest of the filename is the **/etc/init.d** filename this file is linked to.

For example, the **init.d** file **cron** is linked to the **rc2.d** file and **rc0.d** file **K70cron.** When you enter **init 2,** this file is executed with the **start** option: **sh S75cron start.** When you enter **init 0,** this file is executed with the **stop** option: **sh K70cron stop.** This particular shell script will execute **/usr/bin/cron** when run with the **start** option and **kill** the **cron** process when run with the **stop** option.

Because these files are shell scripts, you can read them to see what they do. You can modify the files, though it is preferable to add your own since the delivered scripts may change in future releases. To create your own scripts you should follow these rules:

- Place the file in **/etc/init.d.**
- Link the file to files in appropriate run state directories using the naming convention described above.
- Have the file accept the **start** and/or **stop** options.



KEY

= USER ACTION

= ROUTINE

= SYSTEM STATE

Figure 4-2. A Look at the System Life Cycle

4.2.4.3 Going to Single-User Mode

At times in a given work week, you will need to perform some administrative functions in the single-user mode, such as backing up the hard disk onto some diskettes (see Procedure 3.3). The normal way to go to single-user mode is through the **/etc/shutdown** command. This procedure executes all the files in **/etc/rc0.d** and **/etc/shutdown.d** directories by calling the **/etc/rc0** procedure, accomplishing, among other things, the following:

- closing all open files and stopping all user processes
- stopping all daemons and services
- writing all system buffers out to the disk
- unmounting all file systems except root

The entries for single-user processing in the sample **/etc/inittab** are:

```
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
```

There are two major ways to start the shutdown processing.

1. You can enter the **shutdown —i1** command (recommended).
2. You can enter the **init 1** command, which forces the **init** process to scan the table. The first entry it finds is the **s1** entry, and it starts the shutdown processing.

Now the system is in the single-user environment, and you can perform the appropriate administrative tasks.

4.2.4.4 Run Level 3

Init 3 is used to enter the Remote File Sharing state (run level 3). This procedure executes **/etc/rc2** and **/etc/rc3** to run processes in all directories associated with those two states. On top of the multi-user state (state 2) processes, the processes in **/etc/rc3.d** will:

- start Remote File Sharing and connect you to the Remote File Sharing network
- advertise your resources to remote computers
- mount remote resources on your computer

If you are in a Remote File Sharing environment you may want to change your **initdefault** entry in **/etc/inittab** from **2** to **3**, so you automatically come up in Remote File Sharing mode when you reboot. Details on Remote File Sharing are discussed in Chapter 11.

4.2.4.5 Run Levels 5 and 6

To go to the firmware mode or to reboot the system due to reconfiguration, use the **shutdown** command with the —**i5** or the —**i6** option. Note that the pertinent entries in **/etc/inittab** are very similar, and in some cases the same:

```
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
RB:6:wait:echo "\nThe system is being restarted." >/dev/console 2>&1
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
```

For both system states, the **/etc/rc0** procedure is called. Then, the **/etc/uadmin** procedure with different arguments for each state is called. This procedure invokes the **uadmin**(2) system call, which directly prepares either system state.

While the computer stays as long as you want it to in state 5, it stays only briefly in state 6 to restart the entire system initialization. Both states are similar, however, since they exist when the computer is under firmware control.

4.2.4.6 Turning the System Off

The final step in the life cycle of the system is turning it off. The following entries apply to powering the system down:

```
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
RB:6:wait:echo "\nThe system is being restarted." >/dev/console 2>&1
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
```

Invoke the **/etc/shutdown —i0** command to turn the system off. The **/etc/shutdown** and **/etc/rc0** procedures are called to clean up and stop all user processes, daemons, and other services and to unmount the file systems. Finally, the **/etc/uadmin** procedure is called, which indicates that the last step—physically removing power from the system—is under firmware control (see Procedure 3.2).

4.3 FIRMWARE MONITOR INFORMATION

This section describes the *Series 32000* Firmware Monitor commands.

The *Series 32000* Firmware Monitor is a debugger which executes *Series 32000* Assembly Language programs. The monitor may also be used to conduct cyclic tests of the system memory, print or change the contents of system registers or memory locations, move data from one memory location to another, and fill contiguous locations of memory with specific data. Other commands allow the user to dump portions of system memory, download program instructions or data from a host or target system, execute the instructions of a program in a single-step manner beginning at the current PC, or simply run a program that was previously downloaded.

The monitor is also employed by the Serial Host program to initialize the *Series 32000* Computer and invoke the *bootstrap loader* program. The *bootstrap loader* in turn boots the GENIX V.3.3 operating system. Furthermore, the stand-alone diagnostic programs used to test the *Series 32000* Computer and its peripherals, are invoked and executed through the monitor.

4.3.1 Invocation

The monitor is invoked from the Menu mode of the Level 2 tests by selecting the **M** option. The monitor is active when a header is displayed on the console. The following is an example of a header from one of the *Series 32000* computers:

National VR32 Monitor (Rev *n.n*) *date*

E SVC

*

The SVC indicates a supervisor call error message; this is a normal error on all hardware resets of the *Series 32000.*

The asterisk is the monitor prompt; it indicates that the monitor is prepared to accept a command.

The monitor reads each command input line until a carriage return (< CR>) is encountered. A < CR> prompts the monitor to execute the commands entered. Line feeds < lf> (< ctrl/J>) are ignored, and the backspace key (< BS>) may be used to delete previously typed input.

All numeric quantities must be entered in hexadecimal notation, and spaces may be inserted before any number. All commands or hex numbers may be entered in upper or lower case.

The monitor's output responses may be suspended by entering a < ctrl/S>. They may be resumed by entering a < ctrl/Q>.

Pressing the < ESC> key aborts any monitor command under execution, and discards any input; it also terminates a < ctrl/S> wait request.

The monitor is aborted by resetting the system and proceeding as appropriate when the familiar prompt shown below is displayed on the console:

Menu mode - y/n ?

Sections 3.3.2 through 3.3.7 details how the monitor associated memory is organized. These sections introduce and define the monitor print commands, change commands, MMU and configuration register commands, data commands, program execution commands, test commands, and monitor error messages.

NOTE: A module table resides in PROM at address 10H to 20H.

4.3.2 Monitor Memory Organization

The monitor requires space on the interrupt stack, but otherwise uses no RAM. It is possible for a program to move the interrupt stack and then reenter the monitor.

The interrupt stack begins at the top of memory. The interrupt stack (sp0) may be changed by a supervisor mode program; it doesn't matter where the stack is in RAM when the monitor is entered.

The monitor runs in Supervisor mode and always uses the interrupt stack. A program can run in either mode with either stack. There are two reasonable stack configurations for the monitor and a program:

1. Program on sp1 and separate memory for each stack.
2. Program on sp0 (user and monitor run on the same stack).

In the first case, the program must avoid the memory used by the interrupt stack, because those locations will be overwritten when the monitor is entered.

In the second case, the program can allocate memory in any fashion; however, if the program fails because its stack isn't in RAM, the monitor will fail as well.

The monitor begins with the first configuration. To select configuration two, specify: **cps=0.**

The following system commands may be executed by the monitor:

v *a1 l*	Compute and print the 32-bit checksum of the **l** bytes beginning at address *a1.*
q	Do a nonsequential fetch step, using NS32082 MMU.
b	Run boot program, which asks for device and file to load (executes immediately).
j	Run boot program, which asks for device and file to loads (halts after loading).
k	Run boot program, which loads kernel in single-user mode.
o	Boot kernel in multi-user mode.

4.3.3 Print Commands

Print the contents of system memory and system registers with the following commands:

all	—	Print all the registers.
ppsr	—	Print the Processor Status register.
psb	—	Print the Static Base register.
pis	—	Print the Interrupt Stack pointer (SP0).
pmod	—	Print the Mod register.
pfp	—	Print the Frame pointer.
pus	—	Print the User Stack pointer (SP1).
psp	—	Print the User Stack pointer (SP1).
		NOTE: **pus** and **psp** are functionally identical.
pintbase	—	Print the Interrupt Base register.
pr*n*	—	Print General register *n*.
ppc	—	Print the program counter.
pmb*a*	—	Print one byte of memory at address *a*.
pmw*a*	—	Print one word of memory at address *a*.
pmd*a*	—	Print one double-word of memory at address *a*.
pvb*a*	—	Print one byte of memory at virtual address *a*.
pvw*a*	—	Print one word of memory at virtual address *a*.
pvd*a*	—	Print one double-word of memory at virtual address *a*.

The **p** commands are constructed as follows: the letter **p** followed by a register name, *e.g.*, **fp** or **r1** or **m** for physical memory, or **v** for virtual memory. The **m** or **v** is then followed by a **b**, **w**, or **d** for byte, word or double-word and then the address of interest.

All register names may be abbreviated to two characters, *e.g.*, **in** for intbase.

4.3.4 Change Commands

Change memory and registers with the following commands:

cpsr=*v*	—	Change the Processor Status register to value *v*.
cmod=*v*	—	Change the Mod register to value *v*.
cfp=*v*	—	Change the Frame pointer to value *v*.
cus=*v*	—	Change the User Stack pointer (SP1) to value *v*.
csp=*v*	—	Change the User Stack pointer (SP1) to value *v*.
		NOTE: **cus** and **csp** are functionally identical.
cintbase=*v*	—	Change the Interrupt Base register to value *v*.
cr*n*=*v*	—	Change General register *n* to value *v*.
cpc=*v*	—	Change the program counter to value *v*.
cmb*a*=*v*	—	Change one byte of memory at address *a* to value *v*.
cmw*a*=*v*	—	Change one word of memory at address *a* to value *v*.
cmd*a*=*v*	—	Change one double-word of memory at address *a* to value *v*.
cvb*a*=*v*	—	Change one byte of memory at virtual address *a* to value *v*.
cvw*a*=*v*	—	Change one word of memory at virtual address *a* to value *v*.
cvd*a*=*v*	—	Change one double-word of memory at virtual address *a* to value *v*.

The **c** commands are constructed in the same manner as the **p** commands.

All register names may be abbreviated to two characters; *e.g.*, **mo** for mod.

There is no Change Register command for the Interrupt Stack or Static Base registers. The Interrupt Stack pointer can be changed with a supervisor mode program; the static base register can be changed by altering the module entry selected by the mod register.

4.3.5 MMU and Configuration Register Commands

The MMU registers are read from, and written to, with:

w*n*=*v*	—	Write value *v* to MMU register *n*.
r*n*	—	Read (and print) MMU register *n* (the registers and the corresponding value of *n*: bpr0=0 bpr1=1 pf0=4 pf1=5 sc=8 msr=a bcnt=b ptb0=c ptb1=d eia=f)

The configuration can be changed with the x" command:

x*v*	—	Execute a setcfg instruction for configuration *v* (icu=1 fpu=2 mmu=4)

4.3.6 Data Commands

m *a1 a2 n*	—	Move *n* bytes from *a1* to *a2*.
m *a1 a2 n* [b|w|d]	—	Move *n* bytes, words, or double-words from *a1* to *a2*.
f *a1 a2 dd*	—	Fill memory from *a1* to *a2* with data byte *dd*.
f *a1 a2 v* [b|w|d]	—	Fill memory from *a1* to *a2* with byte, word, or double-word value *v*.
d *a1 n*	—	Dump (print) *n* bytes of memory starting at *a1*. The format for the output of this command is the same as the format for input to the load command. NOTE: Although the data is printed in 8-digit groups, these groups are not double-words; the order of the bytes is from the least significant to the most significant which is the reverse of the order for double-words.
l *a ddddddcc*	—	Load bytes *dd...* starting at address *a*. Bytes are represented by pairs of hex digits, up to 16 data bytes (32 hex characters) per line. The line is terminated by a checksum byte, which is the 2-digit hex representation of the 8-bit sum of the data bytes on the line. If the transmitted checksum does not match the computed checksum, the message **E CRC** is printed with the value of the computed checksum.
image	—	Start the binary image loader. The image loader reads a header, data bytes, and then a 1-byte checksum from the serial line. The header consists of a 32-bit starting address and a 32-bit length value. These parameters are sent as 8-bit bytes, least significant byte first. The length parameter specifies the number of data bytes, to be transferred as 8-bit binary bytes. The checksum is also transferred as an 8-bit byte. < ESC>, < BS>, and < ctrl/S> have no special meaning in this mode.

4.3.7 Program Execution Commands

s	—	Single-step: execute the instruction at the current pc.
g	—	Start the program (with the current pc).
q	—	Do a nonsequential fetch step, using NS32082 MMU.
v *a l*	—	Compute and print the 32-bit checksum of the *l* bytes beginning at address *a*.
!	—	Ignore the remainder of a line.

4.3.8 Test Commands

t1	—	Run Test 1. This test probes the RAM configuration at every 4K interval. The test reports the first and last address of available memory. The memory is assumed to be continuous in this test.
t2 *a n*	—	Run Test 2 starting at address *a* for *n* bytes.
t3	—	Branch back to the Menu mode of the diagnostics.

The **t2** command starts Test 2, the memory test sequence. Eight subtests are run in sequence; the sequence is restarted after the last subtest. The tests destroy the original contents of the addresses checked. The memory test itself uses no RAM; however, if a failure is found, RAM locations on the stack are used temporarily by the formatted output routines. If an error occurs and the stack is within the memory range being checked, spurious errors may be reported in addition to any actual errors. The first error reported will always be a real error.

4.3.8.1 Test Patterns

Each of the subtests cycles through seven patterns (see Table 4-2).

4.3.8.2 Test Sequence

Each subtest prints its name as it starts (see Table 4-3).

TABLE 4-2. MEMORY TEST PATTERNS IN HEXADECIMAL

NO.	PATTERN	FUNCTION
1.	00000000	All zeros
2.	f f f f f f f f	All ones
3.	aaaaaaaa	Adjacent bits different
4.	55555555	Adjacent bits different
5.	11224488	A different high in each byte
6.	eeddbb77	A different low in each byte
7.	39a7c736	Miscellaneous bits

TABLE 4-3. MEMORY TESTS

NO.	NAME	CONTEXT	TEST
1.	parity	words	Parity(address) xor pattern
2.	addr ->	double-words	Address xor pattern, low to high
3.	addr < -	double-words	Address xor pattern, high to low
4.	doubles	double-words	Pattern
5.	e. words	even words	Pattern
6.	e. bytes	even bytes	Pattern
7.	load	doubles	High transfer rate

4.3.8.3 Test Failures

When a test fails, a message is printed with the following format:

Failure(=addr=xor=test=mem)=*aaaa*=*xxxx*=tttt=*mmmm*

aaaa — address containing incorrect data.

xxxx — pattern used to provide the current test variation.

tttt — expected contents of memory.

mmmm — actual contents of memory.

4.3.8.4 Memory Test 1: Address Parity

This test computes even parity for each word address. The parity bit is then extended to form 16 bits of zeros or ones; and the exclusive-OR of this value and the current pattern is written to the word.

The words are checked on a second pass.

4.3.8.5 Memory Test 2: Increasing Addresses

For each double-word address, this test writes the exclusive-OR of the pattern and the address to the double-word. The addresses are written to in increasing order from the base address. They are checked on a second pass.

4.3.8.6 Memory Test 3: Decreasing Addresses

This test is similar to Memory Test 2, but the addresses are written in decreasing order of address, starting from the base address. The addresses are checked, in increasing order, on a second pass.

4.3.8.7 Memory Test 4: Double-Words

Test 4 writes each pattern into each double-word.

4.3.8.8 Memory Test 5: Even Words

The lower word of each pattern is written to each even word.

4.3.8.9 Memory Test 6: Even Bytes

The lower byte of each pattern is written to each even byte.

4.3.8.10 Memory Test 7: Load

A sequence of double-word string moves are generated in order to test the memory system with a constant high transfer rate. This test uses memory parity hardware to detect errors. If the target board does not have memory parity hardware, the test is not useful.

4.3.8.11 Monitor Error Messages

The following list is a tabulation of the error messages output by the *Series 32000* monitor.

NVI	—	Nonvectored Interrupt.
NMI	—	Nonmaskable Interrupt.
FPU	—	FPU Trap; FPU Exception Encountered.
ABT	—	Abort; Instruction to be executed was aborted.
ILL	—	Illegal Operation.
SVC	—	Supervisor Call instruction was executed.
DVZ	—	An attempt was made to divide an integer by zero.
FLG	—	The FLAG instruction detected a "1" in the CPU PSR F bit.
BPT	—	The Breakpoint instruction was executed.
TRC	—	The instruction just executed is being traced.
UND	—	An undefined op code was encountered by the CPU.

4.4 DIAGNOSTIC INFORMATION

This section describes the *Series 32000* stand-alone diagnostics, and the power-on diagnostics.

The stand-alone diagnostics test the *Series 32000* standard peripherals (disk, floppy, CRTs, and printer); they are invoked from the monitor.

The power-on diagnostics are invoked automatically when the system is powered up; they can also be invoked from a warm start. The information in this section details the tests and describes the various output messages.

4.4.1 Stand-Alone Diagnostics

In addition to the basic diagnostic tests, the *Series 32000* has a stand-alone diagnostic program that may be invoked and executed to test peripheral devices. This program, *dcutest*, is used to test or format the hard disk drives.

The stand-alone diagnostic program is invoked through the firmware monitor and is controlled by user-selectable commands.

4.4.1.1 Dcutest Diagnostic

Dcutest is supported by five main categories of command:

- Start and Stop commands
- Test and Unit Selection commands
- Information commands
- Sector Selection commands
- Formatting commands

Dcutest is unusual in that the testing of a disk pack occurs in the background. Once a test has been set up and initiated, or when a test is under execution, *dcutest* redisplays the command prompt (DCUTEST (running) >) indicating that another command may be entered. If *dcutest* is not executing a test, or has terminated an operation, "DCUTEST (stopped) >" is displayed. Commands entered while a test operation is in progress take effect immediately.

Commands to *dcutest* are self-terminating. A carriage return is not always necessary to complete a command. One or two character commands are accepted when correct. Commands which specify a numeric argument require a carriage return to terminate the command line input. Incomplete commands or errors may be edited using the backspace key or a < ctrl/U>.

In many cases, commands from one category may directly support commands from another. Examining the definition of each command will clarify this.

Dcutest can be booted from the master GENIX V.3 floppy and run under the firmware monitor if the disk is not operating properly, or booted from the stand-alone directory, if the formatting procedures are needed for installing an add-on disk/disk module.

To load and run *dcutest*:

1. Reset the system and enter the monitor.
2. When the banner and the command prompt (*) appear, enter a **b** (for boot) to load and initiate the stand-alone boot program.
3. If loading from the floppy disk, boot up the file system floppy from the set of binary distribution floppies and enter:

   /diskette/dcutest < CR>

   To load *dcutest* from the disk enter:

   dc(0,0)

   .

   .

   .

   $$ /stand/dcutest < CR>

4. When *dcutest* "wakes up" or is first initiated, it attempts to identify and configure itself to the disk drives connected to the *Series 32000.* It does this by reading the disk parameters (sectors/cylinders/tracks) from the disk pack's header.

   Once the auto configuration is complete, *dcutest* indicates the success of the auto configuration, identifies each disk type connected to the *Series 32000,* and displays their disk parameters.

   If a disk pack does not have a header installed on it, *dcutest* will use default disk parameters. In such a case, *dcutest* will also warn that these disk packs require headers.

   Since *dcutest* has not determined the parameters of the disk packs missing headers, one of the formatting commands should be used to install the correct headers on each disk type requiring one. The formatting command used to install a header on a disk pack is discussed in the latter part of this section.

To terminate *dcutest*, press < DEL >. The < DEL > key prompts *dcutest* to type the status of an operation, exit the program, and enter the monitor. Once in the monitor, reset the system and proceed as appropriate.

The rest of this section defines commands supported by *dcutest* according to category.

Start and Stop commands are as follows.

p	Proceed with the assigned test operation.
o	Complete one last cycle of the current test operation and then stop.
s	Type any status for the current test operation and stop.
ep	On errors, type any status, and then proceed with the current or assigned test operation.
es	On errors, type any status, then stop the current or assigned test operation.
ei	Ignore all errors, and proceed with the current or the assigned test operation.
l *n*	Execute the assigned test operation for *n* cycles and then stop. (Default is infinite.)

The **p** command begins the execution of an assigned test operation.

The **o** command specifies the completion of one final cycle of the test operation under execution. If used when no test operation is under execution, the **o** command prompts *dcutest* to execute one cycle of the assigned test operation.

The **s** command directs *dcutest* to type the status of a test operation under execution and then terminate the test operation.

The **es** command directs *dcutest* to type the status of a current or an assigned test operation if an error is detected and terminate the test operation.

The **ei** command directs *dcutest* to ignore all errors detected during the execution of a current or an assigned test operation. The error status for detected errors is ignored but the errors are still logged by *dcutest*. The **ei** command is also a default command for a test operation, unless the **ep** or **es** command is specified.

The **l** *n* command sets the maximum number of cycles that a test operation may complete. If this value is left unspecified for a particular test operation, the default value for the maximum number of test cycles is infinite.

The Test and Unit Selection commands are as follows:

tr	Set up to test read disk unit (default).
tw	Set up to test write disk unit (destructive).
tb	Set up to test read and test write disk unit (nondestructive).
u *n*	Select disk unit number *n* for testing.
we	Enable writes to currently selected disk unit.
wl	Disable writes to currently selected disk unit.

wa Display warning prompt when set up to test write a disk unit.

The Information commands are followed by a brief description of their functions. These commands are also supported by command confirmation prompts:

space Type the status of the current test operation under execution without terminating it.

dv Dump the current values of all parameters (test and format) set for *dcutest.*

di Dump the last input/output parameters block (IOPB) which logged an error.

de *n* Dump the meaning of an error status code *n* output by specifying the space bar, **s, ep,** or **es** commands.

dl Dump a list of all the currently logged errors.

d *n* Dump data in sector *n.*

db Dump the contents of the data buffer.

ds Initiate a self-test and display the disk parameters of each drive connected to the *Series 32000.*

dt Dump the disk drive table containing parametric and partitioning data on known disk drive types.

du Dump the parameters available for all disks connected to the system.

f *n* Set the frequency of the status output and type the status of an executing test operation every *n*th cycle (default is 1000).

c *n* Convert sector number *n* to cylinders/track/sector number.

h Display a complete list of the *dcutest* commands.

The **dv** command displays the unit under test, the current test operation, the sector to be tested, the sector range for testing, the number of sectors per transfer, and the current sector. It also displays what action will be taken when an error is detected (as set by the **ei, ep,** or **es** commands), the frequency **f** *n* of the status display, and the maximum number of retries when *dcutest* is directed to format a disk.

The **di** command dumps the data of the last IOPB transmitted to the *Series 32000* disk controller board that experienced an error. The data of the IOPB includes the op code, the sector count, the specific cylinder, the drive, head, and sector and the buffer address.

The **de** command decodes an error status code and translates the information into an error message that can be displayed.

The **dl** command instructs *dcutest* to dump a list of all currently logged errors. The listing takes into account all errors detected for any number of test operations run during any given period, unless one of the general housekeeping commands is used to reset the *dcutest* counters. In such a case, the listing would include only those errors logged during tests run after the housekeeping command was issued.

The **db** command performs a dump of the data buffer. The data buffer contains the data of the last sector transferred (or tested), or the sector currently selected by one of the Sector Selection commands.

The **ds** command instructs *dcutest* to run the self-test, initiated when *dcutest* is first invoked, to display the header information of all disks connected to the *Series 32000.* The header information is either read from a disk pack, or is determined by a routine associated with the auto configure sequence. The header information includes the disk type and the number of cylinders, tracks, and sectors on a disk. Disks without headers and disks that fail the auto configure sequence is not included in the information displayed by the **ds** command.

The **dt** command specifies the dumping of the disk drive table which contains the basic header information of a number of industry standard drives. The information in the disk drive table is used by *dcutest,* during an auto configuration, to determine the header parameters of a disk drive connected to the *Series 32000* that does not have a header installed on it.

The **du** command specifies the dumping of all the disk data collected from the disk drives connected to the *Series 32000* when *dcutest* was initiated and the auto configuration program was run for each disk. Any disk drive that failed the auto configure will not be included in the output listing.

The **f** *n* command sets the frequency of automatic status typeouts, and is above and beyond the use of the **ep, es,** and **ei** commands. As an example, if **f 50** is entered after the *dcutest* prompt, and a test operation is initiated, *dcutest* will type the status of the test operation each time the test operation completes 50 cycles. The default value of this command is 1000.

The **c** *n* command is used to convert a given number of sectors into its equivalent cylinders/tracks/sectors values. As an example, on the 43-Mbyte disk, 136 sectors are equal to 1 cylinder, 17 sectors are equal to 1 track, and 8 tracks, therefore, are equal to 1 cylinder. Therefore, if C 153 were entered after the *dcutest* prompt, *dcutest* would output: 1 cylinder 1 track 0 sectors as the equivalent values.

The following list shows the sector selection commands, and brief descriptions of their uses:

b *l u*	Select the range or the lower *l* and upper *u* sector bounds for a test operation.
i *n*	Select sector *n* for a test operation.
il *n*	Select the number *n* of sectors to be transferred (tested) from the current sector under test or the sector to be tested.
ii *n*	Select every *n*th sector for testing beginning with the current sector.
iy *n*	Select the retry count for I/O (read and write) to *n* before reporting or logging an error.
ir *n*	Select random sectors for testing using seed value *n* (default command).
is	Select the sequential testing of sectors beginning with the current sector.

The **b** command is used to set the range of sectors to be tested during a test operation. The **l** parameter sets the lower sector boundary, and the **u** parameter sets the upper sector boundary.

The **i** command is used to select a sector *n* for a test operation, or for use by one of the information commands. Specifically, when the **i** command is used to select sector *n,* entering the **db** command (dump data buffer) displays the data in sector *n*. The execution of any test operation will also begin at sector *n.*

The **il** command specifies the number of sectors (starting from the current sector), to transfer during each test operation initiated. If a value is not specified, the default value for all test operations is one. The input parameter may be reset to its default value by resetting the *dcutest* parameters using the **r** command. A complete description of the **r** command is presented in the latter part of this section.

The **ii** command instructs *dcutest* to test every *n*th sector, for a given test operation, starting with the current sector. As an example, if the currently selected sector was sector 10 and **ii 10** was entered after the *dcutest* prompt, an assigned test operation would begin execution at sector ten, then sector 20, and so forth as directed until terminated. A test operation under execution, when the **ii** command is entered, will begin testing every *n*th sector starting with the current sector under test.

The **ir** command instructs *dcutest* to begin the random testing of a disk. The value of *n* is used as a seed number for a random number generator. If a test operation is run without specifying a sector or specific sectors, the **ir** command is the default command used by *dcutest*. The specified test is conducted on random disk sectors. Furthermore, a default seed value is also used.

The **is** command specifies the sequential testing of disk sectors beginning with the currently logged sector.

The formatting commands are as follows:

ua	Initiate the auto configuration sequence for the currently logged disk unit.
uc c*n1* t*n2* s*n3*	Set the header parameters for the currently selected disk unit.
xs	Toggle between destructive/non-destructive formatting.
xh	Install a header on the currently selected disk unit.
x*l u*	Format the currently selected disk unit from cylinder *l* to cylinder *u*.
xr*n*	Set the maximum number of formatting retries to *n*.
xi*n*	Set the interleave formatting value.
xp*z nz*	Set the PLOZ and PLONZ formatting values for the currently selected disk unit.

The **ua** command is used to initiate the auto configuration of the currently selected disk unit. The auto configure sequence is the means by which *dcutest* configures itself to a disk drive. This process requires access to the parameters of the disk drive, which are contained in the disk drive's header. Under normal circumstances, the needed information is read off the header. However, if a disk drive does not have a header installed on it, the auto configuration sequence uses the default parameters. In the latter case the **xh** command may be used to install the header (containing the calculated parameters) on the currently selected disk unit. If the routine fails to determine the disk type, the **uc** command may be used to set the header parameters manually, and then the **xh** command could be used to install the corresponding header.

NOTE: You should first format cylinder 0 (via **x0** command) before doing a **xh** command. Also, you should initialize the defect table via the **ad** command after doing an **uc** command.

The **uc** command is used to manually set the disk parameters for the disk header of the currently selected disk unit. The **c***n1* specifies the number of cylinders, the **t***n2* specifies the number of tracks, and **s***n3* specifies the number of sectors. The **uc** command may be used after the auto configuration sequence for a disk or it may be used to change standard settings, or set the header parameters of a new disk unit. In either case, the **uc** command and parameters are entered before the **xh** (install header) command is entered.

The **xh** command is used to install a header on a previously selected (using the **u***n* command) disk unit. The header could have been determined by the auto configuration sequence, or it can be manually set using the **uc** command. If the header parameters of the currently selected disk unit were determined correctly by the auto configuration sequence (at power-up, the use of the **ua** or **ds** commands), simply entering the **xh** command would install the parameters on the currently selected disk unit. If the parameters are wrong, they must be entered manually (using the **uc** command) before the **xh** command can be used to install the header on the currently selected disk. In such a case, the **uc** command and the appropriate parameters must be entered before the **xh** command is entered. Once the **uc** command is properly specified and entered, use the **ad** command to initialize the defect table and then the **xh** command to install the header on the currently selected disk unit.

The **x** command specifies the formatting of the currently selected disk unit. The *l* and *u* parameters set the lower and upper cylinder boundaries for the formatting operation. If only *l* is given, only that cylinder is formatted. If neither *l* or *u* is given, all cylinders (except unused alternates) are formatted. Formatting does not normally destroy the data currently on the disk. The **xs** command may speed up formatting at the expense of destroying current data.

The **xr** command is used to set the maximum number of formatting retries. When the **x***l u* command is used to begin the formatting of the currently selected disk unit, in some cases, *dcutest* must retry the formatting of a cylinder. The *n* parameter specified in the **xr** command, sets the maximum number of formatting retrys that *dcutest* will conduct.

In addition to the commands already given are three general "housekeeping" commands. They are:

z	Zero the status and event counters.
r	Reset parameters to initial (default) values.
< DEL>	Type the status of *dcutest* and exit *dcutest*.

The **z** command resets or zeros the status and event counters.

The **r** command resets parameters to initial values during the operation of the diagnostic.

The **< DEL>** command types the status of an operation and then exits *dcutest*.

4.4.1.2 Dcutest for Series 32000 Defect Mapping

Dcutest for *Series 32000* can format an entire track as defective and map it to an alternate track. The **x** command for *dcutest* will format a track listed as bad in the defect table as defective and also format the corresponding alternate track. This means that all I/O requests for the defective track will automatically be done to the alternate track instead. Furthermore, all direct I/O for the alternate track will result in an error. It is an error to try to read or write a track formatted as an alternate.

Dcutest allows from 63 to 511 defective tracks to be alternated depending on disk size. It automatically allocates enough cylinders at the end of the disk (highest cylinders) as the alternate area. Once a track is alternated by the format command, the controller handles all mapping of the defective track to the alternate track. The software (*e.g.*, operating system) need not get involved.

The alternate area includes one diagnostic track used by the disk controller's diagnostic feature. The diagnostic track is the last track on head 0 and must be formatted as a normal track (not defective, not alternate).

The defect table is kept starting on sector 1 (2nd sector) of the disk and contains the bad (defective) and the corresponding alternate tracks for the disk.

The user may input the defects manually or have *dcutest* scan the disk for defects via the **aa** command.

The defect table on disk is used if it exists already.

NOTE: All manual changes to the defect table are immediately updated on disk. After a defective track is entered in the table, the user must format the track to have the defect "take effect."

The following is a list and description of the defect commands of *dcutest*.

aa	This command enables forwarding, (*i.e.*, scan entire disk for defects). Should not modify sector/transfer via **il** command while it is scanning. If aborted, should do a **r** command.
am *cyl track*	This command adds the track to the defect table and writes the table to the second sector on disk.
ad *index*	This command deletes the entry in the defect table. The *index* is that printed by the **dm** command. NOTE: The *index* changes with each **ad** so dump the defect table before doing an **ad** command.
dm	This command dumps the defect table.
df	This command dumps the defect table during forwarding (**aa** command).
xm	Write current defect table to disk on cylinder zero.
um *n*	Set number of entires in a defect table.
ub *n*	Set boot start sector number. Sector should be the one after all of defect table sectors on cylinder zero.

ut *n*

Set current active disk type to *n*, which is an index of a disk type as displayed by a **dt** command.

Chapter 5

DISK MANAGEMENT

5.1 INTRODUCTION

This chapter covers what you need to know about the disk and cartridge tape devices on your *Series 32000* Computer. The topics are:

- Disk device types and sizes
- Making devices known to the operating system
- Formatting disks
- Duplicating disks
- Verifying disk usability
- Using the bad block handling feature

This chapter does not deal with file systems or the information stored on disk devices. Those subjects are covered in Chapter 6.

5.2 DEVICE TYPES

The *Series 32000* Computer has two types of disk devices: sealed, integral hard disk units, and drives for removable diskettes. Possible configurations on the different models of the *Series 32000* Computer permit you to have two or more hard disk units.

All system software and user files are kept on the hard disk device(s). The diskette device is used primarily as a means of getting software or user files into the system where they can be used, or out of the system for storage.

5.2.1 Hard Disk Devices

The hard disk units come in various sizes; examples are:

- 40 megabyte
- 140 megabyte

The units are sealed to protect them from dust, smoke, and other contaminants in the air. Sealing has both advantages and disadvantages. It means that the *Series 32000* Computer can operate without special dust-free climate control, but it also means that disk storage packs cannot easily be swapped in and out of use. Units that are part of a delivered *Series 32000* Computer are housed within the cabinet.

5.2.2 Diskette Drives

A *Series 32000* Computer comes with one integral diskette drive. The drive accommodates 5¼ inch flexible diskettes. (You sometimes hear them called "floppy disks.") A diskette holds a little over 700 kilobytes of data. Most microcomputer software is distributed on diskettes. The standard procedure is to read the software in from a diskette and store it on the hard disk device. The reverse path is used to make backup copies of data stored on the hard disk.

It is possible to use the diskette drive as the way to get programs and data into main memory for processing. In fact, on smaller microcomputers that do not have hard disk units, this is not only possible, but is a must. This is not, however, the optimal way of working if hard disks are available. The larger storage capacity and faster access time of the hard disk make it much more desirable. In addition, using the diskette drive in this way means it is unavailable for its more normal use.

5.3 IDENTIFYING DEVICES TO THE OPERATING SYSTEM

Before a disk or tape device can be used on a computer running the GENIX V.3 operating system, it must be made known to the system. For equipment that comes with your computer, the process of identifying devices is part of configuration and is done automatically as the system is booted.

The traditional way of handling the identification is through an entry in the **/dev** directory of the **root** file system. Of course, an entry in a directory is a file (or another directory), and conceptually a disk device is treated as if it were a file. There is a difference, however, which leads to the practice of referring to devices as "special" files. In the place where a regular file would show the character count for the file, for a special file you find two decimal numbers called the major and minor numbers. Figure 5-1 shows excerpts from the output of **ls**(1) **—l** commands on a user's directory and the **/dev** directory structure.

The extracts from directory listings in Figure 5-1 show a regular file (indicated by the dash (-) in the first position of the line) with these characteristics:

- It has 1050 characters.
- The filename is **dm.ol.**
- It is owned by user **abc** who is a member of **dsg** group.
- The owner has read/write permission, group members have read permission, other users have no permissions.

There are also device files with these characteristics:

- Major and minor numbers appear in place of the character count.

  Major is the number of the device controller or driver (actually, an offset into a table of devices in the kernel); minor is the identifying number of the specific device.

- There are devices that have identical major and minor numbers, but they are designated in one entry as a block device (a **b** in the first column) and in another entry as a character device (a **c** in the first column).

```
                (a regular file)

-rw-r----- 1   abc    dsg      1050 Apr  23 08:14 dm.ol

                (integral hard disk device files)

brw------- 2   root   sys 17, 0     Apr  15 10:59 /dev/dsk/c1d0s0
brw------- 2   root   sys 17, 1     Apr  12 13:51 /dev/dsk/c1d0s1

crw------- 2   root   sys 17, 0     Apr  15 10:58 /dev/rdsk/c1d0s0
crw------- 2   root   sys 17, 1     Apr  12 13:51 /dev/rdsk/c1d0s1

                (floppy diskette device files)

brw------- 2   root   sys 17, 128   Apr  12 13:51 /dev/dsk/c0d0s0
brw------- 7   root   sys 17, 134   Apr  22 17:56 /dev/dsk/c0d0s6

crw------- 2   root   sys 17, 128   Apr  12 13:51 /dev/rdsk/c0d0s0
crw------- 7   root   sys 17, 134   Apr  12 13:51 /dev/rdsk/c0d0s6

brw------- 7   root   sys 17, 134   Apr  22 17:56 /dev/SA/diskette1
crw------- 7   root   sys 17, 134   Apr  12 13:51 /dev/rSA/diskette1

                (cartridge tape device files)

brw-r--r-- 3   root   root3,  3     May  17 12:01 /dev/SA/ctape1
crw-r--r-- 3   root   root3,  3     May  17 12:01 /dev/rSA/ctape1
```

Figure 5-1. Directory Listing Extracts: Regular and Device Files

Notice that such pairs of files have different file names or are in different directories (for example, **/dev/SA/ctape1** and **/dev/rSA/ctape1**).

- There are alias names, for example, file **diskette1** is an alias for **c0d0s6.**
- The files are owned by **root**, and no group or other user has any permission to use them. This means that only processes with the **root** ID can read from and write to the device files. (The cartridge tape device is an exception to this rule.)

5.3.1 Block and Character Devices

The identification as a block device or a character device has more to do with how the device is accessed rather than with any physical characteristics. A block device name is used when the intent is to read from or write to the device in logical, 1024-byte blocks. In the GENIX V.3 system, standard C language subroutines for handling file I/O work with blocks.

A character device name is used to read from or write to the device one character at a time. A character device is also referred to as a "raw" device. This is reflected in the device names or directory names shown in Figure 5-1, where the character device version of the diskette drive is in directory **/dev/rdsk**. The character-at-a-time method is used by some file maintenance utilities.

5.3.2 Defining a New Special File

The need to define new special device files occurs infrequently. If you add devices, the autoboot process takes care of defining the files. When the need does occur, however, there is a GENIX V.3 system command, **mknod**(1M), available to do it. The general format of the **mknod** command is:

mknod *name* **b** | **c** *major minor*

mknod *name* **p**

The options of **mknod** are:

name	Specifies the *name* of the special file.
b	Specifies a block device.
c	Specifies a character device. The or sign (|) indicates you must specify one or the other.
major	The *major* number is the slot number.
minor	The *minor* number is the physical device.
p	Specifies the special file as a first-in, first-out device. This is also known as a named pipe. (For more on pipes, see the *Programmer's Guide.*)

5.4 FORMATTING AND PARTITIONING

Formatting a disk means establishing addressable areas on the medium. Partitioning means assigning file systems or other logical units to addressable areas.

5.4.1 Formatting Disks

Before a disk can be used for the storage of information, it must be formatted. Until the medium is formatted its surfaces are simply uncharted areas treated with a substance that accepts and holds magnetic charges. Formatting imposes an addressing scheme onto these magnetic surfaces. For disks, formatting maps both sides of the disk into tracks and sectors that can be addressed by the disk controller. A portion of the disk is reserved for data having to do with the specific disk. On a hard disk, an additional use of the reserved area is to map portions of the disk that may not be usable. Formatting a previously used disk, in addition to redefining the tracks, erases any data that may be there.

You will have much more need to format diskettes than hard disks. It is good practice to format an entire box of diskettes at the time the box is first opened. By formatting diskettes on a box basis, you avoid the problem of keeping track of which ones are or are not formatted (that is, if the box is open, it means all the diskettes are formatted). Diskettes are formatted by the GENIX V.3 system **fmtflop**(1M) command or the **sysadm format**(1) subcommand.

5.4.2 Hard Disk Partitioning

Partitions on the hard disk devices on your *Series 32000* Computer are allocated in a standard arrangement. The arrangement varies according to whether you have one or two disk drives, and what size the drives are. Appendix A describes the default partitioning.

If you have a single hard disk device, it is partitioned to accommodate the **root** and **/usr** file systems and swap space.

The default partitions are fundamentally a compromise. After your system has been in operation for a few months, you may come to feel that a different arrangement would better serve the needs of your users.

5.4.3 Planning to Change Hard Disk Partitions

The basic question in reaching a decision about re-partitioning your hard disk devices is whether you would be better off having a larger number of smaller file systems, or staying with **/usr** and **/usr2** (for which the default is the entire disk). Here are some subordinate questions that bear on the basic one:

- What group-IDs are defined? Do we have the right number of groups and are users assigned to them appropriately?
- What is the nature of the processing done by the user groups? Does their work require temporary data storage? Is there a big difference between the type of processing done by one group and that done by the others?

- Have we added, or are we planning to add, system software that changes our thinking about space requirements?

Helpful information about the performance of your existing file system arrangement can be obtained with the **sadp**(1M) command. The use of **sadp** is described in Chapter 7.

If you decide that re-partitioning is needed, it can be done using **mkpt** program in stand-alone mode.

5.4.4 Changing Partitions to Increase Swap Space

If you frequently get console messages warning of insufficient memory, it may mean that the system's current configuration of main memory and swap area is insufficient to support user demands. Before adding more main memory, an alternate solution is to expand the swap area (on either single or multiple hard disk systems).

This is accomplished by the full restore procedure described in Procedure 3.9. But before you run the procedure there are three things you should do first:

1. Find out about your present partitions. (Check Appendix A for default partitioning, or if your system has been previously repartitioned so the defaults no longer apply, check your records or use the **sysadm harddisk** menu to get the information.)
2. Decide what the new partition sizes should be. (The disk is already fully allocated, so increasing the size of the swap partition means you have to reduce the size of one of the other partitions.)
3. Do a complete backup (see Procedure 5.4). (The process of changing partitions is going to erase everything that is presently there.)

You are now ready to go ahead with Procedure 3.9.

5.5 OTHER DISK OPERATIONS

Additional operations you may need to perform from time to time are duplicating disks and verifying the usability of a disk.

5.5.1 Duplicating Disks

The contents of an existing diskette are copied to another diskette by using the **dd**(1) command or the **sysadm cpdisk**(1) subcommand. If your computer has a single diskette drive, the contents of the diskette to be duplicated are first copied to a temporary file on the hard disk. When **dd** is used, either the character or the block device can be specified, but the same device (character or block) must be specified for the entire procedure. The source diskette is then replaced and the temporary file copied to the destination diskette.

While any file system can be used for the temporary file space, it is wise to use space in either **/tmp** or **/usr/tmp**. Files in those two directories are automatically deleted during the transition to the multi-user mode (run-level 2). No matter which directory is specified, a

minimum of 1422 blocks must be available (free) in that file system to use as the temporary file space. If you have more than one diskette drive, the temporary file on hard disk is not needed.

5.5.2 Verifying Usability

An option, **—v**, of the **fmtflop**(1M) command performs a verification check to see if the formatting was done without error. When a diskette contains data, however, a different procedure must be used to find out if the diskette is usable (that is, can be read from accurately).

The integrity of the storage medium of a formatted diskette can be verified without changing the data on the disk by using the **dd**(1) command. The technique is to copy the data on the disk to **/dev/null**. Either the character (**/dev/rSA/diskette1**) or the block (**/dev/SA/diskette1**) device can be specified as the source. On completion, the **dd** command reports the number of whole and partial data blocks processed (input and output). A "good" diskette provides 158 blocks (4608-byte blocks). The block size of 4608 is the number of bytes per track (9 times 512 bytes). Specifying a block size that matches the number of bytes per track is the most efficient way to do the verify procedure. By directing the output to a null file (**/dev/null**), this procedure is independent of the amount of free disk space, and it requires no file cleanup at the end of the copy.

5.6 THE BAD BLOCK HANDLING FEATURE

NOTE: This feature applies only to hard disk device(s). There is no comparable feature for diskettes or tapes.

The *Series 32000* Computer has a software feature called bad block handling. The purpose of this feature is to extend the useful life of the integral hard disk by providing mechanisms for:

- detecting and remembering blocks that are no longer usable
- reminding you that you need to "fix" some remembered bad blocks
- restoring the usability of the disk in spite of the bad blocks that exist

It should be pointed out that new bad blocks seldom occur, particularly with the sealed environment of the integral disk, as long as you take reasonable precautions against movement or vibration of the computer while the disk is still spinning. But when a new bad block does occur, the data stored in the bad block is lost and the disk may be unusable in its current state.

The bad block handling feature addresses the problem of restoring the usability of the disk. However, you must address the data loss yourself with whatever backup procedures you deem appropriate for your situation. Backup procedures are also needed to protect against operational errors and other types of hardware failures. These other problems, particularly operational errors, are the dominant cause of data loss on a system. System backups provide protection against operational errors as well as protection against lost data due to new bad blocks. A discussion of backup procedures can be found in Chapter 6.

5.6.1 When Is a Block Bad?

A block is bad when it cannot reliably store data. This is discovered only when an attempt is made to read the data and the read fails. To make life more difficult, a read fail does not always guarantee a bad block. A read fail might also mean problems in the format of the disk or a failure in the controller or the hardware.

A write fail generally signals a problem with the format of the disk or a more basic failure in the disk or disk controller hardware. While all failures are reported, the bad block handling feature does not distinguish genuine bad blocks from format problems or hardware problems. To fix problems of those types you will need to reformat the disk or get the hardware repaired. In either case, you should call your service representative. Several distinct failures occurring about the same time should also prompt you to contact your service representative to check them out.

5.6.1.1 What Makes a Block Unreliable?

A disk is an analog medium used to store digital data. The analog phenomena involve the magnetic properties of the film coating on disk surfaces. The data are recorded with a very high bit density to get millions of bits in a small space. Because of the density, the significance of small variations in the magnetic properties of the recording medium become magnified. The variations mean that a given portion of the medium may prefer to represent some bit patterns and dislike representing others. Normally, these preferences are insignificant compared to the signal level thresholds. When variations pass the point of being insignificant, the disk has a bad block. If the data pattern in a block matches the preferences in the recording medium, the bad block may escape recognition for a while. If the disk is active, however, the block will eventually be judged unreliable.

5.6.1.2 How Are Bad Blocks Fixed?

It is not really so much that a bad block is fixed, but that the system finds a way to live with it. A small portion of the disk is set aside from the normally accessible portion of the disk. This portion, call it the media-specific data area, cannot be reached by normal GENIX V.3 system commands and system calls. This reserved portion of the disk contains a description of the properties of the disk and other media-specific data.

The media-specific data portion of the disk includes a set of blocks called the surrogate image region. The mechanism for preserving the apparent accessibility of most disk blocks is to use surrogate image blocks to contain the data that were to have been stored in the bad blocks. The media-specific data also includes a mapping table that maps bad blocks on the disk to these surrogate image blocks. The disk driver software in the operating system translates disk accesses so the data are read from or written to the surrogate image block. This disk address translation is transparent to the calling software.

Most disks come with a few manufacturing defects. Bad blocks detected in the manufacturer's quality control checks are identified on a label when the unit is delivered. The bad block handling feature provides special software for remembering bad blocks that have been found and for mapping any additional ones that are found. If a surrogate block becomes bad, the software even remaps the original bad block to a new surrogate block.

5.6.1.3 A Few Blocks Cannot Be Mapped

A few special blocks, all in the media-specific data portion of the disk, cannot be mapped:

- the disk block containing the physical description of the disk
- the disk block(s) containing the mapping table

All other blocks, including surrogate image blocks, can be mapped.

5.6.2 When Are Bad Blocks Detected?

Bad blocks are detected when input/output disk operations fail for several successive attempts. This means that data being input or output are lost, but the system can restore use of the disk by mapping bad blocks to surrogate blocks that are readable.

5.6.2.1 Often Asked Questions

Why doesn't the system try to discover that a given block is bad while the system still has the data in memory?

> Besides the undesirable increase in system size and complexity, severe performance degradation would result. Also, a block can become a bad block after the copy in memory no longer exists.

Why doesn't the system periodically test the disk for bad blocks?

> Reading blocks with their current contents may not show a bad block to be bad. A thorough bit pattern test would take so long that you would never run it, even assuming a thorough test could be devised using ordinary write/read operations. The disk manufacturer already has tested the disk using extensive bit pattern tests and special hardware. All manufacturing defects have been dealt with already.

Why are disks with manufacturing defects used?

> Allowing the disks to contain a modest number of manufacturing defects greatly increases the yield, thereby considerably reducing the cost. Many systems, including this one, take advantage of this cost reduction to provide a more powerful system at lower cost.

5.6.3 Bad Block Handling: Normal Operation

The automated mechanisms are part of the normal GENIX V.3 system execution environment. The error logging daemon is running when the *Series 32000* Computer is operating in multi-user mode (run-level 2). The first phase of automated mechanisms involves detecting new bad blocks and remembering them for later mechanisms.

An understanding of the ways of referring to disk blocks helps in understanding the examples that follow.

- A block number is an integer counter.
- A physical block number is the integer counter that describes how the sectors are numbered on the disk.

- A logical block number is the counter for sectors, starting with 0 at the portion of the disk where disk partitions begin.
- A partition block number is the counter for sectors, starting with 0 at the beginning of the partition.
- A file system block number is the counter for file system blocks, starting with 0 at the beginning the first partition in the file system.

5.6.4 A Bad Block Handling Scenario

In this scenario you are the administrator of a *Series 32000* Computer with a 40-megabyte integral disk. On the disk, physical disk block 3 is a surrogate block for physical block X in the **/usr** file system.

Block X is in a text file. Block 3 has become a bad block sometime after the last time the file was read. Now the file needs to be read again. When block X is reached, the driver for the integral disk sees that block X is mapped to block 3 and attempts to read block 3. But block 3 is now bad and cannot be read. When the integral disk driver determines that block 3 is unreadable, the following messages are output to the system console.

```
WARNING: unreadable CRC hard disk error: maj/min = 17/0
        block # = 3
HARD ERROR ON DSDU WINCHESTER DRIVE, dev maj/min 17/0, logical block 3
Disk Error Daemon: successfully logged error for block 3 on disk
maj=17 min=0
```

The attempt to read the text file has failed. As the system administrator, you notice the message on the console and run **shutdown**(1M) to go to single-user mode. While shutdown is running, the following message is output to the system console.

```
Disk Error Daemon: Disk maj=17 min=0: 1 errors logged
```

The following sections discuss what was happening in the scenario.

5.6.4.1 Disk Identification

These mechanisms support both single-disk and multi-disk models of the *Series 32000* Computer. To avoid confusion, and to support all possible configurations, disks are identified by their major/minor device numbers. Messages printed out by bad block handling use the major/minor numbers rather than any other name. The utilities of this feature can be given these names as arguments when more specialized operations must be used.

5.6.4.2 Detecting New Bad Blocks

The disk driver software detects the new bad blocks for the disk in question. The disk driver determines that a block is definitely not accessible by attempting several accesses. The disk driver also repositions the disk read/write heads between some of the retries to be sure that the problem is not a head positioning error.

5.6.4.3 Reporting and Logging New Bad Blocks

When a block is determined to be inaccessible, the disk driver tells the bad block logging mechanism about the bad block. The logging mechanism reports the problem on the system console. The report for the previous example is as follows.

```
WARNING: unreadable CRC hard disk error: maj/min = 17/0

    block # = 3
```

The data in this message are in the correct form for use with the specialized options of the bad block handling commands.

After reporting the error to the logging mechanism, the disk driver also reports the error in a different, device-dependent form, as follows.

```
HARD ERROR ON DSDU WINCHESTER DRIVE, dev maj/min 17/0, logical block 3
```

The logging mechanism then attempts to remember (write) the error in the disk error log. The disk error log is in the media-specific portion of the disk. If the error gets successfully logged, a message similar to the following is output to the system console.

```
Disk Error Daemon: successfully logged error for block 3 on disk
maj=17 min=0
```

Normally, the sequence of events previously described is what happens when the automated mechanisms can handle the identification and reporting and logging of new bad blocks. Most of the cases of new bad blocks are handled automatically.

The logging mechanism is implemented using a special driver (**hdelog**-Hard Disk Error Log driver) in the operating system and a disk error daemon process (**hdelogger**(1M)) run by the **/etc/init**(1M) process. The special driver provides access to the reserved disk areas needed by this feature, as well as providing the mechanisms for reporting and queuing up reports until they get logged. This special driver can queue up as many as 18 reports. The disk error daemon gets reports from this queue and attempts to add each report to the disk error log.

The disk error daemon also has another reporting role. When the system changes its run level (for example, when you turn on the *Series 32000* Computer, shut it off, or shut down to the single-user mode), the daemon checks the error log. If the daemon finds outstanding bad block reports in a log, it outputs a message on the system console. In the previous example, that message was as follows.

Disk Error Daemon: Disk maj=17 min=0: 1 errors logged

The normal run-level for the system is the multi-user mode (run-level 2). The **hdelogger** daemon is running in run-level 2. The daemon also runs in run-levels 3 and 4. Run-levels 5 and 6 are used for returning to firmware and for rebooting the operating system, respectively. Run-level 1, s, or S is the single-user mode. The bad block handling daemon is not running in run-levels 1, 5, or 6.

5.6.5 Errors in Single-User Mode

If errors happen while the system is in the single-user mode, the errors stay queued until the system is returned to a run-level where the bad block handling daemon is running. However, if you shut your system off or reboot it without going to another run-level, error reports in the queue are lost. When errors occur while in the single-user mode, only the messages from the logging mechanism and disk driver are output to the system console.

If you get errors while in the single-user mode and you are not ready to fix them for some reason (the mechanism for fixing them takes error reports from the queue as well as from the disk error logs), you can switch to another run-level to get them logged. You will get a "successfully logged" message for each error that occurred. When all are logged, you can switch back to the single-user mode. When you do that, a reminder message from the disk error daemon will print on the console.

5.6.6 Dealing With Data Loss

Although the useful life of disk hardware has been greatly extended through the bad block handling feature, keep in mind that when a bad block has been logged any data in the block have been lost. You must be prepared to restore files or file systems that are important to you.

Chapter 6

FILE SYSTEM ADMINISTRATION

6.1 INTRODUCTION

6.1.1 How the File System is Organized

A primary function of the GENIX V.3 operating system is to support file systems. In the GENIX V.3 system, a file is a one-dimensional array of bytes with no other structure implied. Files are attached to a hierarchy of directories. A directory is merely another type of file that the user is permitted to use, but not to write; the operating system itself retains the responsibility for writing directories. The combination of directories and files make up a file system. Figure 6-1 shows the relationship between directories and files in a GENIX V.3 file system. The circles represent directories.

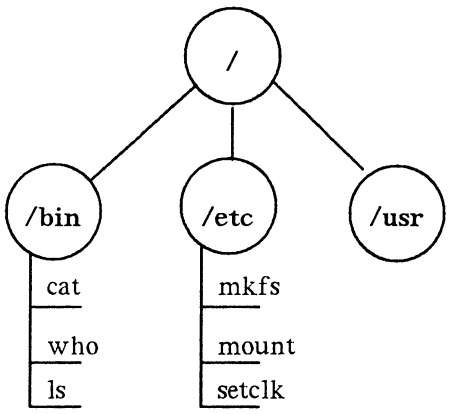


Figure 6-1. A GENIX V.3 File System

The starting point of any GENIX V.3 file system is a directory that serves as the root. In the GENIX V.3 operating system there is always one file system that is itself referred to by that name, **root**. Traditionally, the root directory of the **root** file system is represented by a single slash (/). The file system diagrammed in Figure 6-1, then, is a **root** file system. If we graft another file system onto **root** at a directory called, for example, **/usr**, the result can be illustrated by the diagram in Figure 6-2.

A directory such as **/usr** is referred to in various ways. You sometimes see the terms "leaf" and "mount point" used to describe a directory that is used to form the connection between the **root** file system and another mountable file system. Regardless of the terms used, such a directory is the root of the file system that descends from it. The name of that file system is, coincidentally,

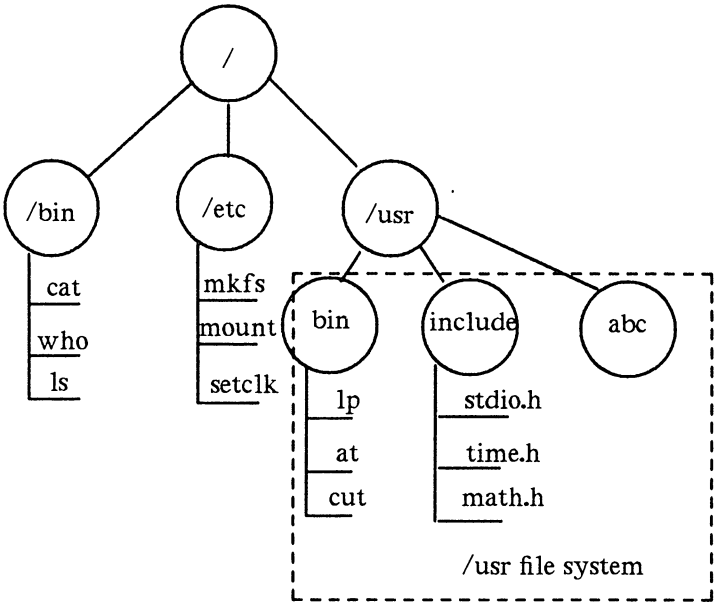


Figure 6-2. Adding the /usr File System

the name of the directory. In our example, the file system is **/usr.**

The diagrams in Figures 6-1 and 6-2 may be a convenient representation of the file and directory structure of file systems, but it is not a particularly accurate, or helpful, way of illustrating how a file system is known to the GENIX V.3 operating system. The operating system views a file system as an arrangement of addressable blocks of disk space that can be classified in four categories:

- block 0
- block 1: the super-block
- a variable number of blocks comprising the i-list
- a variable number of storage blocks: most contain data, some contain the freelist and indirect addresses

This scheme is illustrated in Figure 6-3.

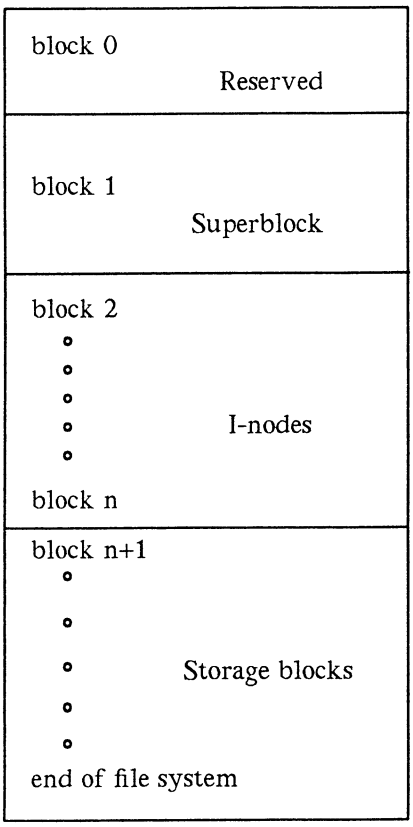


Figure 6-3. The GENIX V.3 View of a File System

6.1.2 Block 0

Block 0, although considered to be part of the file system, is actually not used by it. It is reserved for storing booting procedures. Not all file systems are involved in booting. For a file system that is not, block 0 is left unused.

6.1.3 Block 1: the Super-Block

Much of the information about the file system is stored in the super-block, including such things as

- file system size and status
  - — label, file system name
  - — size in physical and logical blocks

- read-only flag
- super-block modified flag
- date and time of last update

- i-nodes
  - total number of i-nodes allocated
  - number of i-nodes free
  - array containing 100 free i-node numbers
  - an index into the free i-node number array
- storage blocks
  - total number of free blocks
  - array containing 50 free-block numbers
  - an index into the free-block number array

A diagram of the fields in the super-block is shown in Figure 6-4. Note that the super-block does not maintain complete lists of free i-nodes and free blocks, but only enough to meet current demands as the file system is used. At almost any time, unless the file system is close to running out of i-nodes and storage blocks, there is sure to be more free i-nodes and blocks than are listed in the super-block. The information about them is kept in one of the storage blocks.

6.1.4 I-Nodes

The term "i-node" stands for information node. (You will often see it spelled with no hyphen: inode.) The same formulation is used in other references to things associated with i-nodes. For example, the list of i-nodes is referred to as the i-list (or ilist); an i-number is the position of an i-node in the i-list.

The i-node contains all the information about a file except for its name, which is kept in a directory. An i-node is 64 bytes long, so there are 8 i-nodes to a physical block. There is no set number of blocks occupied by the i-node list; it depends on how many i-nodes are specified at the time the file system is created. An i-node contains:

- the type and mode of file: type is regular (—), directory (d), block (b), character (c), or FIFO, also known as named pipe, (p); mode is the set of read-write-execute permissions
- the number of links to the file
- the owner's user-ID number
- the group-ID number to which the file belongs
- the number of bytes in the file
- an array of 13 disk block addresses
- the date and time last accessed

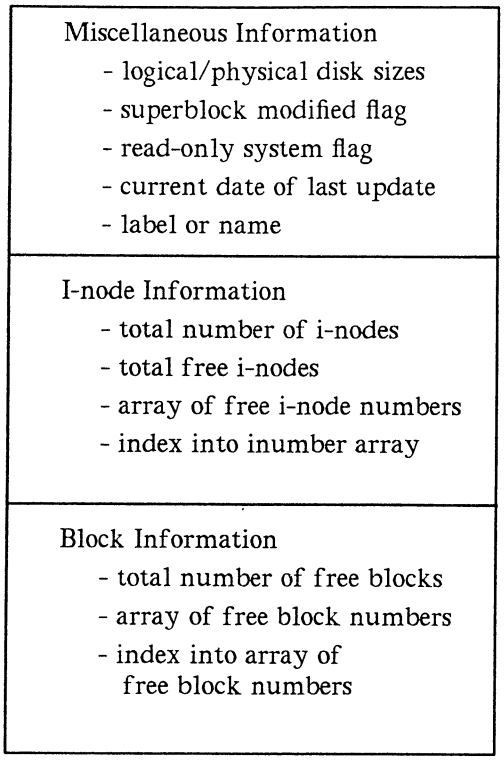


Figure 6-4. The Super-Block

- the date and time last modified
- the date and time created

The array of 13 disk block addresses is the heart of the i-node. The first 10 are direct addresses; that is, they point directly to the first 10 storage blocks of the contents of the file. If the file is larger than 10240 characters, the 11th address points to an indirect block that contains 256 more block addresses; the 12th address points to a double indirect block that contains the addresses of another 256 indirect blocks each of which contains the addresses of 256 storage blocks. Finally, and we're now up to files larger than 67,381,248 bytes, the 13th address in the array is the address of a triple indirect block that contains the addresses of 256 double indirect blocks, and so on. The theoretical maximum size of a GENIX V.3 system file is well beyond the limits of the amount of disk storage on your *Series 32000* Computer. Figure 6-5 illustrates this chaining of address blocks stemming from the i-node.

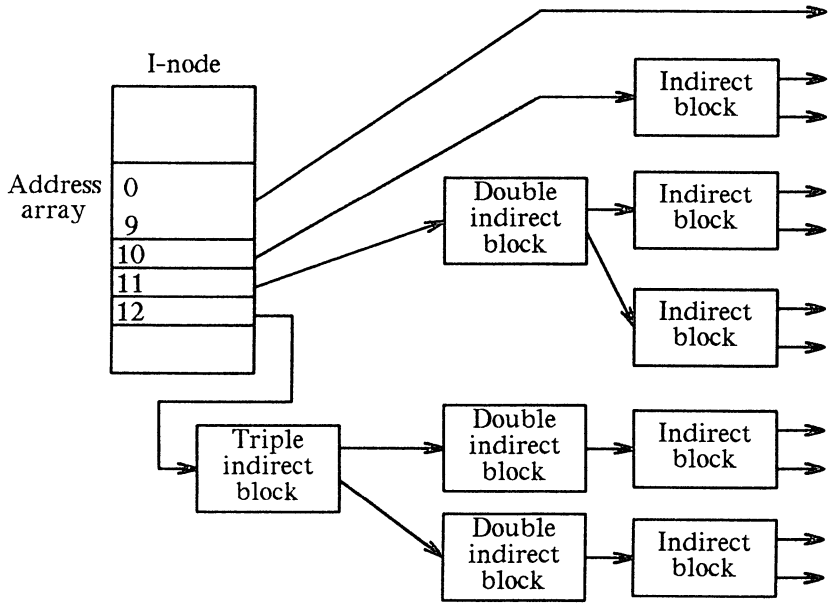


Figure 6-5. The File System Address Chain

6.1.5 Storage Blocks

The remainder of the space allocated to the file system is taken up by storage blocks, also called data blocks. For a regular file, the storage blocks contain the contents of the file. The contents are undefined. For a directory, the storage blocks contain 16-byte entries (64 to a block). Each entry represents a file or subdirectory that is a member of the directory. An entry consists of 2 bytes for the i-number and 14 bytes for the filename of the member file or subdirectory.

6.1.6 Free Blocks

Blocks not currently being used as i-nodes, as indirect address blocks, or as a storage blocks are chained together in a linked list. Each block in the list carries the address of the next block in the chain.

6.1.7 Summary

What we have described thus far is an abstract view of a GENIX V.3 file system, the components of a file system, and something of the way they relate to each other. In later sections of this chapter we will see how file systems are stored on disks, and what happens to a file system when it is in use.

6.2 THE RELATIONSHIP BETWEEN THE FILE SYSTEM AND THE STORAGE DEVICE

In the GENIX V.3 system, file systems reside on random-access disk devices. (You can put a file system on tape, but that is almost always for backup security rather than for actual access.) Before you can install a file system on a disk there are some preliminaries that must be taken care of. The material in this part of the chapter is a summary of material described in more detail in Chapter 5.

6.2.1 Disk Format

Before a disk can be used by the GENIX V.3 system it must be formatted into addressable sectors. A disk sector is a 512-byte portion of the storage medium that can be addressed by the disk controller. The number of sectors is a function of the size and number of surfaces of the disk device.

NOTE: Hard disk units on the *Series 32000* Computer are formatted when installed. The only time they might need to be reformatted would be after a catastrophic hardware failure. We recommend that you contact your service representative if such an event occurs.

Diskettes are made to be usable in more than one machine. Manufacturers produce diskettes unformatted, leaving it to customers to format them for the particular machine on which they are to be used. The commands to use are:

fmtflop(1M) to format a diskette

sysadm format(1) to format a diskette using the System Administration DISK MANAGEMENT menu

The System Administration TAPE MANAGEMENT menu has a **format** subcommand. It is used to format a cartridge tape.

6.2.2 Partitions

The next level in disk formatting is partitions. On *Series 32000* Computers, up to 16 partitions can be defined on a hard disk device, up to 8 on a floppy diskette. Partitioning of a diskette is hardcoded into the driver and may not be changed. The number of sectors allocated to a partition is specified, and a tag that is a hex code tells what the intended use of the partition is. Partition tags 0-5 are reserved. The list below shows how the tags may be used on the *Series 32000* Computer.

NAME	NUMBER
UNASSIGNED	0
BOOT	1 or 0
ROOT	2
SWAP	3
USR	4
BACKUP	5 or 0

When you first get your *Series 32000* Computer the primary hard disk device has already been formatted, and the partitions assigned are something like those shown in Table 6-1. (The example shown is for a 30-megabyte disk device.)

The illustration shows two file systems defined on this drive: the **root** file system and one called **/usr**. Space has also been set aside for the boot file and for swap space. Tables showing the default partitioning for all supported devices are in Appendix A.

TABLE 6-1. DISK PARTITIONS, 30-MEGABYTE DRIVE

CONFIGURATION	PARTITION	USE	SECTOR START	SIZE	I-NODES
QUANTUM Q540 40 Megabyte Hard Disk (Gap = 9) (Blks/Cyl = 136)	c0d0s0	root	136	12648	1568
	c0d0s1	swap	12784	10200	---
	c0d0s2	usr	22984	45560	5680
	c0d0s7	entire disk	136	68408	---
MAXTOR X7-1140 140 Megabyte Hard Disk (Gap = 9) (Blks/Cyl = 240)	c0d0s0	root	240	15120	1888
	c0d0s1	swap	15360	10320	---
	c0d0s2	usr	25680	192240	24016
	cod057	entire disk	240	217680	---

6.2.3 Size Limitations

The maximum number of blocks that can be allocated to a file system is close to the total number of sectors on the disk device. This maximum may be reduced by space set aside for swapping or paging.

The maximum number of i-nodes that can be specified is 65,500.

The size of a block on a disk is 512-bytes, the same as a disk sector. On the *Series 32000* Computer the size of a logical block is 1024 bytes. Subroutines that handle file I/O work with logical blocks of 1K rather than with the 512-byte physical blocks.

6.3 HOW THE FILE SYSTEM WORKS

What we have discussed so far has been the organization of a GENIX V.3 file system on paper and on the physical storage disk. We now want to describe what the GENIX V.3 system does with a file system when it is being used.

6.3.1 Tables in Memory

When a file system is identified to the GENIX V.3 system through a **mount**(1M) command, an entry is made in the mount table, and the super-block is read into an internal buffer maintained by the kernel. Parts of the super-block that are most needed in memory are the lists of free i-nodes and free storage blocks, and the flags and time fields that are constantly being modified.

6.3.1.1 The System I-Node Table

The GENIX V.3 system maintains a structure known as the system i-node table. Whenever a file is opened its i-node is copied from the secondary storage disk into the system i-node table. If two or more processes have the same file open, they share the same i-node table entry. The entry includes, among other things:

- the name of the device from which the i-node was copied
- the i-node number or i-number
- a reference count of the number of pointers to this entry
  (A file can be open for more than one process.)

A diagram of the system i-node table is shown in Figure 6-6.

6.3.1.2 The System File Table

The system maintains another table called the system file table. Because files may be shared among related processes a table is needed to keep track of which files are accessible by which process. For each file descriptor, an entry in the system file table contains:

- a flag to tell how the file was opened (read/write)
- a count of the processes pointing to this entry
  (When the count drops to zero, the system drops the entry.)
- a pointer to the system i-node table
- a pointer that tells where in the file the next I/O operation will take place

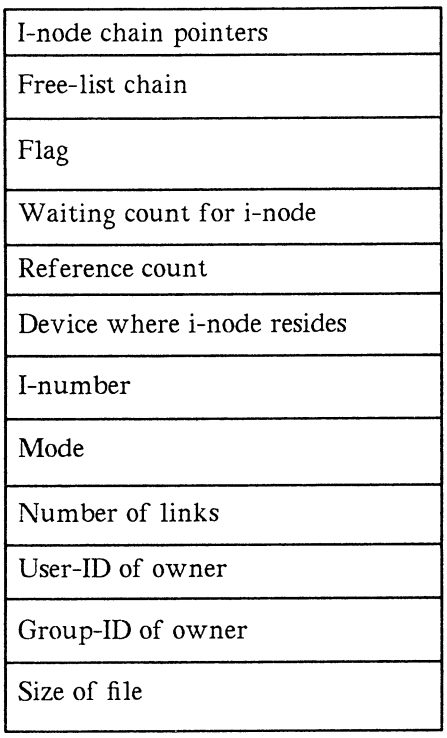


Figure 6-6. The System I-Node Table

6.3.1.3 The Open File Table

The last table that is used to provide access to files is the open file table. It is located in the user area portion of memory. There is a user area for each process and, consequently, an open file table for each process. An entry in the open file table contains a pointer to the appropriate system file table entry. Figure 6-7 shows how these tables point to each other.

6.3.2 System Steps in Accessing a File

In the next few paragraphs we describe steps followed by the operating system in opening, creating, reading, or writing a file.

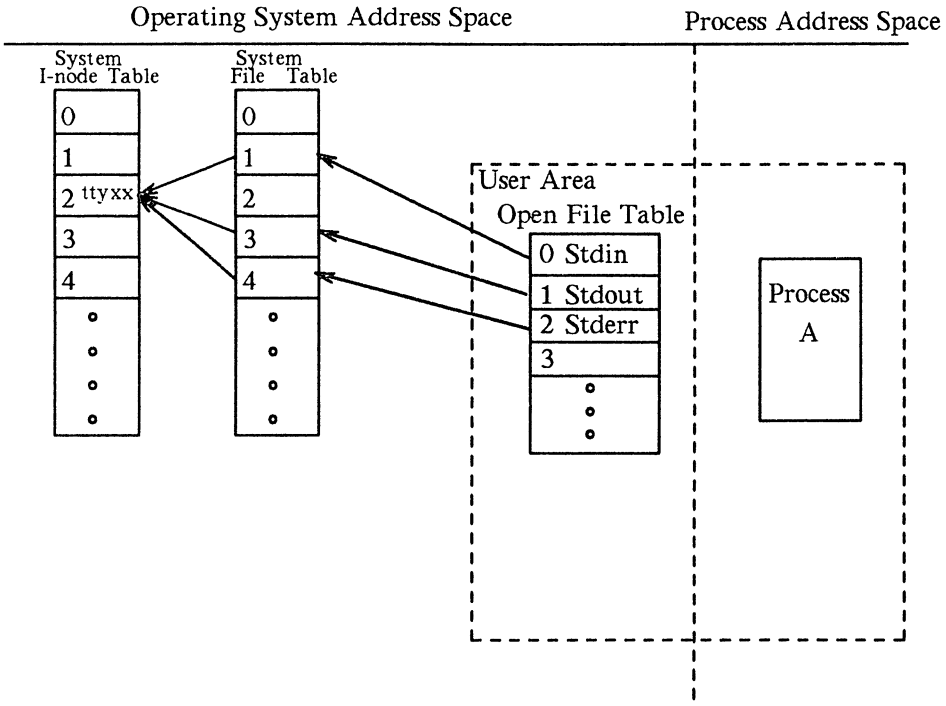


Figure 6-7. File System Tables and Their Pointers

6.3.2.1 Open

Suppose we give the pathname **/a/b** to the **open**(2) system call. (Our program probably uses the **fopen**(3) subroutine from the standard I/O library, but that in turn invokes the system call.)

1. The operating system sees that the pathname starts with a slash, so the root i-node is obtained from the i-node table.
2. Using the root i-node, the system does a linear scan of the root directory file looking for an entry **a**. When **a** is found, the operating system picks up the i-number associated with **a**.
3. The i-number gives the offset into the i-node list at which the i-node for **a** is located. At that location, the system determines that **a** is a directory.
4. Directory **a** is searched linearly until an entry **b** is found.
5. When **b** is found, its i-number is picked up and used as an index into the i-list to find the i-node for **b**.
6. The i-node for **b** is determined to be a file and is copied to the system i-node table (assuming it's not already there), and the reference count is initialized.

7. The system file table entry is allocated, the pointer to the system i-node table is set, the offset for the I/O pointer is set to zero to indicate the beginning of the file, and the reference count is initialized.

8. The user area file descriptor table entry is allocated with a pointer set to the entry in the system file table.

9. The number of the file descriptor slot is returned to the program.

The linear scan algorithm for locating the i-node of a file illustrates why it is advisable to keep directories small. Search time is also speeded up by keeping subdirectory names near the beginning of a directory file. (Use **dcopy**(1M) to do this.)

6.3.2.2 Create

Creating a file (the **creat**(2) system call) has these additional steps at the beginning:

1. The super-block is referenced for a free i-node number.

2. The mode of the file is established (possibly **and**-ed with the complement of a **umask** entry; see **umask**(2)) and entered in the i-node.

3. Using the i-number, the system goes through a directory search similar to that used in the **open** system call. The difference is that in the case of **creat** the system writes the last portion of the pathname into the directory that is the next to last portion of the pathname. The i-number is stored with it.

6.3.2.3 Reading and Writing

Both the **read**(2) and **write**(2) system call follow these steps:

1. Using the file descriptor supplied with the call as an index, the user's open file table is read and the pointer to the system file table obtained.

2. The user buffer address and number of bytes to read(write) are supplied as arguments to the call. The correct offset into the file is read from the system file table entry.

3. (Reading) The i-node is found by following the pointer from the system file table entry to the system i-node table. The operating system copies the data from storage to the user's buffer.

4. (Writing) The same pointer chain is followed, but the system writes into the data blocks. If new direct or indirect blocks are needed, they are allocated from the file system's list of free blocks.

5. Before the system call returns to the user, the number of bytes read(written) is added to the offset in the system file table.

6. The number of bytes read or written is returned to the user.

6.3.2.4 Files Used by More Than One Process

If related processes are sharing a file descriptor (as happens after a **fork**(1)) they also share the same entry in the system file table. Unrelated processes that access the same file have separate entries in the system file table, because they may be reading from or writing to different places in the file. In both cases the entry in the i-node table is shared; the correct offset at which the read or write should take place is tracked by the offset entry in the system file table.

6.3.2.5 Pathname Conversion

The directory search and pathname conversion takes place only once as long as the file remains open. For subsequent access of the file the system supplies a file descriptor that is an index into the open file table in your user process area. The open file table points to the system file table entry where the pointer to the system i-node table is picked up. Given the i-node, the system can find the data blocks that make up the file.

6.3.3 Synchronization

The above description, while complex, may seem rather neat and orderly. The situation is complicated, however, by the fact that the GENIX V.3 system is a multi-tasking system. To give some tasks prompt attention, the system may make the decision that other tasks are less urgent. In addition, the system keeps a buffer cache and a cache of free blocks and i-nodes in memory together with the super-block to provide more responsive service to users. The stability that comes from having every byte of data in a file immediately written to the storage disk is traded for the gain of being able to provide more service to more users.

In normal processing, disk buffers are flushed periodically to the disk devices. This is a system process that is not related directly to any reads or writes of user processes. The process is called "synchronization." It includes writing out the super-blocks in addition to the disk buffers. The **sync** command can be used to cause the writing of super-blocks and updated i-nodes and the flushing of buffers. It is worth noting, however, that the return from the command simply means that the writing was scheduled, not necessarily completed. For this reason, many people enter the command twice in a row to introduce delay.

6.3.4 Search Time

There are two things that have a bearing on the amount of time the system needs to spend in looking for and reading in a file:

- the size of the directories being searched
- the size of the file itself

As described above, when the GENIX V.3 system is locating a file to be opened it searches linearly through all the directories in the pathname. Search time can be reduced in two ways:

1. Keep the number of entries in a directory low. Unless compressed with **dcopy**(1M), a directory retains it largest size. If you have a directory that has had more than 640 entries you have reached the point of indirect addresses.

2. Move subdirectory names to the start of the directory. The **dcopy**(1M) utility has an option that does this. When subdirectory names are grouped at the start of a directory the system finds a match with less searching.

Large files are slower to read because of the need to chain through indirect or double indirect addresses.

6.3.5 Holes in Files

When a file is created by a program, rather than by a person using an editor, the program may seek to locations in the file that are blocks away from where previous data in the file were stored. Imagine a program that has written to blocks 1 and 2 of a file. Those storage blocks are pointed to from the i-node. If the program seeks to block 5 and writes data there, the fifth address pointer in the i-node will hold a live address, but the third and fourth will still be zeros pointing to nothing. The file is now said to have a hole in it. There is nothing really serious about this; the blocks used by the file are the ones that have been written to. If the file system is reorganized (for example, by **dcopy**(1M)), storage blocks will be assigned to blocks 3 and 4; addresses will be added to the i-node; the storage blocks will be taken off the free list and initialized to zeros.

6.3.6 Summary

We have tried in this section to give you an understanding of how the GENIX V.3 operating system controls file systems. Seeing how things are supposed to work can give us an appreciation of the steps that must be taken to keep a file system consistent.

6.4 ADMINISTERING THE FILE SYSTEM

6.4.1 Creating a File System and Making it Available

Once a disk is formatted the next step is to define the file system. The **mkfs**(1M) command is used for this purpose. The **sysadm makefsys**(1) command can be used to define a file system on diskette or cartridge tape.

6.4.2 Using mkfs

The **mkfs** command has two formats:

mkfs special_file blocks[:i-nodes] [gap blocks/cyl]
mkfs special_file proto [gap blocks/cyl]

Notice that in neither format is the file system actually given a name; it is identified by the filename of the special device file on which it will reside. The special device file, traditionally located in the directory **/dev**, is tied to the identifying controller and unit numbers (major and minor, respectively) for the physical device.

In the first format the only other information that must be furnished on the **mkfs** command line is the number of 512-byte blocks the file system is to occupy. The second format lets you include that information in a prototype file that can also define a directory and file structure for the new file system, and it even allows for reading in the contents of files from an existing file system.

Both formats let you specify information about the interrecord gap and the blocks per cylinder. If this information is not given on the command line, default values are used. Table 6-2 shows the recommended values for use with the **mkfs** command for devices supported on the *Series 32000* Computer. The recommended values are different from the defaults used by the command (check Appendix A). In the first **mkfs** format, even though the number of blocks in the file is required, the number of i-nodes may be omitted. If the number of i-nodes is omitted, the command uses a default value of one i-node for every four logical storage blocks.

TABLE 6-2. INTERRECORD GAP AND BLOCKS/CYLINDER RECOMMENDATIONS

DEVICE	GAP SIZE	BLKS/CYL	
43M Hard Disk	17	136	(Quantum)
140M Hard Disk	16	240	(Maxtor XT-114)
Floppy Disk	1	8	

If you use the first format of **mkfs**, the file system is created with a single directory. If you use a prototype file, as noted above, it can include information that causes the command to build and initialize a directory and file structure for the file system. The format of a prototype file is described in the **mkfs**(1M) pages of the *Administrator's Reference Manual.* Note that the **sysadm makefsys** subcommand has no provision for the use of prototype files.

6.4.3 Relating the File System Device to a File System Name

A GENIX V.3 file system is generally referred to by the name of the highest level directory in its hierarchy. The file system shown in Figure 6-2 at the beginning of this chapter is called **/usr** because that's the directory it is tied to. Similarly, the root file system is called that because its first directory is "root" (represented in GENIX V.3 system parlance by a slash (/)). But we saw above that when the file system was created, the only name on the command line (other than the name of a prototype file, if you used that option) was the name of a special device file. There are a couple of ways in which the file system name and the directory name can be tied together.

The first, and most explicit, is through the **labelit**(1M) command. **Labelit** makes the connection between the device special file and the mounted name of the file system. It writes the name of the file system, that is, its highest level directory, into a field in the super-block. When **labelit** is used for removable file systems, such as those on diskette, one command line argument can be the identifying number of a volume. This number, too, is stored in a field in

the super-block, but it is common practice to write it on a self-adhesive label that is attached to the diskette or tape that holds the file system.

The connection between the device and the file system name is also made by the **mount**(1M) command. This step is mandatory if the file system is to be available to users.

6.4.4 Mounting and Unmounting File Systems

For a file system to be available to users, the GENIX V.3 system has to be told to "mount" it. The **root** file system is always mounted as part of the boot procedure. The **/usr** file system, which may be on the same disk device as **root**, is also automatically mounted as the system is being brought up to multi-user mode. The issuing of the **mount** command that brings these two file systems on line is hidden in start-up shell procedures. Regardless of whether the **mount** command is hidden or not, its execution causes the file system on a specified disk device to be listed in an internal GENIX V.3 system table (called the mount table, or **/etc/mnttab**) paired with the directory that is specified. For example, the command

 # mount /dev/dsk/c0d0s2 /usr

tells the system that **/dev/dsk/c0d0s2** contains a file system that begins at directory **/usr.**

NOTE: The **mount** command has other arguments. See the *Administrator's Reference Manual* for complete details.

If you try to change directories (**cd**(1)) to a directory in **/usr** before the **mount** command is issued, the **cd** command will fail. Until the **mount** command completes, the system does not know about any of the directories beneath **/usr.** True, there is a directory **/usr** (it must exist at the time the **mount** command is issued), but the structure of files and directories below that remain hidden from the GENIX V.3 system until the **mount.**

It is common practice for small file systems to be contained completely on one diskette. A diskette can hold as much as 1422 512-byte blocks. That's close to 3/4 of a megabyte. You can define file systems on diskette and use them either for storage or for live access. Using a diskette for live access has the following two disadvantages:

1. The access time is not as fast as the hard disk.
2. It ties up the diskette device.

It is more common for users to copy in such a file system to a directory on the hard disk. To do that, the file system must first be mounted. A user who plans to establish a file system that can be brought in from diskettes needs first to create two directories on the hard disk: one to serve as a mount point, one to be the root directory of the file system being brought in. Let's say you have created the directories. The mount point is named **/hk**, and the root is named **/myfs**. You could bring a file system from diskette to hard disk with the following command sequence:

 # mount /dev/diskette /hk —r

(*The* **—r** *means read-only.*)

 # cd /hkl

 # find . —print | cpio —pdm /myfs

(*See the* **find**(1) *and* **cpio**(1) *manual pages for an explanation of the options used.*)

The command for unmounting a file system requires only the name of the special device. After you have copied in a file system from a diskette, for example, you would issue the command

 # umount /dev/diskette

to free up the diskette drive.

Unmounting is frequently a first step before using other commands that operate on file systems. For example, **fsck**(1M), which checks and repairs a file system, and **dcopy**(1M), which copies and compresses a file system, work on unmounted file systems. Unmounting is also an important part of the process of shutting the system down.

6.4.5 Summary

Thus far we have looked at file systems in the abstract. We have seen something of the way they are created, stored on disks or diskettes, made available to the system, or removed from the system. In the next portion of this chapter we will see how you maintain the integrity of an active file system.

6.5 MAINTAINING A FILE SYSTEM

Once a file system is created and made available to users it is always necessary to monitor how it is being used by the people in the organization. We get into a somewhat fuzzy area here where the distinction between a file system and its files may result in some confusion. The administrator's view is more likely to be of the file system, while users tend to think and work in terms of files. When we begin to talk about the tasks involved in keeping file systems working smoothly for users we have to be ready to deal with users' individual files as well as with the entire system.

Once a file system has been created and made available, there are several tasks routinely done to make certain that the file systems in regular use on a *Series 32000* Computer are providing the level of service and stability they should. They can be grouped into procedures for:

- checking for file system consistency
- monitoring disk usage
- compressing and reorganizing file systems
- backing up and restoring file systems

6.5.1 The Need for Policies

As with most other aspects of administering a *Series 32000* Computer, file system administration should be based on establishing a set of policies that are appropriate for your organization. There can be no hard-and-fast rules for such things as the size of file systems, the number of users in a file system, the way in which backups are done, the extent to which users can be allowed to keep inactive files in the system, or the amount of disk space a single user is entitled to occupy. These questions can only be resolved within the context of the organization. The number of users, the type of work they are doing, the number of files needed—all are variables. The responsible administrator must determine what best meets the needs of the organization.

6.5.2 Shell Scripts for File System Administration

Once policies have been agreed on, many of the routine tasks connected with file system administration can be incorporated in shell scripts. Monitoring disk usage, for example, can be handled through shell scripts that do the monitoring for you and transmit messages to the system console when exceptions are detected. Here are a few ideas:

- Use a shell script running under **cron**(1M) control to investigate free blocks and free i-nodes and to report on file systems that fall below a given threshold.
- Use a shell script to do automatic clean-ups of files that grow.
- Use a shell script to highlight cases of excessive use of disk space.

6.5.3 Checking for File System Consistency

There is a separate section later in this chapter that describes **fsck**(1M), the file system checking utility. However, we want to include this mention of it here because file system checking is central to the whole problem of normal file system maintenance.

6.5.4 Monitoring Disk Usage

You need to monitor the level of usage of a file system for the following reasons.

- If not watched regularly, the percentage of disk space used increases until the allocated space is used up.
- When the allocated space is used up processes run very slowly, or not at all, the system spends its time putting out a message about being out of file space.
- There is a natural tendency for users to forget about files they no longer use so that the files just sit there taking up space.
- Some files grow larger as a result of perfectly normal use of the system. It is an administrative responsibility to keep them under control.
- Some directories, notably **/tmp**, accumulate files during the day. When the system is first brought up, **/tmp** needs to have enough free blocks to carry it through to **shutdown**(1M).

There are four tasks that are part of keeping disk space uncluttered:

1. monitoring percent of disk space used
2. monitoring files and directories that grow
3. identifying and removing inactive files
4. identifying large space users

The tasks mentioned here are shown in Procedure 5, File System Administration Procedures. In the procedures the tasks are done through the System Administration menus. This section supplements the procedures in Part 1 and shows some GENIX V.3 system commands that you may use.

6.5.5 Monitoring Percent of Disk Space Used

Monitoring disk space may be done at any time to see how close to capacity your system is running. Until a pattern has emerged, it is advisable to check every day. In this example, the **df**(1M) command is used.

```
$ df —t
```

The **—t** option causes the total allocated blocks and i-nodes to be displayed, as well as free blocks and i-nodes. When no file systems are named, information about all mounted file systems is displayed.

6.5.6 Monitoring Files and Directories that Grow

Almost any system that is used daily has several files and directories that grow through normal use. Some examples are:

FILE	USE
/etc/wtmp	history of system logins
/usr/adm/sulog	history of **su** commands
/usr/lib/cron/log	history of actions of **/etc/cron**
/usr/lib/help/HELPLOG	actions of **/usr/bin/help**
/usr/lib/spell/spellhist	words that **spell**(1) fails to match

The frequency with which you should check growing files depends on how active your system is and how critical the disk space problem is. A good technique for keeping them down to a reasonable size uses a combination of **tail**(1) and **mv**(1):

$ tail —50 /usr/adm/sulog > /tmp/sulog

$ mv /tmp/sulog /usr/adm/sulog

This sequence puts the last 50 lines of **/usr/adm/sulog** into a temporary file, and then it moves the temporary file to **usr/adm/sulog,** thus effectively truncating the file to the 50 most recent entries.

6.5.7 Identifying and Removing Inactive Files

Part of the job of cleaning up heavily loaded file systems involves locating and removing files that have not been used recently. The commands you might use to do this work are shown below; the policy decisions involved are:

- How long should a file remain unused before it becomes a candidate for removal?
- Should users be warned that old files are about to be purged?
- Should the files be permanently removed or archived?

The **find**(1) command locates files that have not been accessed recently. **Find** searches a directory tree beginning at a point named on the command line. It looks for filenames that match a given set of expressions, and when a match is found, performs a specified action on the file. This example barely begins to suggest the full power of **find.**

$ find /usr —type f —mtime +60 —print > /tmp/deadfiles &

Here is what the example shows:

/usr specifies the pathname where **find** is to start. Presumably, your machine is organized in such a way that inactive user files will not often be found in the **root** file system.

—type f tells **find** to look only for regular files, and to ignore special files, directories, and pipes.

—mtime +60 says you are interested only in files that have not been modified in 60 days.

—print means that when a file is found that matches the **—type** and **—mtime** expressions, you want the pathname to be printed.

> /tmp/deadfiles &
directs the output to a temporary file and indicates that the process is to run in the background. This is a sensible precaution if your experience tells you to expect a substantial amount of output.

The **sysadm fileage**(1) command can be used to produce similar information (see Procedure 5.3).

6.5.8 Identifying Large Space Users

Here again the most important questions are not what commands to use to learn who is occupying excessive amounts of disk space, but rather policy questions concerned with deciding what the limits should be. Policy questions include:

- On our system, what constitutes a reasonable amount of disk space for a user to need?
- If a user exceeds the normal amount by 25%, say, is it possible the user's job requires extraordinary amounts of disk space?
- Is our system as a whole running short of space? Do our existing limits need to be reviewed?

Two commands produce useful information in this area: **du**(1) and, once again, **find**(1).

Du produces a summary of the block counts for files or directories named in the command line. For example:

$ du /usr

displays the block count for all directories in the **/usr** file system. Optional arguments allow you to refine the output somewhat. For example, **du —s** may be run against each user's login to monitor individual users.

The **find** command can be used to locate specific files that exceed a given size limit.

$ find /usr —size +10 —print

This example produces a display of the pathnames of all files (and directories) in the **/usr** file system that are larger than 10 (512-byte) blocks. Similar information can be produced by the **sysadm filesize**(1) command (see Procedure 5.3).

6.5.9 File System Backup and Restore

The importance of establishing and following a file system backup plan is too often not appreciated until data are lost and cannot be recovered. Backing-up a file system takes time. Trying to recover lost or damaged data from paper records and best-guess-work takes even more time. The value of an effective system backup plan lies in the ability to recover lost or damaged data easily.

System Administration menus are provided to help you with complete and incremental file system backups to diskette and cartridge tape, and for restoring backup data to the hard disk (see Procedure 5.4). The capability to copy selected directories and files to diskettes or tape is provided by using the **sysadm store**(1) command or the **find**(1) command with **cpio**(1) or **ctccpio**(1M). The directories and files are read back to the hard disk by using **sysadm restore**(1) or the appropriate **cpio** command.

The backup plan that you use can include any or all of these methods. The important consideration is that you evaluate the need for system backup and form a backup plan. This plan should be reevaluated as the use of the machine changes. A System Administration facility for establishing a check of your backup schedule is available; it is described in Section 6.5.13.

6.5.10 Complete Backup

The complete backup provided by the System Administration backup facilities copies directories and files of a specified mounted file system to diskettes or cartridge tape. Complete plus incremental backups combine to form a unified backup strategy. Complete backups can be done without intervening incremental backups, but incremental backups must be preceded by at least one complete backup to provide the base.

If the backup medium is diskettes, you must have enough formatted diskettes on-hand to contain the directories and files being backed up. As a rule-of-thumb, the number of diskettes needed for a complete backup is equal to the number of data blocks used in the file system divided by 1300, rounded to the nearest whole number. For example, a file system that occupies 13294 blocks requires 10 diskettes for a complete backup.

The time required to do a complete backup of a file system is about 6 minutes per diskette. The 13294 block file system is estimated to require 10 diskettes and will thus take about one hour to backup.

If the backup medium is cartridge tape, one tape can contain a file system of 45,000 (512-byte) blocks. The time required for backups to tape is significantly less than for diskettes; a full tape can be written in about 20 minutes.

6.5.11 Incremental Backup

The incremental backup provided by the System Administration backup facilities copies files that have changed since the last backup to diskettes or cartridge tape.

NOTE: Not all changed directories and files are copied in an incremental backup. The contents of **/etc/save.d/except** specifies files and directories that ARE NOT copied. A typical **/etc/save.d/except** file is shown in Figure 6-8.

```
# Patterns of filenames to be excluded from saving by savefiles.
# These are ed(1) regular expressions.
/.news_time$
/.yesterday$
/a.out$
/core$
/dead.answer$
/dead.letter$
/ed.hup$
/nohup.out$
/tmp/
^/etc/mnttab$
^/etc/save.d/timestamp/
^/etc/utmp$
^/etc/wtmp$
^/usr/adm/
^/usr/at/
^/usr/crash/
^/usr/dict/
^/usr/games/
^/usr/learn/
^/usr/news/
^/usr/spool/
^/usr/tmp/
```

Figure 6-8. Sample /etc/save.d/except File

The incremental file system backup is fast in comparison to the time required to do a complete backup because of the obvious fact that only files that have changed since the last prior backup

(whether complete or incremental) are being collected. However, while incremental backups save time, they may tie up a sizable number of diskettes, and restoring from incremental backups can result in your giving back all the time saved.

The question of when to do a new complete backup is the critical point of a system backup plan. Some things to think about are:

- If your backup medium is diskettes, how many diskettes do you want to dedicate to incremental backups? To illustrate, if incremental backups are made daily and complete backups once a month, over 40 diskettes could be required for backing-up the 13,294 block **/usr** file system mentioned above.
- How much time do you want to devote to complete backups? Fewer diskettes are required to maintain a backup library the more frequently a complete backup is done. A complete backup, however, may take 8 or 10 times as long as any single increment.
- How much time are you willing to devote to restoring a file system from backup diskettes? All diskettes of each incremental backup must be read to restore a file system completely. (This is rather a difficult decision to weigh properly; ideally, you will never need to restore a whole file system from backup.)
- What is the approximate rate of change in the file system under scrutiny? Do changes occur throughout the file system or in a small percentage of the files? If change is frequent and widespread, it may be best to schedule complete backups more often. If change is concentrated in a few files, perhaps selective backup should be a part of the overall plan.

6.5.12 Selective Backup

Specific directories and files can be quickly saved on a single diskette by using the **find**(1) and **cpio**(1) commands. The amount of data copied to a single diskette cannot exceed 1220 blocks (512 bytes per block). Before you copy the selected directory(ies) or file(s), check the number of blocks involved.

The **find** and **ctccpio** combination may be used to backup directories and files on tape. One cartridge tape can hold over 45,000 512-byte blocks.

6.5.13 Backup Schedule Reminder

A feature of the System Administration FILE MANAGEMENT menu is a way to create reminders for yourself about your backup schedule. The backup schedule reminder feature lets you do the following two things:

1. Schedule reminder messages that say, in effect, "If the machine is shut down within this time range, send a message to the console to do a backup of this or that file system."
2. Schedule reminder checks that say, "If it gets to be this time and the machine has not been shut down, take a look to see if any reminder messages would have been sent had the machine been shut down."

This reminder mechanism does not relieve you of the job of working out a reasonable schedule of backups. It merely nudges you about the schedule you have set up. Procedure 5.4 shows the steps you follow to use the reminder feature. You can, if you prefer, set up reminder notices using **cron**(1M) and **crontab**(1).

6.5.14 Restoring a File System from Backup

You can restore directories and files from system backups for a single file, a directory structure, or all files of a system backup. Use the **sysadm restore**(1) command for all three types (see Procedure 5.4 for examples).

To restore a file system using GENIX V.3 system commands, change to the appropriate directory and execute

```
$ cpio —icdumv < /dev/rdiskette
```

to take input from the diskette drive. The diskettes used must be the same ones that were used for backup, and they must be presented in the same order (see Section 6.5.12).

6.6 WHAT CAN GO WRONG WITH A FILE SYSTEM

Most of the things that can corrupt a file system have to do with the failure of the correct pointer and count information to make it out to the storage medium. This can be caused by

- hardware failure
- program interrupts
- human error

or a combination of hardware/program failures and incorrect procedures.

6.6.1 Hardware Failure

There is no very effective way of predicting when hardware failure will occur. The best way of dealing with it is to be sure that recommended diagnostic and maintenance procedures are followed conscientiously. For the *Series 32000* Computer there is a utility that flags bad blocks on hard disk and uses a substitute area for blocks the system attempts to write to a flagged block (see Chapter 4, Disk Management).

6.6.2 Program Interrupts

It is possible that errors that cause a program to fail might result in the loss of some data. It is not easy to generalize about this because the range of possibilities is so large. Perhaps the best thing to be said is that programs should be exhaustively tested before they are put into production with valuable data.

6.6.3 Human Error

While it may be painful to admit it, probably the greatest cause of file system corruption falls under this heading. We are going to recommend here four rules that should be followed by anyone who manages file systems.

1. ALWAYS check a file system before mounting it. Nothing complicates the problem of cleaning up a corrupt file system so much as allowing it to be used when it is bad.
2. NEVER remove a file system physically without first unmounting it.
3. ALWAYS use the **sync** command before shutting the system down and before unmounting a file system.
4. NEVER physically write-protect a mounted file system, unless it is mounted "read only."

The random nature of all these mishaps simply underscores the importance of establishing and observing good backup practices. It is the most effective form of insurance against data loss.

6.7 HOW TO CHECK A FILE SYSTEM FOR CONSISTENCY

When the GENIX V.3 operating system is brought up, a consistency check of the file systems should always be done. On the *Series 32000* Computer this check is automatically done as part of the power-up process. Included as part of that process is the command **fsstat**(1M). **Fsstat** returns a code for each file system on the hard disk indicating whether the consistency checking and repair program, **fsck**(1M), should be run.

These same commands, or **sysadm checkfsys**(1), should be used to check file systems not mounted routinely as part of the power-up process. If inconsistencies are discovered, corrective action must be taken before the file systems are mounted. The remainder of this section is designed to acquaint you with the command line options of the **fsck** utility, the type of checking it does in each of its phases, and the repairs it suggests.

It should be said at the outset that file system corruption, while serious, is not all that common. Most of the time a check of the file systems finds everything all right. The reason we put so much emphasis on file system checking is that if errors are allowed to go undetected, the ultimate loss can be substantial.

6.7.1 The Fsck Utility

The file system check (**fsck**) utility is an interactive file system check and repair program. Procedure 5.3 shows the steps required to run **fsck** with the **sysadm checkfsys** command. **Fsck** uses the information carried in the file system itself to perform consistency checks. If an inconsistency is detected, a message describing the inconsistency is displayed. You may elect to have **fsck** either make the repair or not. The reason you might choose to have **fsck** ignore an inconsistency is that you judge the problem to be so severe that you want either to fix it yourself using the **fsdb(1M)** utility, or you plan to go back to an earlier version of the file system. The decision to have **fsck** ignore inconsistencies and then do nothing about them yourself is not a viable one. File system inconsistencies do not repair themselves. If they are ignored, they only get worse.

6.7.2 The Fsck Command

The **fsck** command is used to check and repair inconsistencies in a file system. With the exception of the **root** file system, a file system should be unmounted while it is being checked. The **root** file system should be checked only when the computer is in run level S and no other activity is taking place in the machine.

The following is the general format of the **fsck** command:

fsck [**—y**][**—n**][**—b**][**—s***X*][**—S***X*][**—t***file*][**—q**][**—D**][**—f**][*fsdevice*]

The options of the **fsck** command are as follows:

—y	Specifies a "yes" response for all questions. This is the normal choice when the command is being run as part of a shell procedure. It generally causes **fsck** to correct all errors.
—n	Specifies a "no" response for all questions. **Fsck** will not write the file system.
—b	Reboots the system if the file system being checked is the root (/) file system and changes have been made by **fsck**. If only minor, fixable damage is found, the file system is remounted.

—s*X* Specifies an unconditional reconstruction of the free list. Following the reconstruction of the free list, the system should be rebooted so that the in-core copy of the super-block is updated (see the **—b** option). The *X* argument specifies the number of blocks-per-cylinder and the number of blocks to skip (rotational gap). The default values are those specified when the file system was created. The format for various configurations of disk drives are as follows:

Device	—s*blocks/cylinder:gap*
140M Hard Disk	**—s240:16**
43M Hard Disk	**—s136:17**
Diskette	**—s8:1**

—S*X* Specifies a conditional reconstruction of the free list, to be done only if corruption is detected. The format of the *X* argument is the same as described above for the **—s** option.

—t*file* Specifies a scratch file for use in case the file system check requires additional memory. If this option is not specified, the process asks for a filename when more memory is needed.

—q Specifies a "quiet" file system check. Output messages from the process are suppressed.

—D Checks directories for bad blocks. This option is used to check file systems for damage after a system crash.

—f Specifies that a fast file system check be done. Only Phase 1 (check blocks and sizes) and Phase 5 (check free list) are executed for a fast check. Phase 6 (reconstruct free list) is run only if necessary.

fsdevice Names the special device file associated with a file system. If no device name is specified, **fsck** checks all file systems named in **/etc/checklist.**

6.7.3 Sample Command Use

The command line below shows **fsck** being entered to check the **root** file system. No options are specified. The system response means that no inconsistencies were detected. The command operates in phases, some of which are run only if required or in response to a command line option. As each phase is completed, a message is displayed. At the end of the program a summary message is displayed showing the number of files (i-nodes), blocks, and free blocks.

```
# fsck /dev/dsk/c1d0s0

 /dev/dsk/c1d0s0
 File System: root Volume: root

 ** Phase 1 — Check Blocks and Sizes
 ** Phase 2 — Check Pathnames
 ** Phase 3 — Check Connectivity
 ** Phase 4 — Check Reference Counts
 ** Phase 5 — Check Free List
 289 files 6522 blocks 3220 free
#
```

6.7.4 File System Components Checked by Fsck

Before getting into a discussion of the **fsck** phases and the messages that may appear in each, it is important to review the components of a GENIX V.3 file system and to describe the kinds of consistency checks that are applied to them.

6.7.4.1 Super-Block

The super-block is vulnerable because every change to the file system blocks or i-nodes modifies the super-block. If the CPU is halted, and the last command involving output to the file system is not a **sync** command, the super-block is almost certainly corrupted. The super-block can be checked for inconsistencies involving:

- file system size
- i-node list size
- free-block list
- free-block count
- free i-node count

6.7.4.2 File System Size and I-Node List Size

Total file system size must be greater than the number of blocks used by the super-block plus the blocks used by the list of i-nodes. The number of i-nodes must be less than 65,500. While there is no way to check these sizes, **fsck** can check that they are within reasonable bounds. All other checks of the file system depend on the reasonableness of these values.

6.7.4.3 Free-Block List

The free-block list starts in the super-block and continues through the free-list blocks of the file system. Each free-list block can be checked for

- list count out of range
- block numbers out of range
- blocks already allocated within the file system

A check is made to see that all the blocks in the file system were found.

The first free-block list is in the super-block. The **fsck** program checks the list count for a value of less than 0 or greater than 50. It also checks each block number to make sure it is within the range bounded by the first and last data block in the file system. Each block number is compared to a list of already allocated blocks. If the free-list block pointer is not 0, the next free-list block is read and the process is repeated.

When all the blocks have been accounted for, a check is made to see if the number of blocks in the free-block list plus the number of blocks claimed by the i-nodes equals the total number of blocks in the file system. If anything is wrong with the free-block list, **fsck** can rebuild it leaving out blocks already allocated.

6.7.4.4 Free-Block Count

The super-block contains a count of the total number of free blocks within the file system. The **fsck** program compares this count to the number of blocks it found free within the file system. If the counts do not agree, **fsck** can replace the count in the super-block by the actual free-block count.

6.7.4.5 Free I-Node Count

The super-block contains a count of the number of free i-nodes within the file system. The **fsck** program compares this count to the number of i-nodes it found free within the file system. If the counts do not agree, **fsck** can replace the count in the super-block by the actual free i-node count.

6.7.4.6 I-Nodes

The list of i-nodes is checked sequentially starting with i-node 1 (there is no i-node 0). Each i-node is checked for inconsistencies involving

- format and type
- link count
- duplicate blocks
- bad block numbers
- i-node size

6.7.4.7 Format and Type

Each i-node contains a mode word. This mode word describes the type and state of the i-node. I-nodes may be one of five types:

- regular
- directory
- block special
- character special
- fifo (named pipe)

If an i-node is not one of these types, it is illegal. I-nodes may be in one of three states: unallocated, allocated, and partially allocated. This last state means an incorrectly formatted i-node. An i-node can reach this state if, for example, bad data are written into the i-node list through a hardware failure. The only corrective action **fsck** can take is to clear the i-node.

6.7.4.8 Link Count

Each i-node contains a count of the number of directory entries linked to it. The **fsck** program verifies the link count of each i-node by examining the total directory structure, starting from the root directory, and calculating an actual link count for each i-node.

If the link count stored in the i-node and the actual link count determined by **fsck** do not agree, the reason may be

- stored count not 0, actual count 0

  No directory entry appears for the i-node.
  fsck can link the disconnected file to the lost+found directory.

- stored count not 0, actual count not 0, counts unequal

  Directory entry possibly removed without i-node update.
  fsck can replace the stored link count with the actual link count.

6.7.4.9 Duplicate Blocks

Each i-node contains a list of all the blocks claimed by the i-node. The **fsck** program compares each block number claimed by an i-node to a list of allocated blocks. If a block number claimed by an i-node is on the allocated-blocks list, it is put on a duplicate-blocks list. If the block number is not on the allocated-blocks list, it is put there. If a duplicate-blocks list develops, **fsck** makes a second pass of the i-node list to find the other i-node that claims the duplicated block. While there is not enough information available to determine which i-node is in error, most of the time the i-node with the latest modification time is correct. This condition can occur by using a file system with blocks claimed by both the free-block list and by other parts of the file system.

A large number of duplicate blocks in an i-node may be caused by an indirect block not being written to the file system. The **fsck** program prompts the user to clear both i-nodes.

6.7.4.10 Bad Block Numbers

The **fsck** program checks each block number claimed by an i-node for a value lower than that of the first data block or greater than the last block in the file system. If the block number is outside this range, the block number is bad.

If there are many bad block numbers in an i-node, it may be caused by an indirect block not being written to the file system. The **fsck** program prompts the user to clear the i-node.

NOTE: A certain amount of semantic confusion is possible here. A bad block number in a file system is not the same as a bad (that is, unreadable) block on a hard disk.

6.7.4.11 I-Node Size

Each i-node contains a 32-bit (4-byte) size field. This size shows the number of characters in the file associated with the i-node. A directory i-node within the file system has the directory bit set in the i-node mode word. The directory size must be a multiple of 16 because a directory entry contains 16 bytes (2 bytes for the i-node number and 14 bytes for the file or directory name).

If the directory size is not a multiple of 16, **fsck** warns of directory misalignment. This is only a warning because not enough information can be gathered to correct the misalignment.

For a regular file, a rough check of the consistency of the size field of an i-node can be performed by using the number of characters shown in the size field to calculate how many blocks should be associated with the i-node, and comparing that to the actual number of blocks claimed by the i-node.

6.7.4.12 The Algorithm

The **fsck** program calculates the number of blocks that should be in a file by dividing the number of characters in an i-node by 512 (the number of characters per block) and rounding up. One block is added for each indirect block associated with the i-node.

If the actual number of blocks does not match the computed number of blocks, **fsck** warns of a possible file-size error. This is only a warning. Logical blocks can be created in random order, and the GENIX V.3 system does not fill them in. A check of the file would be required to tell if the error is real or not (see Section 6.3.5).

6.7.4.13 Indirect Blocks

Indirect blocks are owned by an i-node. Therefore, inconsistencies in an indirect block directly affect the i-node that owns it. Inconsistencies that can be checked are:

- blocks already claimed by another i-node
- block numbers outside the range of the file system

The consistency checks described in Sections 6.7.4.9 and 6.7.4.10 for indirect blocks as well as for the direct blocks of an i-node.

6.7.4.14 Directory Data Blocks

Directories are distinguished from regular files by an entry in the mode field of the i-node. Data blocks associated with a directory contain the directory entries. Directory data blocks are checked for inconsistencies involving:

- directory i-node numbers pointing to unallocated i-nodes
- directory i-node numbers greater than the number of i-nodes in the file system
- incorrect directory i-node numbers for "." and ".." directories
- directories disconnected from the file system

6.7.4.15 Directory Unallocated

If a directory entry i-node number points to an unallocated i-node, **fsck** can remove that directory entry. This condition occurs if the data blocks containing the directory entries are modified and written out while the i-node is not yet written out.

6.7.4.16 Bad I-Node Number

If a directory entry i-node number is pointing beyond the end of the i-node list, **fsck** can remove that directory entry. This condition occurs if bad data are written into a directory data block.

6.7.4.17 Incorrect "." and ".." Entries

The directory i-node number entry for "." should be the first entry in the directory data block. Its value should be equal to the i-node number for the directory data block.

The directory i-node number entry for ".." should be the second entry in the directory data block. Its value should be equal to the i-node number for the parent of the directory entry (or the i-node number of the directory data block if the directory is the root directory). If the directory i-node numbers for "." and ".." are incorrect, **fsck** can replace them with the correct values.

6.7.4.18 Disconnected Directories

The **fsck** program checks the general connectivity of the file system. If directories are found not to be linked into the file system, **fsck** links the directory back into the file system in the **lost+found** directory. This condition can be caused by i-nodes being written to the file system with the corresponding directory data blocks not being written to the file system. When a file is linked into the **lost+found** directory, the owner of the file needs to be told about it.

6.7.4.19 Regular Data Blocks

Data blocks associated with a regular file hold the file's contents. **Fsck** does not attempt to check the validity of the contents of a regular file's data blocks.

6.7.5 Running fsck

The **fsck** program runs in phases. Each phase reports errors it detects. If an error is one that **fsck** can correct, the user is asked if the correction should be made. This section describes the messages that are produced by each phase. Figure 6-9 shows abbreviations used in the **fsck** error messages.

6.7.5.1 Initialization Phase

Command line syntax is checked. Before the file system check can be performed, **fsck** sets up some tables and opens some files. The **fsck** program terminates on initialization errors.

BLK	block number
DUP	duplicate block number
DIR	directory name
MTIME	time file was last modified
UNREF	unreferenced

The following single-letter abbreviations, used in the messages shown in the pages that follow, are replaced by the corresponding value when the message appears on your screen:

B	block number
F	file (or directory) name
I	i-node number
M	file mode
O	user-id of a file's owner
S	file size
T	time file was last modified
X	link count,
or	number of BAD, DUP, or MISSING blocks
or	number of files (depending on context)
Y	corrected link count number
or	number of blocks in file system (depending on context)
Z	number of free blocks

Figure 6-9. Error Message Abbreviations in fsck

6.7.5.2 General Errors

Three error messages may appear in any phase. While they seem to offer the option to continue, it is generally best to regard them as fatal, end the run, and investigate what may have caused the problem.

CAN NOT SEEK: BLK B (CONTINUE?)

The request to move to a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem (probably a hardware failure) that may require additional help.

CAN NOT READ: BLK B (CONTINUE?)

The request for reading a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem (probably a hardware failure) that may require additional help.

CAN NOT WRITE: BLK B (CONTINUE?)

The request for writing a specified block number *B* in the file system failed. The disk may be write-protected.

6.7.5.3 Meaning of Yes/No Responses

An **n**(no) response to the CONTINUE? prompt says:

Terminate program.
(This is the recommended response.)

A **y**(yes) response to the CONTINUE? prompt says:

Attempt to continue to run file system check.
Often, however, the problem persists. The error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system.

6.7.5.4 Phase 1: Check Blocks and Sizes

This phase checks the i-node list. It reports error conditions resulting from:

- checking i-node types
- setting up the zero-link-count table
- examining i-node block numbers for bad or duplicate blocks
- checking i-node size
- checking i-node format

6.7.5.5 Types of Error Messages—Phase 1

Phase 1 has three types of error messages:

1. information messages
2. messages with a CONTINUE? prompt
3. messages with a CLEAR? prompt

There is a connection between some information messages and messages with a CONTINUE? prompt. The meaning of the CONTINUE? prompt generally is that some limit of tolerance has been reached.

6.7.5.6 Meaning of Yes/No Responses—Phase 1

In Phase 1, an **n**(no) response to the CONTINUE? prompt says:

Terminate the program.

In Phase 1, a **y**(yes) response to the CONTINUE? prompt says:

Continue with the program.
This error condition means that a complete check of the file system is not possible. A second run of **fsck** should be made to recheck this file system.

In Phase 1, an **n**(no) response to the CLEAR? prompt says:

Ignore the error condition.

A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 1, a **y**(yes) response to the CLEAR? prompt says:

Deallocate i-node *I* by zeroing its contents.

This may invoke the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this i-node.

6.7.5.7 Phase 1 Error Messages

UNKNOWN FILE TYPE I=I (CLEAR?)

The mode word of the i-node *I* suggests that the i-node is not a pipe, special character i-node, regular i-node, or directory i-node.

LINK COUNT TABLE OVERFLOW (CONTINUE?)

An internal table for **fsck** containing allocated i-nodes with a link count of zero has no more room.

B BAD I=I

I-node *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 if i-node *I* has too many block numbers outside the file system range. This error condition invokes the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE BAD BLOCKS 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 the last block in the file system associated with i-node *I*.

B DUP I=I

I-node *I* contains block number *B*, which is already claimed by another i-node. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if i-node *I* has too many block numbers claimed by other i-nodes. This error condition invokes Phase 1B and the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE DUP BLKS I=I (CONTINUE?)

There is more than a tolerable number (usually 10) of blocks claimed by other i-nodes.

DUP TABLE OVERFLOW (CONTINUE?)

An internal table in **fsck** containing duplicate block numbers has no more room.

POSSIBLE FILE SIZE ERROR I=I

The i-node *I* size does not match the actual number of blocks used by the i-node. This is only a warning. If the **—q** option is used, this message is not printed.

DIRECTORY MISALIGNED I=I

The size of a directory i-node is not a multiple of 16. This is only a warning. If the **—q** option is used, this message is not printed.

PARTIALLY ALLOCATED INODE I=I (CLEAR?)

I-node *I* is neither allocated nor unallocated.

6.7.5.8 Phase 1B: Rescan for More DUPS

When a duplicate block is found in the file system, the file system is rescanned to find the i-node that previously claimed that block. When the duplicate block is found, the following information message is printed:

B DUP I=I

I-node *I* contains block number *B*, which is already claimed by another i-node. This error condition invokes the BAD/DUP error condition in Phase 2. I-nodes with overlapping blocks may be determined by examining this error condition and the DUP error condition in Phase 1.

6.7.5.9 Phase 2: Check Path Names

This phase removes directory entries pointing to bad i-nodes found in Phase 1 and Phase 1B. It reports error conditions resulting from

- root i-node mode and status
- directory i-node pointers out of range
- directory entries pointing to bad i-nodes

6.7.5.10 Types of Error Messages—Phase 2

Phase 2 has 4 types of error messages:

1. information messages
2. messages with a FIX? prompt
3. messages with a CONTINUE? prompt
4. messages with a REMOVE? prompt

6.7.5.11 Meaning of Yes/No Responses—Phase 2

In Phase 2, an **n**(no) response to the FIX? prompt says:

Terminate the program since **fsck** will be unable to continue.

In Phase 2, a **y**(yes) response to the FIX? prompt says:

Change the root i-node type to "directory."
If the root i-node data blocks are not directory blocks, a very large number of error conditions are produced.

In Phase 2, an **n**(no) response to the CONTINUE? prompt says:

Terminate the program.

In Phase 2, a **y**(yes) response to the CONTINUE? prompt says:

Ignore DUPS/BAD error condition in root i-node and attempt to continue to run the file system check.
If root i-node is not correct, then this may result in a large number of other error conditions.

In Phase 2, an **n**(no) response to the REMOVE? prompt says:

Ignore the error condition.
A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 2, a **y**(yes) response to the REMOVE? prompt says:

Remove duplicate or unallocated blocks.

6.7.5.12 Phase 2 Error Messages

ROOT INODE UNALLOCATED. TERMINATING
The root i-node (always i-node number 2) has no allocate mode bits. The occurrence of this error condition indicates a serious problem. The program stops.

ROOT INODE NOT DIRECTORY (FIX?)
The root i-node (usually i-node number 2) is not directory i-node type.

DUPS/BAD IN ROOT INODE (CONTINUE?)
Phase 1 or Phase 1B found duplicate blocks or bad blocks in the root i-node (usually i-node number 2) for the file system.

I OUT OF RANGE I=I NAME=F (REMOVE?)
A directory entry *F* has an i-node number *I* that is greater than the end of the i-node list.

UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F (REMOVE?)
A directory entry *F* has an i-node *I* without allocate mode bits. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. If the file system is not mounted and the **—n** option was not specified, the entry is removed automatically if the i-node it points to is character size 0.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE?)

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory entry *F*, directory i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T FILE=F (REMOVE?)

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file entry *F*, i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed.

BAD BLK B IN DIR I=I OWNER=O MODE=M SIZE=S MTIME=T

This message only occurs when the **—D** option is used. A bad block was found in DIR i-node *I*. Error conditions looked for in directory blocks are nonzero padded entries, inconsistent "." and ".." entries, and embedded slashes in the name field. This error message means that the user should at a later time either remove the directory i-node if the entire block looks bad or change (or remove) those directory entries that look bad.

6.7.5.13 Phase 3: Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. It reports error conditions resulting from

- unreferenced directories
- missing or full **lost+found** directories

6.7.5.14 Types of Error Messages—Phase 3

Phase 3 has 2 types of error messages:

1. information messages
2. messages with a RECONNECT? prompt

6.7.5.15 Meaning of Yes/No Responses—Phase 3

In Phase 3, an **n**(no) response to the RECONNECT? prompt says:

Ignore the error condition.

This invokes the UNREF error condition in Phase 4.

A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 3, a **y**(yes) response to the RECONNECT? prompt says:

Reconnect directory i-node *I* to the file system in directory for lost files (usually **lost+found**).

This may invoke a **lost+found** error condition if there are problems connecting directory i-node *I* to **lost+found.** This invokes CONNECTED information message if link was successful.

6.7.5.16 Phase 3 Error Messages

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)

The directory i-node *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 i-node *I* are printed. The **fsck** program forces the reconnection of a nonempty directory.

SORRY. NO lost+found DIRECTORY

There is no **lost+found** directory in the root directory of the file system; **fsck** ignores the request to link a directory in **lost+found.** This invokes the UNREF error condition in Phase 4. Possible problem with access modes of **lost+found.**

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the **lost+found** directory in the root directory of the file system; **fsck** ignores the request to link a directory in **lost+found.** This invokes the UNREF error condition in Phase 4. Clean out unnecessary entries in **lost+found** or make **lost+found** larger (see Procedure 5.2).

DIR I=I1 CONNECTED. PARENT WAS I=I2

This is an advisory message indicating a directory i-node *I1* was successfully connected to the **lost+found** directory. The parent i-node *I2* of the directory i-node *I1* is replaced by the i-node number of the **lost+found** directory.

6.7.5.17 Phase 4: Check Reference Counts

This phase checks the link count information seen in Phases 2 and 3. It reports error conditions resulting from:

- unreferenced files
- missing or full **lost+found** directory
- incorrect link counts for files, directories, or special files
- unreferenced files and directories
- bad and duplicate blocks in files and directories
- incorrect total free-i-node counts

6.7.5.18 Types of Error Messages—Phase 4

Phase 4 has 5 types of error messages:

1. information messages
2. messages with a RECONNECT? prompt
3. messages with a CLEAR? prompt
4. messages with an ADJUST? prompt
5. messages with a FIX? prompt

6.7.5.19 Meaning of Yes/No Responses—Phase 4

In Phase 4, an **n**(no) response to the RECONNECT? prompt says:

Ignore this error condition.
This invokes a CLEAR error condition later in Phase 4.

In Phase 4, a **y**(yes) response to the RECONNECT? prompt says:

Reconnect i-node *I* to file system in the directory for lost files (usually **lost+found**).
This can cause a **lost+found** error condition in this phase if there are problems connecting i-node *I* to **lost+found.**

In Phase 4, an **n**(no) response to the CLEAR? prompt says:

Ignore the error condition.
A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 4, a **y**(yes) response to the CLEAR? prompt says:

Deallocate the i-node by zeroing its contents.

In Phase 4, an **n**(no) response to the ADJUST? prompt says:

Ignore the error condition.
A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 4, a **y**(yes) response to the ADJUST? prompt says:

Replace link count of file i-node *I* with *Y*.

In Phase 4, an **n**(no) response to the FIX? prompt says:

Ignore the error condition.
A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 4, a **y**(yes) response to the FIX? prompt says:

Replace count in super-block by actual count.

6.7.5.20 Phase 4 Error Messages

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)
I-node *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 i-node *I* are printed. If the **—n** option is omitted and the file system is not mounted, empty files are cleared automatically. Nonempty files are not cleared.

SORRY. NO lost+found DIRECTORY
There is no **lost+found** directory in the root directory of the file system; **fsck** ignores the request to link a file in **lost+found.** This invokes the CLEAR error condition later in Phase 4. Possible problem with access modes of **lost+found.**

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the **lost+found** directory in the root directory of the file system; **fsck** ignores the request to link a file in **lost+found.** This invokes the CLEAR error condition later in Phase 4. Check size and contents of **lost+found.**

(CLEAR)

The i-node mentioned in the immediately previous UNREF error condition cannot be reconnected.

LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST?)

The link count for i-node *I*, which is a file, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed.

LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST?)

The link count for i-node *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory i-node *I* are printed.

LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST?)

The link count for *F* i-node *I* is *X* but should be *Y*. The filename *F*, owner *O*, mode *M*, size *S*, and modify time *T* are printed.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

I-node *I*, which is a file, 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 i-node *I* are printed. If the **—n** option is omitted and the file system is not mounted, empty files are cleared automatically. Nonempty directories are not cleared.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

I-node *I*, which is a directory, 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 i-node *I* are printed. If the **—n** option is omitted and the file system is not mounted, empty directories are cleared automatically. Nonempty directories are not cleared.

BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file i-node *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed.

BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory i-node *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed.

FREE INODE COUNT WRONG IN SUPERBLK (FIX?)

The actual count of the free i-nodes does not match the count in the super-block of the file system. If the **—q** option is specified, the count will be fixed automatically in the super-block.

6.7.5.21 Phase 5: Check Free List

This phase checks the free-block list. It reports error conditions resulting from

- bad blocks in the free-block list
- bad free-block count
- duplicate blocks in the free-block list
- unused blocks from the file system not in the free-block list
- total free-block count incorrect

6.7.5.22 Types of Error Messages—Phase 5

Phase 5 has 4 types of error messages:

1. information messages
2. messages that have a CONTINUE? prompt
3. messages that have a FIX? prompt
4. messages that have a SALVAGE? prompt

6.7.5.23 Meaning of Yes/No Responses--Phase 5

In Phase 5, an **n**(no) response to the CONTINUE? prompt says:

Terminate the program.

In Phase 5, a **y**(yes) response to the CONTINUE? prompt says:

Ignore rest of the free-block list and continue execution of **fsck.**
This error condition will always invoke BAD BLKS IN FREE LIST error condition later in Phase 5.

In Phase 5, an **n**(no) response to the FIX? prompt says:

Ignore the error condition.
A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 5, a **y**(yes) response to the FIX? prompt says:

Replace count in super-block by actual count.

In Phase 5, an **n**(no) response to the SALVAGE? prompt says:

Ignore the error condition.
A NO response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 5, a **y**(yes) response to the SALVAGE? prompt says:

Replace actual free-block list with a new free-block list.

The new free-block list will be ordered according to the gap and cylinder specs of the **—s** or **—S** option to reduce time spent waiting for the disk to rotate into position.

6.7.5.24 Phase 5 Error Messages

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE?)

The free-block list contains more than a tolerable number (usually 10) of blocks with a value less than the first data block in the file system or greater than the last block in the file system.

EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE?)

The free-block list contains more than a tolerable number (usually 10) of blocks claimed by i-nodes or earlier parts of the free-block list.

BAD FREEBLK COUNT

The count of free blocks in a free-list block is greater than 50 or less than 0. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

X BAD BLKS IN FREE LIST

X blocks in the free-block list have a block number lower than the first data block in the file system or greater than the last block in the file system. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

X DUP BLKS IN FREE LIST

X blocks claimed by i-nodes or earlier parts of the free-list block were found in the free-block list. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

X BLK(S) MISSING

X blocks unused by the file system were not found in the free-block list. This error condition will always invoke the BAD FREE LIST condition later in Phase 5.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX?)

The actual count of free blocks does not match the count in the super-block of the file system.

BAD FREE LIST (SALVAGE?)

This message is always preceded by one or more of the Phase 5 information messages. If the **—q** option is specified, the free-block list will be salvaged automatically.

6.7.5.25 Phase 6: Salvage Free List

This phase reconstructs the free-block list. It has one possible error condition that results from bad blocks-per-cylinder and gap values.

6.7.5.26 Phase 6 Error Messages

DEFAULT FREE-BLOCK LIST SPACING ASSUMED This is an advisory message indicating the blocks-to-skip (gap) is greater than the blocks-per-cylinder, the blocks-to-skip is less than 1, the blocks-per-cylinder is less than 1, or the blocks-per-cylinder is greater than 500. The values of 7 blocks-to-skip and 400 blocks-per-cylinder are used.

Because the values used are incorrect for the *Series 32000* Computer, care must be taken to specify correct values with the **-sX** option on the command line. See the **fsck**(1M) and **mkfs**(1M) manual pages for further details.

6.7.5.27 Cleanup Phase

Once a file system has been checked, a few cleanup functions are performed. The cleanup phase displays advisory messages about the file system and status of the file system.

6.7.5.28 Cleanup Phase Messages

X files Y blocks Z free
: This is an advisory message indicating that the file system checked contained *X* files using *Y* blocks leaving *Z* blocks free in the file system.

***** BOOT GENIX V.3 (NO SYNC!) *****
: This is an advisory message indicating that a mounted file system or the root file system has been modified by **fsck**. If the GENIX V.3 system is not rebooted immediately without **sync**, the work done by **fsck** may be undone by the in-core copies of tables the GENIX V.3 system keeps. If the **—b** option of the **fsck** command was specified and the file system is **root**, a reboot is automatically done.

***** FILE SYSTEM WAS MODIFIED *****
: This is an advisory message indicating that the current file system was modified by **fsck**.

Chapter 7

PERFORMANCE MANAGEMENT

7.1 INTRODUCTION

This chapter describes ways to monitor and enhance the performance of your *Series 32000* Computer system.

- General approach to performance management

  Finding and fixing performance problems

- Improving performance
  - — Tuning the kernel for minimum overhead and tuning the disk subsystem for maximum throughput
  - — Workload analysis and housekeeping techniques for reducing peak load
  - — Estimating capacity
  - — Samples of typical procedures
- Performance tools

  Description of the performance tools

- Tunable parameters

  Extensive definition of the tunable parameters

7.2 GENERAL APPROACH TO PERFORMANCE MANAGEMENT

Performance management is an activity that may need your attention when you first set up your *Series 32000* Computer. When you bring the computer up for the first time, the system is automatically set to a basic configuration that is satisfactory for most applications. This configuration, however, cannot take into account the usage patterns and the behavior of your particular applications. For this reason, the structure of the system allows you to reconfigure it to enhance the performance for your particular application over that of the standard configuration.

It is very possible that you may never have to do any special fine tuning of your system, and that your only experience with reconfiguration is when you add new memory and peripherals.

7.2.0.1 Finding Problems

The system is automatically configured to its default configuration the first time it is booted. After the system has been running a day or so, you may receive signals from a variety of sources that the system needs tuning. In particular, you may see that the response time at the console is frequently slow. Your job of performance management really begins then. To proceed, you will use the tools described later in this chapter (most notably the **sar** command) to pinpoint the problem.

7.2.0.2 Fixing Problems

At this point, you need to take some corrective action. Some of the major areas for action are:

- Modifying the tunable configuration parameters.

  This is usually referred to as tuning the kernel, since you are adjusting the essential control structures at the heart of the system (the kernel). Many of these parameters are described in detail later in this chapter.

  When you change any of these parameters you must then reconfigure the system, which is the way to recreate a new bootable version of the operating system that incorporates the new parameter definitions.

- Uninstalling optional kernel packages not needed by your applications.

  This procedure makes disk and memory space available for user programs and can benefit performance.

- Improving disk utilization.

  In addition to the allocation of system memory space defined by the tunable parameters, you have some control over how your file systems are organized on the disk and policies to cache frequently used programs in memory.

- Defining best system usage patterns.

  Finally, you can establish the model for best system usage, such as encouraging users to run large non-interactive programs at night.

These areas are discussed in more detail in the remaining sections of the chapter.

7.3 IMPROVING PERFORMANCE

7.3.1 Modifying the Tunable Configuration Parameters

The setting of the core system tunable parameters is accomplished by editing the parameter entries in the **/etc/master.d/kernel** file, or other files in **/etc/master.d**, such as **shm, msg, sem,** or **disp.** For full definitions of the individual tunable parameters, and suggestions about setting them, refer to Sections 7.3.1 through 7.6.8. (See Figure 7-5 for the recommended initial values for the tunable parameters relative to a particular memory size.) Use the **/etc/sysdef** command to see what the current values of the tunable parameters are in the present configuration of your system.

Generally, the default parameters for your configuration will result in acceptable performance. If, however, you are running an application that has special performance needs, you can use the tools described in Section 7.5 to measure system load and determine which parameters might be changed to improve performance. Two key parameters, NBUF (system buffers) and NHBUF (hash buffers), have a strong impact on the efficiency of disk utilization. They are discussed below in Section 7.3.2.

See Procedure 6.1, Reconfiguring the System, for examples of editing the tunables and reconfiguring the system. Also, look at the end of this section for a sample of a typical reconfiguration.

7.3.2 Improving Disk Utilization

Disk input/output may cause a bottleneck in system performance. There are three steps in tuning the disk subsystem for better utilization.

- Choosing the proper number of buffers.
- Setting text-bit (sticky-bit).
- Organizing the file systems to minimize disk activity.

7.3.2.1 Choosing Buffer Size

Use the number of buffers (NBUF and NHBUF) given in Figure 7-5 as a starter. These values come close to optimum for most system workloads.

The NBUF parameter controls the number of buffers in the system buffer cache used to reduce the need to access the disks. These buffers hold recently-used data on the chance that it will be needed again. NHBUF specifies the number of hashing buckets in the buffer cache. The more buffers, the greater chance that needed data can be found in the buffers without the system having to do a time-consuming disk read. The read and write cache hit ratios listed by the **sar —b** command indicate how effective the system buffers are. If the value for NBUF is left null, the system calculates the values shown in Figure 7-5. The value for NHBUF is a power of 2 that is roughly one-quarter the value of NBUF; it must be specified in **/etc/master.d/kernel.**

Increasing NBUF and NHBUF, up to a point, improves system performance. A system with two megabytes of memory can typically devote roughly 250K bytes of memory to buffers (250

buffers at approximately 1K bytes each) while a system with 4 megabytes of memory can devote 600K bytes of memory to buffers. However, if too many buffers are allocated, there may not be enough memory space for efficient operation of user processes, and the amount of swapping done by the system will increase. The swapping activity usually costs more in system efficiency than is gained by having a large amount of buffer space. If the **sar —w** command shows that your system has **swpot/s** greater than 1.0, adding buffers may not be beneficial.

After the GENIX V.3 system has run for a day or so, you will want to check for excessive swapping activity. If such activity is found, reduce the number of buffers (NBUF and NHBUF) or increase your system memory.

7.3.2.2 Setting Text-Bit (Sticky-Bit)

Setting the text-bit can reduce the disk traffic of a select group of commands. Text pages of sticky commands are kept resident in memory, even when the process terminates. Once loaded into memory, such pages will usually remain. One exception is if the pages are reclaimed by the paging daemon in a tight memory situation (that is, when the number of available pages falls below the low-water mark as defined by the tunable parameter GPGSLO). Finding sticky pages already in memory can reduce considerably the loading time for the text pages of a process.

NOTE: Sticky text is not kept in the **swap** file, as is the case in earlier, non-paging *Series 32000* Computers.

On systems that usually run a light to medium workload and that are seldom in a tight-memory situation, setting the text-bit can cause a significant improvement in performance. On systems with limited memory or a heavy workload, however, the text-bit should not be used. Follow these guidelines:

SYSTEM MEMORY	TEXT-BIT GUIDELINE
1M	Do not use the text-bit.
2M to 4M	Use the text-bit if the average amount of free memory is greater than the high-water mark (GPGSHI) plus 100.

The average amount of free memory should be determined by setting the text-bit on some test commands with the **chmod**(1) command, and by using the **—r** option of **sar**(1) to determine the average FREEMEM count over a typical interval of system activity. If the **sar** report shows that it is safe to set the text-bit on for some commands, logical candidates would be frequently used, (preferably) small commands. If the average free memory for the interval is less than the high-water mark plus 100, then performance is not likely to be significantly improved and may be hurt by setting the text-bit.

7.3.2.3 File System Organization

This section describes several actions that can be taken to reduce the overhead of file access. As file systems are used, the blocks of individual member files tend to become physically scattered around the disk(s) and I/O becomes less efficient. This scattering yields poor ordering of blocks with files and poor directory structure.

7.3.2.4 Organization of File System Free List

The *Series 32000* Computer file systems are set up to allocate free blocks in a manner that allows the files to be read or written with efficiency. A free list array is created when a file system is created with **/etc/mkfs**(1M). The free list is set up with the rotational gap specified by **mkfs** options. On the *Series 32000* Computer a rotational gap of 10 should be explicitly stated on the **mkfs** command line. (The **mkfs** default value is 7.) The difference between successive block numbers in the free list is the rotational gap. For example, a file created on a system with a rotational gap of 10 may consist of blocks 510, 520, and 530. When the file is read, I/O requests are sent to the disk drive to read blocks 510, 520, and 530. As soon as the drive finishes reading block 510 and has started to process the second request, block 520 will be moving over the read/write head just as the drive is ready to read that request. This method makes for efficient I/O operation.

However, as you start changing files (changing size or removing), the efficiency starts to decrease. When several files are being created at once, they will be contending for blocks from the free list. Some of the blocks allocated to the files will be out of sequence. As you can see, the free list also becomes scattered about the disk as blocks are allocated and freed.

7.3.2.5 Directory Organization

Directory organization also affects input/output performance. The problems show up when files are removed by users. When a file is removed from a directory, the i-node number is nulled out. This leaves an unused slot for that i-node; over time the number of empty slots may become quite large. If you have a directory with 100 files in it and you remove the first 99 files, the directory still contains the 99 empty slots, at 16 bytes per slot, preceding the active slot. In effect, unless a directory is reorganized on the disk, it will retain the largest size it has ever achieved.

7.3.2.6 Restoring Good File System Organization

There is no automatic way to solve these problems; however, you can manually rearrange the file system. Here are a variety of ways to do this. Note that in all cases the file system(s) must be unmounted.

Step 1: To reorganize the free list, run **fsck**(1M) **—s**.

Step 2: To reorganize particular directory structures, use **cpio**(1) **—pdm** to copy them to a temporary location; remove the original structure; then use **cpio —pdm** to copy them back to their original location.

Step 3: To reorganize the file system, run **dcopy**(1M) **—s**. This is probably the most comprehensive way to handle file system reorganization. **Dcopy** performs Steps 1 and 2 above, and it also provides the opportunity to change the file system and i-node list sizes.

Step 4: If you have the Cartridge Tape Utilities, run **sysadm compress**(1) or **cmpress**(1M) to reorganize the file system. This procedure copies the file system temporarily to the cartridge tape and then back to its original location.

Step 5: To prevent file system indirection (which causes inefficient directory searching), use the following command to find directories with more than 320 entries (5120 bytes).

find / —type d —size +10 —print

If you find directories of this size, consider breaking them up into smaller directories.

Step 6: If you have more than one disk, balance heavily used file systems across the disks.

7.3.3 Defining Best System Usage Patterns

After the kernel and the system activities are tuned, and the file systems organized, the next step for improving system performance is to perform some housekeeping activities and to check whether prime time load can be reduced. The person responsible for administering the system should check for:

- less important jobs interfering with more important jobs
- unnecessary activities being carried out
- scheduling of selected jobs for when the system is not so busy
- the efficiency of user-defined features, such as **.profile** and $PATH

7.3.3.1 Ps

The **ps**(1) command is used to obtain information about active processes. The command gives a "snapshot" picture of what is going on, which is useful when you are trying to identify what processes are loading the system. Things will probably change by the time the output appears; however, the entries that you should be interested in are TIME (minutes and seconds of CPU time used by processes) and STIME (time when process first started).

When you spot a "runaway" process (one that uses progressively more system resources over a period of time while you are monitoring it), you should check with the owner. It is possible that such a process should be stopped immediately via the **kill**(1) **—9** command. When you have a real runaway, it continues to eat up system resources until everything grinds to a halt.

When you spot processes that take a very long time to execute you should consider using **cron**(1M) to execute the job during off-hours.

7.3.3.2 User $PATH Variables

$PATH is searched upon each command execution. Before outputting "not found," the system must search every directory in $PATH. These searches require both processor and disk time. If there is a disk or processor bottleneck, changes here can help performance.

Some things that you should check for in user $PATH variables are:

- Path Efficiency

  $PATH is read left to right, so the most likely places to find the command should be first in the path (**/bin** and **/usr/bin**). Make sure that a directory is not searched more than once for a command.

- Convenience and Human Factors

  Users may prefer to have the current directory listed first in the path (**:/bin**).

- Path Length

  In general, $PATH should have the least number of required entries.

- Large Directory Searches

  Searches of large directories should be avoided if possible. Put any large directories at the end of $PATH.

7.4 SAMPLES OF GENERAL PROCEDURES

This section depicts typical approaches to performance management. First, it describes a general procedure for troubleshooting performance problems. Then, it shows a sample of a typical system reconfiguration.

7.4.1 Sample Procedure for Investigating Performance Problems

Locating the source of the problem can require some careful detective work. Hence, what follows is not a canned procedure, but rather a sample of a typical approach, covering basic areas where problems usually surface, and suggesting some of the actions to take that will alleviate the problem. The most common symptom that a problem exists is consistently poor response time. Refer to Figure 7-1 for an outline of the approach. Note that if you have identified a familiar problem area, see Procedure 6.1, Reconfiguring the System, to make the necessary system parameter changes.

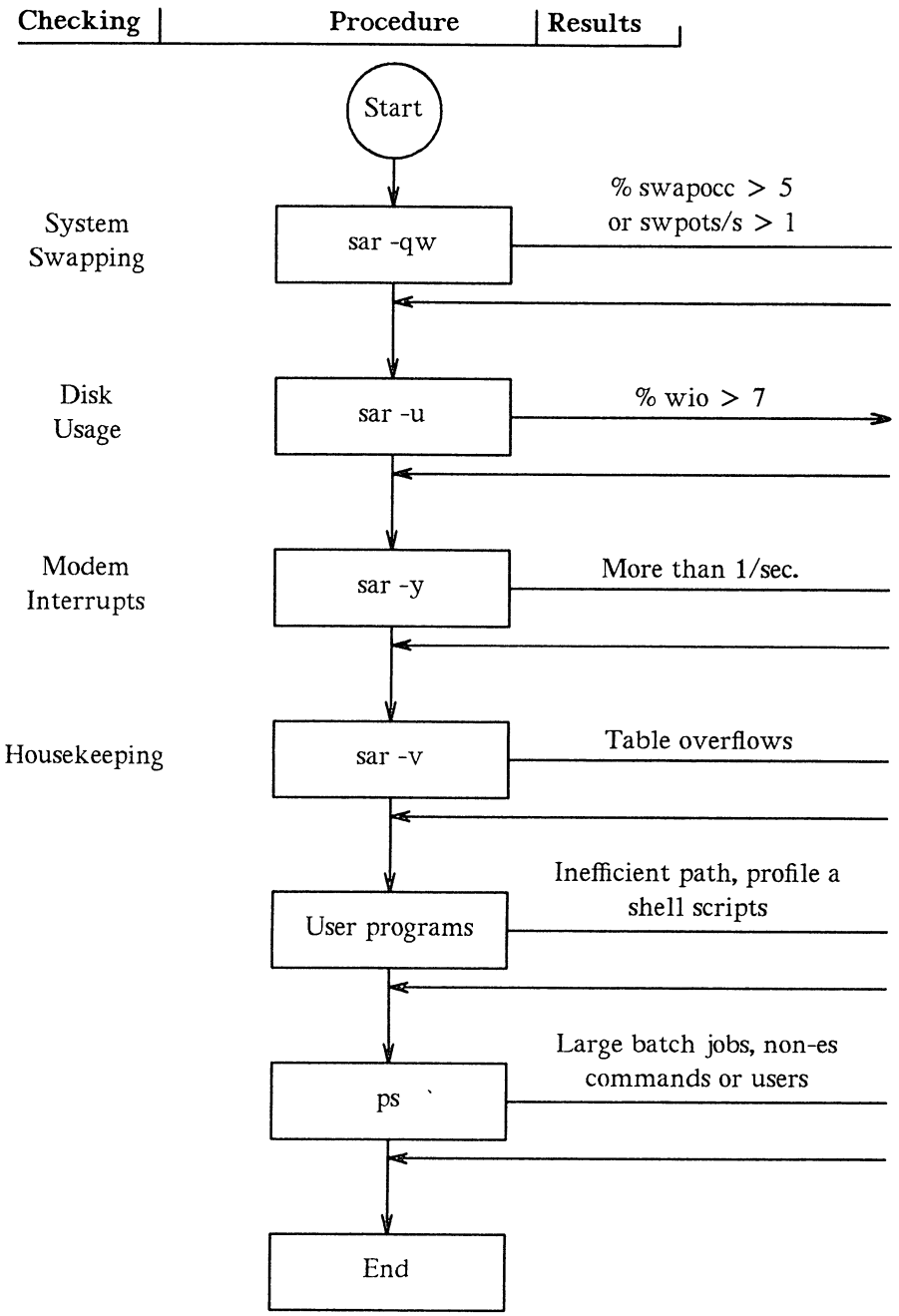


Figure 7-1. Outline of Typical Troubleshooting Procedure

7.4.1.1 Check for Excess Swapping

The first thing to look at is swapping activity, since the swapping of pages is costly in both disk and CPU overhead. Get the **sar**(1) **—qw** report. Look at the percentage that the swap queue is occupied (%swpocc) for values greater than 5. Then look at the swap-out rate (swpot/s) for values greater than 1.00.

Check whether the "freemem" count (number of pages available to user programs), shown by **sar —r**, is consistently less than the value of the tunable parameter GPGSHI (high-water mark).

If any or all of these are happening frequently, increase your system memory or check if you can reduce memory allocated for system buffers. If a large number of buffers have been configured and the buffer cache hit ratios (**sar —b**) are 90% or more, try decreasing the number of buffers (stored in the tunable parameter NBUF). It's possible that returning memory space occupied by some buffers will solve the swapping problem by leaving more space for user programs. Additional memory for user programs can also be obtained by uninstalling optional kernel utilities that are not needed by your applications.

7.4.1.2 Check for Disk Bottleneck

If the value of %wio (from the **sar —u** report above) is greater than 10%, or if the %busy for a disk drive (obtained by **sar —d**) is greater than 50%, then the system has a disk bottleneck. Some ways to alleviate a disk bottleneck are:

1. Increase the number of buffers.
2. Organize the file system to minimize disk activity. If you have two disks, distribute the file systems for a more balanced load.
3. Consider adding more memory if the situation persists. Additional memory reduces swapping/paging traffic and allows an expanded buffer pool (reducing the number of user-level reads and writes that need to go out to disk).
4. Setting the text-bit on for frequently used files may help, too. See the discussion of this subject in Section 7.3.2.2.
5. Consider adding an additional disk and balancing the most active file systems across the two disks.

7.4.1.3 Check for Potential Table Overflows

To check for potential table overflows, get the **sar —v** report. This report will let you know if overflows have occurred in the process, file, or inode tables. Overflows in these tables are avoided by increasing NPROC, NFILE, and NINODE/NS5INODE (in the **/etc/master.d/kernel** file).

7.4.1.4 Shift Workload to Off-Peak Hours

Examine **/usr/spool/crontab** to see if jobs are queued up for peak periods that might better be run at times when the system is idle. Use the **ps** command to determine what processes are heavily loading the system. Encourage users to run large, non-interactive commands (such as **nroff**(1) or **troff**(1)) at off-peak hours. You may also want to run such commands with a low priority by using the **nice**(1) or **batch**(1) commands.

7.4.2 Sample System Reconfiguration

The following is a typical scenario for reconfiguring the system because of a hardware upgrade from 1 megabyte to 2 megabytes of memory. Because of the additional memory, many tunable parameters should be increased. (See Figure 7-5 for parameter values recommended for your system.) Most of them are in the **/etc/master.d/kernel** file. The command line entries and system responses in the illustration below show the reconfiguration and rebooting of the operating system to support these new parameters. The editing of the **/etc/master.d/kernel** file is not shown.

The illustration shows that tunable parameters for semaphores are also being modified; the **/etc/master.d/sem** file is edited and **make** is run to convert SEM into a bootable object file.

```
# cd /etc/master.d
# ed kernel
    NOTE: Editing of /etc/master.d/kernel is not shown.
q
# ed sem
    NOTE: Editing of /etc/master.d/sem is not shown.
q
# cd /usr/conf/kernel
# make unix
# cp /usr/conf/kernel/unixdb332 /NEWunix
# cd
# sysadm firmware
    (A series of messages are displayed ending with the following:)
.
.
.
```

```
.
.
.
INIT: New Run level: 5

The system is down.

GENIX V System V Release 3.0 Version 2
Node unix

Total real memory    = 2087157
Available memory     = 1236992

*******************************************************

Copyright (c) 1984 AT&T - All Rights Reserved

THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF AT&T INC.
The copyright notice above does not evidence any actual or
intended publication of such source code.

*******************************************************
The system is coming up. Please wait.
Generating a new /unix

The system is ready.

Console Login:
```

7.5 PERFORMANCE TOOLS

Internal activity is measured by a number of counters contained in the GENIX V.3 system kernel. Each time an operation is performed, an associated counter is incremented. **Sar** and the other performance tools allow you to monitor the values of these counters. The functions monitored by **sar** are discussed in the following sections. The performance tools for the GENIX V.3 system internal activities described in this section are:

sar Samples cumulative activity counters internal to the GENIX V.3 system and provides reports on various system-wide activities.

sag Graphically displays the information collected by **sar**.

sadp Produces profiles of disk access location and seek distance.

timex Reports both system-wide and per-process activity during the execution of a command or program.

Examples for these tools are provided in the following sections. The examples are from a system with 2 megabytes of main memory and a 30-megabyte integral hard disk. Command outputs are typical values observed for user workloads on the GENIX V.3 operating system. Values you receive may be quite different from values in the examples, depending on your application or benchmark. When tuning your system, it is recommended that you use a benchmark or have the system under normal load for your application to allow you to tune directly toward your specific application.

7.5.1 Sar

Throughout this section, **sar** options are described with an analysis of sample outputs of the options. **Sar** can be used either to gather system activity data or to extract what has been collected in data files created by **sa1** and **sa2**. **Sa1** and **sa2** are initiated by entries put in the **crontab sys** file.

7.5.1.1 Sar —a

The **sar** —**a** option reports the use of file access operations. The GENIX V.3 operating system routines reported are as follows:

iget/s Number of files located by i-node entry per second.

namei/s Number of file system path searches per second. **Namei** calls **iget**, so **iget/s** is always larger than **namei/s**.

dirbk/s Number of directory block reads issued per second.

An example of **sar** —**a** output, with a 30-second sampling interval, follows:

```
sysnam modename   3.0 2 db332   12/30/85
12:41:40          iget/s   namei/s   dirbk/s
12:42:10               4         1         3
12:42:40               2         1         1
12:43:10               5         2         3
Average                4         1         3
```

The larger the values reported, the more time the GENIX V.3 kernel is spending to access user files. This indicates how heavily programs and applications are using the file system(s). The —**a** option is helpful for understanding how disk-dependent the application system is; it is not

used for any specific tuning step.

7.5.1.2 Sar —b

The —**b** option reports the following buffer activity.

bread/s Average number of physical blocks read into the system buffers from the disk (or other block devices) per second.

lread/s Average number of logical blocks read from system buffers per second.

%rcache Fraction of logical reads found in buffer cache (100% minus the ratio of **breads** to **lreads**).

bwrit/s Average number of physical blocks written from the system buffers to disk (or other block devices) per second.

lwrit/s Average number of logical blocks written to system buffers per second.

%wcache Fraction of logical writes found in buffer cache (100% minus the ratio of **bwrit/s** to **lwrit/s**).

pread/s Average number of physical read requests per second.

pwrit/s Average number of physical write requests per second.

The entries that you should be most interested in are the cache hit ratios **%rcache** and **%wcache** which measure the effectiveness of system buffering. If **%rcache** falls below 90, or **%wcache** falls below 65, it may be possible to improve performance by increasing the number of buffers.

An example of **sar —b** output follows:

sysnam modename 3.0 2 *db332* 12/30/85

16:32:57	bread/s	lread/s	%rcache	bwrit/s	lwrit/s	%wcache	pread/s	pwrit/s
16:33:07	3	39	93	1	16	91	0	0
16:33:17	4	40	90	2	16	87	0	0
16:33:27	4	41	90	3	7	64	0	0
Average	4	40	91	2	13	84	0	0

This example shows that the buffers are not causing any bottlenecks, because all data is within acceptable limits.

7.5.1.3 Sar —c

The —c option reports system calls in the following categories:

scall/s All types of system calls per second, generally about 30 per second on a busy 4 to 6 user system.

sread/s Read system calls per second.

swrit/s Write system calls per second.

fork/s Fork system calls per second, about 0.5 per second on a 4 to 6 user system. This number will increase if shell scripts are running.

exec/s Exec system calls per second. (If **(exec/s)** / **(fork/s)** is greater than 3, look for inefficient $PATHs.)

rchar/s Characters (bytes) transferred by read system calls per second.

wchar/s Characters (bytes) transferred by write system calls per second.

Typically, reads plus writes account for about half of the total system calls, although this varies greatly with the activities that are being performed by the system.

An example of **sar** —c output follows:

sysnam modename 3.0 2 *db332* 12/30/85

18:33:04	scall/s	sread/s	swrit/s	fork/s	exec/s	rchar/s	wchar/s
18:33:35	38	16	6	0.03	0.03	6089	1638
18:34:05	39	16	4	0.07	0.07	6123	1602
18:34:35	38	17	5	0.17	0.17	6042	1704
Average	38	16	5	0.09	0.09	6085	1648

A parallel option is available for systems that have Remote File Sharing installed. See the description of **sar** —**Dc** in Chapter 11.

7.5.1.4 Sar —d

The **sar —d** option reports the activity of block devices.

device Name of the block device(s) that **sar** is monitoring.

%busy Percent of time the device was servicing a transfer request.

avque The average number of requests outstanding during the period of time (measured only when the queue is occupied).

r+w/s Number of read and write transfers to the device per second.

blks/s Number of 512 byte blocks transferred to the device per second.

avwait Average time in milliseconds that transfer requests wait idly in the queue (measured only when the queue is occupied).

avserv Average time in milli-seconds for a transfer request to be completed by the device (for disks this includes seek, rotational latency, and data transfer times).

An example of **sar —d** is as follows:

sysnam modename 3.0 2 *db332* 12/30/85

13:46:28	device	%busy	avque	r+w/s	blks/s	avwait	avserv
13:46:58	hdsk-0	6	1.6	3	5	13.8	23.7
	fdsk-0	93	2.1	2	4	467.8	444.0
13:47:28	hdsk-0	13	1.3	4	8	10.8	32.3
	fdsk-0	100	3.1	2	5	857.4	404.1
13:47:58	hdsk-0	17	.7	2	41	.6	48.1
	fdsk-0	100	4.4	2	6	1451.9	406.5
Average	hdsk-0	12	1.2	3	18	8.4	34.7
	fdsk-0	98	3.2	2	5	925.7	418.2

The above example was taken while transferring data from hard disk (hdsk-0) to floppy disk (fdsk-0). Looking at the results tells us why hard disk technology has improved the performance of super-microcomputers.

Note that queue lengths and wait times are measured while the queue had something on it. If **busy** is small, large queues and service times probably represent the periodic **sync** efforts by the system to ensure that altered blocks are written to the disk in a timely fashion.

7.5.1.5 Sar —m

The **sar —m** option reports on inter-process communication activities. Message and semaphore calls are reported as follows:

msg/s Number of message operations (sends and receives) per second.

sema/s Number of semaphore operations per second.

An example of **sar —m** output follows:

```
sysnam modename 3.0 2 db332   12/30/85

15:16:58   msg/s   sema/s
15:17:32    0.00    0.00
15:18:02    0.00    0.00
15:18:32    0.00    0.00
Average     0.00    0.00
```

These figures will usually be zero (0.00) unless you are running applications that use the message or semaphore features.

7.5.1.6 Sar —q

The **sar —q** option reports the average queue length while the queue is occupied and percent of time occupied.

runq-sz Run queue of processes in memory; typically, this should be less than 2. Consistently higher values mean you are CPU-bound.

%runocc The percentage of time the run queue is occupied; the larger this value is the better.

swpq-sz Swap queue of processes to be swapped out; the smaller this number is the better.

%swpocc The percentage of time the swap queue is occupied; the smaller this value is the better.

An example of **sar —q** follows:

sysnam modename 3.0 2 *db332* 12/30/85

11:00:56	runq-sz	%runocc	swpq-sz	%swpocc
11:01:07	1.7	98	1.5	36
11:01:17	1.0	63	1.0	31
11:01:27	1.0	58	1.0	49
Average	1.3	74	1.2	39

In this example, the processor utilization (%runocc) varies between 58% and 98%, while the fraction of time the swap queue is not empty (%swpocc) is 31% to 49%. This means that memory is not causing a major bottleneck in the system throughput, but more memory would help reduce the swapping/paging activity.

If %runocc is greater than 90 and **runq-sz** is greater than 2, the CPU is heavily loaded and response is degraded. In this case, additional CPU capacity my be required to obtain acceptable system response. If %swpocc is greater than 20, more memory or fewer buffers would help reduce swapping/paging activity.

7.5.1.7 Sar —u

The CPU utilization is listed by **sar —u** (default). At any given moment the processor will be either busy or idle. When busy, the processor will be in either user or system mode. When idle, the processor will either be waiting for input/output completion or has no work to do. The **—u** option of **sar** lists the percent of time that the processor is in system mode (%sys), user mode (%user), waiting for input/output completion (%wio), and idle time (%idle).

In typical timesharing use, %sys and %usr are about the same value. In special applications, either of these may be larger than the other without anything being abnormal. A high %wio generally means a disk bottleneck. A high %idle, with degraded response time, may mean memory constraints; time spent waiting for memory is attributed to %idle.

An example of **sar —u** follows:

sysnam modename 3.0 2 *db332* 12/30/85

09:20:08	%usr	%sys	%wio	%idle
09:40:12	6	7	2	86
10:00:03	7	9	3	80
10:20:07	14	16	10	61
Average	9	11	5	76

A parallel option is available for systems that have Remote File Sharing installed. See the description of **sar —Du** in Section 11.4.6.2.

7.5.1.8 Sar —v

The **—v** option reports the status of process, i-node, file, shared memory record, and shared memory file tables. From this report you know when the system tables need to be modified.

proc-sz Number of process table entries presently being-used/allocated in the kernel.

inod-sz Number of i-node table entries presently being-used/allocated in the kernel.

file-sz Number of file table entries presently being-used/allocated in the kernel.

ov Number of times an overflow occurred. (One column for each of the above three items.)

lock-sz The number of shared memory record table entries presently being-used/allocated in the kernel.

fhdr-sz No longer applicable.

The values are given as *level/table size.* An example of **sar —v** follows:

```
sysnam modename  3.0 2  db332  12/30/85
17:36:05         proc-sz     ov inod-sz ov  file-sz  ov lock-sz fhdr-sz
17:36:35         17/ 40       0 39/ 80  0 29/ 80   0   0/ 50   0/ 0
17:37:05         19/ 40       0 46/ 80  0 35/ 80   0   0/ 50   0/ 0
17:37:35         18/ 40       0 43/ 80  0 34/ 80   0   0/ 50   0/ 0
```

This example shows that all tables are large enough to have no overflows. Sizes could be reduced to save main memory space if these are the highest values ever recorded.

7.5.1.9 Sar —w

The **—w** option reports swapping and switching activity. The following are some target values and observations.

swpin/s Number of transfers into memory per second.

bswin/s Number of 512-byte-block units (blocks) transferred for swap-ins (including initial loading of some programs) per second.

swpot/s Number of transfers from memory to the disk swap area per second. If greater than 1, memory may need to be increased or buffers decreased.

bswot/s Number of blocks transferred for swap-outs per second.

pswch/s Process switches per second. This should be 30 to 50 on a busy 4 to 6 user system.

An example of **sar —w** output follows:

sysnam modename 3.0 2 *db332* 12/30/85

19:53:44	swpin/s	bswin/s	swpot/s	bswot/s	pswch/s
19:53:58	0.0	0.0	0.0	0.0	37
19:54:14	0.0	0.0	0.0	0.0	39
19:54:24	0.0	0.0	0.0	0.0	39
Average	0.0	0.0	0.0	0.0	38

This example shows that there is sufficient memory for the currently active users, since no swapping is occurring.

7.5.1.10 Sar —p

The **—p** option reports paging activity. The following page rates are recorded.

vflt/s Number of address translation page faults per second (valid page not present in memory).

pflt/s Number of page faults from protection errors per second (illegal access to page) or "copy-on-writes." **Pflt/s** generally consists entirely of "copy-on-writes."

pgfil/s Number of **vflt/s** per second satisfied by a page-in from the file system. (Each **pgfil** causes two **lreads**; see **sar —b**).

rclm/s Number of valid pages per second that the system has reclaimed (added to list of free pages).

An example of **sar —p** output follows:

```
sysnam modename    3.0 2   db332        12/30/85
12:01:51           vflt/s    pflt/s    pgfil/s    rclm/s
12:56:52           13.91      2.80       5.63      11.21
```

7.5.1.11 Sar —r

The **—r** option records the number of memory pages and swap file disk blocks that are currently unused. The following are recorded.

freemem Average number of 2K pages of memory available to user processes over the intervals sampled by the command.

freeswap Number of 512K disk blocks available for process swapping.

An example of **sar —r** output follows:

```
sysnam modename 3.0 2 db332    12/30/85

12:01:51 freemem freeswap
12:56:52      208      5848
```

7.5.1.12 Sar —y

The **—y** option monitors terminal device activities. If you have a lot of terminal I/O, you can use this report to determine if there are any bad lines. Activities recorded are defined as follows:

rawch/s Input characters (raw queue) per second.

canch/s Input characters processed by canon (canonical queue) per second.

outch/s Output characters (output queue) per second.

rcvin/s Receiver hardware interrupts per second.

xmtin/s Transmitter hardware interrupts per second.

mdmin/s Modem interrupts per second.

The number of modem interrupts per second (**mdmin/s**) should be close to 0, and the receive and transmit interrupts per second (**xmtin/s** and **rcvin/s**) should be less than or equal to the number of incoming or outgoing characters, respectively. If this is not the case, check for bad lines.

An example of **sar —y** output follows:

sysnam modename 3.0 2 *db332* 12/30/85

16:50:11	rawch/s	canch/s	outch/s	rcvin/s	xmtin/s	mdmin/s
16:50:41	112	15	653	103	102	0
16:51:11	107	7	654	104	105	0
16:51:41	99	5	641	99	105	0
Average	106	9	649	102	104	0

7.5.1.13 Sar —A

The **sar —A** option is equivalent to **sar —udDqbwcayvmprS.** The report includes RFS operations (the **—D** and **—S** options) described in Sections 11.4.6.1 and 11.4.6.3. The **—A** option provides a view of overall system performance. Use it to get a more global perspective. If data from more that one time slice is shown, the report includes averages.

An example of **sar —A** follows:

sysnam modename 3.0 2 *db332* 12/30/85

00:00:00	%usr	%sys local	%sys remote	%wio	%idle			
01:00:00	0	0	0	0	100			
00:00:00	device	%busy	avque	r+w/s	blks/s	avwait	avserv	
01:00:00	hdsk-0	0	1.5	0	0	10.9	21.1	
	hdsk-1	0	2.1	0	0	30.6	27.9	
00:00:00	runq-sz	%runocc	swpq-sz	%swpocc				
01:00:00	1.1	0						
00:00:00	bread/s	lread/s	%rcache	bwrit/s	lwrit/s	%wcache	pread/s	pwrit/s
01:00:00	0	0	97	0	0	67	0	0
00:00:00	swpin/s	bswin/s	swpot/s	bswot/s	pswch/s			
01:00:00	0.00	0.0	0.00	0.0	1			
00:00:00	scall/s	sread/s	swrit/s	fork/s	exec/s	rchar/s	wchar/s	
01:00:00								
in	0	0	0		0.00	0	0	
out	0	0	0		0.00	0	0	
local	0	0	0	0.01	0.01	23	0	
00:00:00	iget/s	namei/s	dirbk/s					
01:00:00	0	0	0					
00:00:00	rawch/s	canch/s	outch/s	rcvin/s	xmtin/s	mdmin/s		
01:00:00	0	0	0	0	0	0		
00:00:00	proc-sz	ov	inod-sz	ov	file-sz	ov	lock-sz	fhdr-sz
01:00:00	12/60	0	33/150	0	21/150	0	0/100	0/0
00:00:00	msg/s	sema/s						
01:00:00	0.00	0.00						
00:00:00	vflt/s	pflt/s	pgfil/s	rclm/s				
01:00:00	0.09	0.08	0.01	0.00				
00:00:00	freemem	freeswp						
01:00:00	362	10196						
00:00:00	serv/lo-hi	request	request	server	server			
	3-6	%busy	avglgth	%avail	avgavail			
01:00:00	0	0.0	0	0.0	0			

7.5.2 Sag

Sag graphically displays the system activity data stored in a binary data file by a previous **sar** run. Any of the **sar** data items may be plotted separately or in combination. **Sag** invokes **sar** and matches strings in the data column header. Figure 7-2 shows a typical **sag** display. Running **sar** will show what data are available.

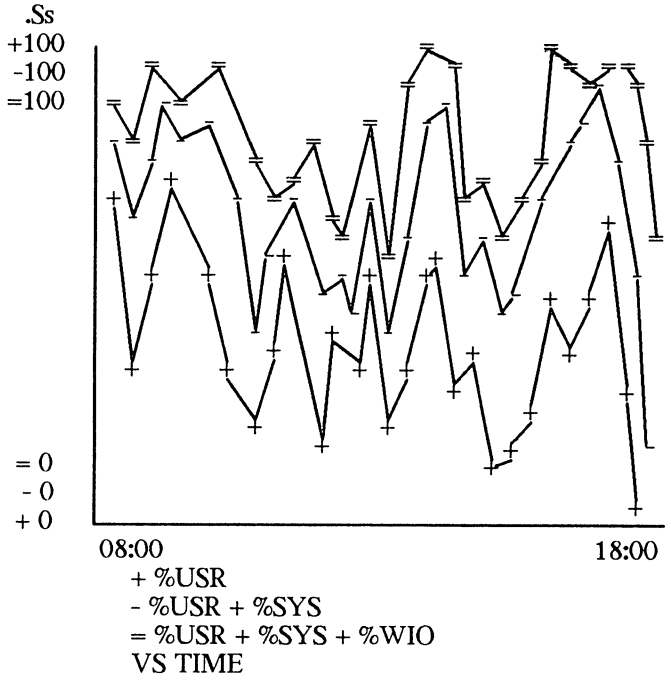


Figure 7-2. Example of sag Output

In Figure 7-2, the processor is completely utilized over three time intervals: 9—10 AM, 1—2 PM, and 3:30—5:30 PM. Remember the actual fraction of time that the processor is busy is the sum of user (%usr) mode time and system (%sys) mode time. When this approaches 100%, the processor is running at its maximum capacity as configured. The sum of %usr + %sys + %wio is about the same as the sum of %usr + %sys (%wio is low). This means that the disk subsystem is able to handle all requests that the processor generates with little delay. From this example, the first place to look to reduce any bottleneck is in reducing processor load.

The **sag** command is useful only if you have a standard output device that can read plotting instructions. Refer to **tplot**(1G) in the *User's Reference Manual* for a list of terminals with this capability.

7.5.3 Timex

The **timex** command times a command and reports the system activities that occurred during the time the command was executing. If no other programs are running, then **timex** can give you a good idea of which resources a specific command uses during its execution. System consumption can be collected for each application program and used for tuning the heavily loaded resources. For our example, the **date** command is used. Enter the following:

```
$ timex —s date
Wed Feb 19 08:32:50 EST 1986

real         0.17
user         0.01
sys          0.09
sysnam modename 3.0 2 db332    12/30/85
08:32:05     %usr      %sys      %sys      %wio      %idle
                       local     remote
08:32:05       10         77         0        13          0
08:32:05  bread/s   lread/s %rcache  bwrit/s  lwrit/s %wcache  pread/s pwrit/s
08:32:05        8       125        94         0        12       100         0       0
08:32:05   device    %busy     avque    r+w/s   blks/s   avwait   avserv
08:32:05   hdsk-0       12       1.0        4        8      0.0     30.0
           hdsk-1        8       1.0        4        8      0.0     20.0
08:32:05 rawch/s  canch/s  outch/s  rcvin/s  xmtin/s  mdmin/s
08:32:05        0         0        56         0        4         0
08:32:05  scall/s   sread/s   swrit/s    fork/s    exec/s    rchar/s   wchar/s
08:32:05
  in            0         0         0                0.00         0         0
  out           0         0         0                0.00         0         0
  local       244        29         8      5.77      7.69     44154       879
08:32:05 swpin/s  bswin/s  swpot/s  bswot/s  pswch/s
08:32:05    0.00       0.0     0.00      0.0       33
08:32:05   iget/s  namei/s  dirbk/s
08:32:05       40        21        42
08:32:05 runq-sz %runocc swpq-sz %swpocc
08:32:05

08:32:05   proc-sz      ov  inod-sz      ov   file-sz ov  lock-sz   fhdr-sz
08:32:05    13/60       0  33/150        0  20/150   0  0/100       0/0
08:32:05    msg/s   sema/s
08:32:05     0.00     0.00
08:32:05    vflt/s   pflt/s  pgfil/s  rclm/s
08:32:05    44.23    46.15     3.85     0.00
08:32:05 freemem freeswp
08:32:05      306    10196
```

```
08:32:05 serv/lo-hi    request    request    server     server
            3-6        %busy      avglgth    %avail     avgavail
08:32:05    0          0.000.00
```

While **date**, for its simplicity, was used for the preceding demonstration, it is not the best example since it is not a major user of system resources.

Timex can be used in the following way:

$ **timex** —**s** *application program*

Your application program will operate normally. When you finish and exit, the **timex** result will be printed on your screen. This can be extremely interesting: you get a clear picture of system resources used by your program.

7.5.4 Sadp

The **sadp** command

sadp [—**th**] [—**d** *device*[—*drive*]] **s** [*n*]

reports disk access locations and seek distance in tabular (—**t**) or histogram (—**h**) form. Disk activity is sampled once every second during a specified interval *n*. An example of **sadp** output for hard disk drive 0 follows.

Using the **sadp** output along with the output of **/etc/mount**(1M) and a table of disk sections (see Appendix A for the default partitioning of your disk(s)), you can identify the file systems with a large amount of I/O activity. In general, try to move areas of high activity close together. This will reduce the number of seeks over large distances.

The first graph (Figure 7-3) shows excellent disk cylinder locality of references. This means that every time the location of a block was referenced by the disk head (for reading or writing), it mostly occurred in the same general region of the disk. In the example, most references (as indicated by percent times referenced) occur for files near cylinders 450 to 600, with a few for files around cylinder 250. There are a few references to other files on the disk, but they occur only a small percent of the time. This graph shows, then, that the most often used files are grouped together in the same general region of cylinders on the disk; the more clustered the stars on the histogram, the better. Another way to say this is that the disk has an excellent file system configuration.

The second graph (Figure 7-4) shows another aspect of an excellent file system configuration: the head seek distance. This refers to the distance the disk head has to move from the current cylinder to the cylinder of the next block referenced. In the example, most physical seeks are under ten cylinders. Specifically, some 14% of the seeks occurred within a distance of 0 to 8 cylinders, and some 17% of the seeks occurred within a distance of 8 to 16 cylinders. This means that for approximately one-third of the disk activity, the disk head was forced to move no more that 16 cylinders to reference a given block, and the more to the left the stars are

```
$ sadp —h —d hdsk-0 3600<CR>
CYLINDER ACCESS HISTOGRAM
disk-0:
Total transfers = 33291
```

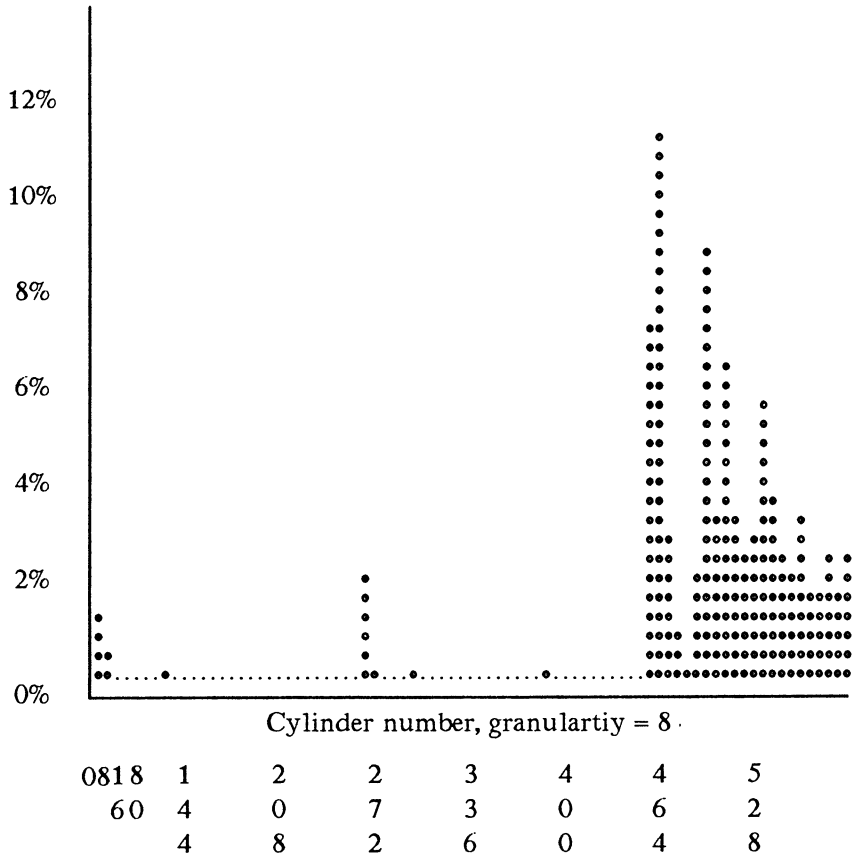


Figure 7-3. Output from Sadp: Cylinder Access Histogram

grouped, the better.

These two graphs give an idea of how finely you can tune your system. If, after a working period of weeks or months, you can identify which file systems are consistently the most active, you might consider repartitioning your disks to achieve the maximum from disk access activity (see Chapters 5 and 6 for more information).

7.6 TUNABLE PARAMETERS

Tunable system parameters are used to set various table sizes and system thresholds to handle the expected system load. Caution should be used when changing these variables since such changes can directly affect system performance. For the most part, the initial tunable parameter values for a new *Series 32000* Computer are acceptable for most configurations and applications. If your application has special performance needs, you may have to experiment with different combinations of parameter values to find an optimal set.

Table 7-1 shows the recommended tunable parameter values for a Release 3.0 system equipped with different sizes of Random Access Memory (RAM). The parameters shown in the figure are defined in the following five files:

- **/etc/master.d/kernel**
- **/etc/master.d/disp**
- **/etc/master.d/msg**
- **/etc/master.d/sem**
- **/etc/master.d/shm**

The default parameter settings defined in the **/etc/master.d/kernel** and **/etc/master.d/disp** files are delivered with the core package. Parameters for Remote File Sharing (RFS) and Networking Support Utilities (NSU) are present in the core package, but should be left at their default setting of 0; non-zero settings are automatically provided when the RFS and NSU packages are installed. Additional RFS tunables are described in Chapter 10, Remote File Sharing. The message, semaphore, and shared memory parameters are distributed with the Inter-Process Communications Utilities, in **/etc/master.d/msg, /etc/master.d/sem,** and **/etc/master.d/shm,** respectively. The following notes apply to Table 7-1:

- The value of a few parameters are calculated each time a new kernel (**/unix**) is generated, unless the value is manually overridden (see note below).
- All the other parameter are set to specific values, as defined in the appropriate **/etc/master.d** file. The default value and the size in bytes for each entry are shown in the figure.
- A dash (—) is used in the size information to indicate parameters that set flags in the kernel. Parameters that set flags do not affect the size of the kernel when their values are changed; only the values of the specific flags are changed.
- No values are shown for the RFS and NSU parameters for a 1-megabyte system. RFS and NSU are not installable on systems with less than 2 megabytes of memory. When the RFS and NSU packages are installed, their respective parameters are set to their 2-

SEEK DISTANCE HISTOGRAM
diks-0:
Total seeks = 30308

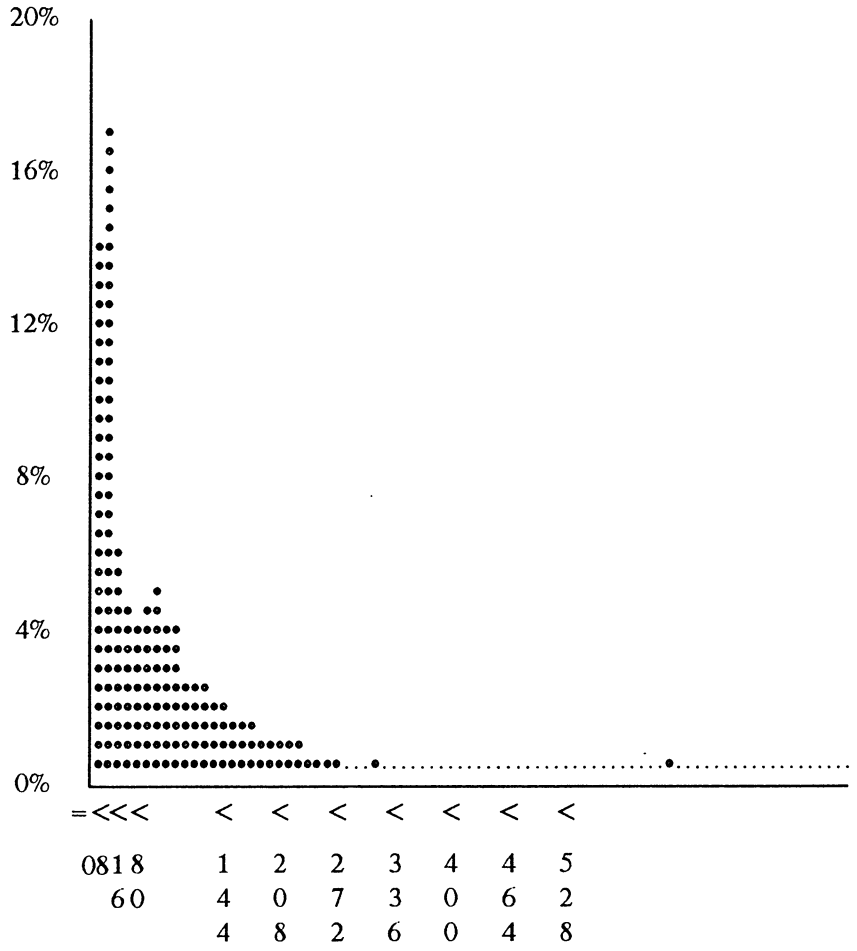


Figure 7-4. Output from Sadp: Seek Distance Histogram

TABLE 7-1. SUGGESTED PARAMETER VALUES: GENIX V.3 SYSTEMS

PARAMETER	EQUIPPED RAM				DEFAULT VALUE	SIZE PER ENTRY IN BYTES
	1 MEG	2 MEGS	3 MEGS	4 MEGS		
NBUF	100	250	400	600	CALCULATED	1076
NCALL	60	60	60	60	60	16
NINODE	100	225	300	400	150	68
NS5INODE	100	225	300	400	150	64
NFILE	100	225	300	400	150	12
NMOUNT	25	25	25	25	25	36
NPROC	60	70	80	100	60	168
NREGION	210	245	280	350	210	36
NCLIST	120	120	160	200	120	72
MAXUP	25	25	30	30	25	—
NOFILES	20	20	20	20	20	—
NHBUF	32	64	128	256	64	12
NPBUF	20	20	20	20	20	52
NAUTOUP	10	10	15	15	10	—
BDFLUSHR	1	1	1	1	1	—
MAXPMEM	0	0	0	0	CALCULATED	—
SHLBMAX	2	2	2	2	2	12 x NPROC
FLCKREC	100	100	100	100	100	28
PUTBUFSZ	2000	2000	2000	2000	2000	—
NSRMOUNT	—	50	50	50	0	28
REL	3.0	3.0	3.0	3.0	3.0	—
NODE	unix	unix	unix	unix	unix	—
SYS	unix	unix	unix	unix	unix	—
VER	2	2	2	2	2	—
MAXSLICE	100	100	100	100	100	—

TABLE 7-1. (Cont)

PARAMETER	EQUIPPED RAM				DEFAULT VALUE	SIZE PER ENTRY IN BYTES
	1 MEG	2 MEGS	3 MEGS	4 MEGS		
VHNDFRC	16	16	16	16	16	—
VHANDR	1	1	1	1	1	—
GPGSLO	25	25	25	25	25	—
GPGSHI	40	40	40	40	40	—
MAXUMEM	8192	8192	8192	8192	8192	—
GPGSMSK	0x00000220	0x00000220	0x00000220	0x00000220	0x00000220	—
MAXFC	1	1	1	1	1	—
MAXSC	1	1	1	1	1	—
MINARMEM	25	40	40	40	25	—
MINASMEM	25	40	40	40	25	—
NQUEUE	—	256	384	384	0	36
NSTREAM	—	32	48	48	0	52
NMUXLINK	—	32	48	48	0	12
NSTREVENT	—	256	256	256	0	12
MAXSEPGCNT	—	1	2	2	0	2048
NBLK4096	—	0	0	0	0	4142
NBLK2048	—	20	40	40	0	2094
NBLK1024	—	12	32	32	0	1070
NBLK512	—	8	18	18	0	558
NBLK256	—	16	48	48	0	302
NBLK128	—	64	128	128	0	174
NBLK64	—	256	256	256	0	110
NBLK16	—	128	256	256	0	61
NBLK4	—	128	128	128	0	50
NSTRPUSH	—	9	9	9	9	—
STRLOFRAC	—	80	80	80	80	—
STRMEDFRAC	—	90	90	90	90	—
STRMSGSZ	—	4096	4096	4096	4096	—
STRCTLSZ	—	1024	1024	1024	1024	—
NLOG	—	3	3	3	3	12
BSIZE	—	20	20	20	20	—

TABLE 7-1. (Cont)

PARAMETER	EQUIPPED RAM				DEFAULT VALUE	SIZe PER ENTRY IN BYTES
	1 MEG	2 MEGS	3 MEGS	4 MEGS		
MSGMAP	100	100	100	100	100	8
MSGMAX	2048	2048	2048	2048	2048	—
MSGMNB	4096	4096	4096	4096	4096	—
MSGMNI	50	50	50	50	50	53
MSGSSZ	8	8	8	8	8	1024
MSGTQL	40	40	40	40	40	12
MSGSEG	1024	1024	1024	1024	1024	8
SEMMAP	10	10	10	10	10	8
SEMMNI	10	10	10	10	10	32
SEMMNS	60	60	60	60	60	8
SEMMNU	30	30	30	30	30	8X(SEMUME+2)
SEMMSL	25	25	25	25	25	—
SEMOPM	10	10	10	10	10	8
SEMUME	10	10	10	10	10	8XSEMMNU
SEMVMX	32767	32767	32767	32767	326767	—
SEMAEM	16384	16384	16384	16384	16384	—
SHMMAX	131072	131072	131072	131072	131072	—
SHMMIN	1	1	1	1	1	—
SHMMNI	100	100	100	100	100	52
SHMSEG	6	6	6	6	6	12XNPROC
SHMALL	512	512	512	512	512	—

megabyte recommendations. In Table 7-1, the default values shown assume that neither of these packages is installed.

NOTE: A calculated parameter value is overridden by adding the parameter name and value to the appropriate **/etc/master.d** file. Overriding the calculated value causes the parameter to be set to the new value each time a new kernel (**/unix**) is generated. When calculated parameter values are overridden, however, any subsequent changes in the hardware configuration (adding new memory for example) require a change of the value that was manually set. This will allow you to optimize the performance of the new configuration.

7.6.1 Kernel Parameters

The following parameters are defined in the **/etc/master.d/kernel** file.

NBUF Specifies how many system buffers to allocate. The GENIX V.3 system buffers form a data cache. The data cache is a memory array containing disk file information. Improvement in the hit rate of this cache increases with the number of buffers. Cache hits reduce the number of disk accesses and thus improve overall performance. The entries are normally in the range of 100 to 600. Each buffer contains 1076 bytes. Hash buffers (NHBUF) should be increased along with system buffers (NBUF) for optimal performance. The value for NBUF is calculated automatically by the system, unless specifically set to a value by adding a line (NBUF = *n*) to the **/etc/master.d/kernel** file.

NCALL Specifies how many call-out table entries to allocate. Each entry represents a function to be invoked at a later time by the clock handler portion of the kernel. This value must be greater than 2 and is normally in the range of 10 to 70. The default value is 60. Each entry contains 16 bytes.

Software drivers may use call entries to check hardware device status. When the call-out table overflows, the system crashes and outputs the following message on the system console:

PANIC: Timeout table overflow

NINODE Specifies how many i-node table entries to allocate. Each table entry represents an in-core i-node that is an active file. For example, an active file might be a current directory, an open file, or a mount point. The file control structure is modified when changing this variable. The number of entries used depends on the number of opened files. The entries are normally in the range of 100 to 400. The value for NINODE pertains directly to the NFILE value. (NINODE is equal to or greater than NFILE). NINODE must always be less than or equal to NS5INODE. NINODE greater than NS5INODE results in an unusable system. When the i-node table overflows, the following warning message is output on the system console:

WARNING: i-node table overflow

NS5INODE NS5INODE must always be equal to or greater than NINODE.

NFILE Specifies how many open file table entries to allocate. Each entry represents an open file. The entry is normally in the range of 100 to 400. Each entry contains 12 bytes. The NFILE entry relates directly to the NINODE entry. (NFILE is less than or equal to NINODE). The NFILE control structure operates in the same manner as the NINODE structure. When the file table overflows, the following warning message is output on the system console.

NOTICE: file table overflow

As a reminder, this parameter does not affect the number of open files per process (see the NOFILES parameter).

NMOUNT Specifies how many mount table entries to allocate. Each entry represents a mounted file system. The root (/) file system is always the first entry. When full, the **mount**(2) system call returns the error EBUSY. Since the mount table is searched linearly, this value should be as low as possible.

NPROC Specifies how many process table entries to allocate. Each table entry represents an active process. The swapper is always the first entry and **/etc/init** is always the second entry. The number of entries depends on the number of terminal lines available and the number of processes spawned by each user. The average number of processes per user is in the range of 2 to 5 (also see MAXUP, default value 25). When full, the **fork**(2) system call returns the error EAGAIN. The NPROC entry is in the range of 50 to 200.

NREGION Specifies how many region table entries to allocate. Each NREGION entry contains 36 bytes. Most processes have 3 regions: text, data, and stack. Additional regions are needed for each shared memory segment and shared library (text and data) attached. However, the region table entry for the text of a "shared text" program will be shared by all processes executing that program. Each shared memory segment attached to one or more processes uses another region table entry. A good starting value for this parameter is about 3.5 times NPROC. If the system runs out of region table entries, the following message is output on the system console.

Region table overflow

NCLIST Specifies how many character list buffers to allocate. Each buffer contains up to 64 bytes. The buffers are dynamically linked to form input and output queues for the terminal lines and other slow speed devices. The average number of buffers needed per terminal is in the range of 5 to 10. Each entry (buffer space plus header) contains 72 bytes. When full, input and output characters dealing with terminals are lost, although echoing continues.

MAXUP Specifies how many concurrent processes a non-superuser is allowed to run. The entry is normally in the range of 15 to 25. This value should not exceed the value of NPROC (NPROC should be at least 10% more than MAXUP). This value is per user identification number, not per terminal. For example, if 12 people are logged in on the same user identification, the default limit would be reached very quickly.

NOFILES Specifies the maximum number of open files per process. Default is 20. Values higher than 20 are accessible only to processes using system calls (**open**(2), **creat**(2); for example). Processes using standard I/O subroutines are limited to 20, independent of the value of NOFILES. Unless an application package recommends that NOFILES be changed, the default setting of 20 should be left as is.

/bin/sh uses 3 file table entries: standard input, standard output, and standard error (0, 1, and 2 are normally reserved for stdin, stdout, and stderr, respectively). This leaves the value of NOFILES minus 3 as the number of other open files available per process. If a process requires up to three more than this number, then the standard files must be closed. This practice is NOT recommended, and must be used with caution, if at all.

If the configured value of NOFILES is greater than the maximum (100) or less than the minimum (20), the configured value is set to the default (20), and a NOTICE message is sent to the console.

NHBUF Specifies how many "hash buckets" to allocate. These are used to search for a buffer given a device number and block number rather than a linear search through the entire list of buffers. **This value must be a power of 2.** Each entry contains 12 bytes. The NHBUF value must be chosen so that the value NBUF divided by NHBUF is approximately equal to 4. While the value of NBUF is normally calculated by the system, that is not true for NHBUF. NHBUF must be specified in **/etc/master.d/kernel.**

NPBUF Specifies how many physical input/output buffers to allocate. One input/output buffer is needed for each physical read or write active. Each entry contains 52 bytes. The default value is 20.

NAUTOUP The NAUTOUP entry specifies the buffer age in seconds for automatic file system updates. A system buffer is written to the hard disk when it has been memory-resident for the interval specified by the NAUTOUP parameter. Specifying a smaller limit increases system reliability by writing the buffers to disk more frequently and decreases system performance. Specifying a larger limit increases system performance at the expense of reliability.

BDFLUSHR Specifies the rate in seconds for checking the need to write the file system buffers to disk. The default is 1 second.

MAXPMEM Specifies the maximum amount of physical memory to use in pages. The default value of 0 specifies that all available physical memory be used.

SHLBMAX Specifies the maximum number of shared libraries that can be attached to a process at one time.

FLCKREC Specifies the number of records that can be locked by the system. The default value is 100. Each entry contains 28 bytes.

PUTBUFSZ Specifies the size of a circular buffer, **putbuf**, that is used to contain a copy of the last PUTBUFSZ characters written to the console bu the operating system. The contents of **putbuf** can be viewed using **crash**(1M).

REL Specifies the GENIX V.3 system release.

NODE Specifies the node name of the system. The default node name is **unix** (see Procedure 1.3, Establish or Change System and Node Names).

SYS Specifies the system name. The default system name is **unix** (see Procedure 1.3, Establish or Change System and Node Names).

VER Specifies the version. This value may be 1 or 2.

MAXSLICE Specifies in clock ticks the maximum time slice for user processes. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it MAXSLICE clock ticks. MAXSLICE, defined in **/etc/master.d/disp**, is normally one second (100 clock ticks on the *Series 32000*).

7.6.2 Paging Parameters

There exists in the system a paging daemon, **vhand**, whose sole responsibility is to free up memory as the need arises. It uses a "least recently used" algorithm to approximate process working sets, and it writes those pages out to disk that have not been touched during some period of time. The page size is 2048 bytes. When memory is exceptionally tight, the working sets of entire processes may be swapped out.

The following tunable parameters determine how often **vhand** runs and under what conditions. The default values in **/etc/master.d/kernel** should be adequate for most applications.

VHNDFRC Used to determine the initial value for the system variable VHANDL. VHANDL is set to the maximum user-available memory divided by VHNDFRC or the value of GPGSHI, whichever is larger. The value of VHANDL determines when the paging daemon **vhand** runs. The amount of available free memory is compared with the value of VHANDL every VHANDR seconds. If free memory is less than VHANDL, then the paging daemon **vhand** is awakened.

The default for VHNDFRC is 16. Decrease the value to make the daemon more active; increase the value to make the daemon less active (must be > 0 and < 25 percent of available memory).

VHANDR Specifies in seconds the maximum rate at which **vhand** can run. **vhand** will only run at this rate if free memory is less than VHANDL, as explained above for VHNDFRC. The default is 1. Increase the value to make the daemon less active (must be an integer > 0 and ⩽ 300). If you have set the value higher, decreasing it makes the daemon more active.

GPGSLO Specifies the low water mark of free memory in pages for **vhand** to start stealing pages from processes. The default is 25. Increase the value to make the daemon more active; decrease the value to make the daemon less active (must be an integer ≥ 0 and $<$ GPGSHI).

GPGSHI Specifies the high water mark of free memory in pages for **vhand** to stop stealing pages from processes. The default is 40. Increase the value to make the daemon more active; decrease the value to make the daemon less active (The value must be an integer > 0, $>$ GPGSLO and $<$ 25 percent of the number of pages of available memory).

GPGSMSK Mask used by the paging daemon to determine the required number of aging passes before stealing a page. The default is 0x00000220. This value should not be changed.

MAXSC Specifies the maximum number of pages which will be swapped out in a single operation. The default value is 1.

MAXFC Specifies the maximum number of pages that will be added to the freelist in a single operation. The default value is 1.

MAXUMEM Specifies the maximum size of a user's virtual address space in pages. This value cannot be greater than 8192. The default is 8192.

MINARMEM Specifies the minimum number of memory pages reserved for the text and data segments of user processes.

MINASMEM Threshold value that specifies the number of memory and swap pages reserved for system purposes (unavailable for the text and data segments of user processes).

7.6.3 Streams Parameters

The following tunable parameters are associated with Streams processing. These parameters are defined in the **/etc/master.d/kernel** file. The values should be left at 0 unless the NSU package has been installed or you're planning on using STREAMS.

NQUEUE The number of STREAMS queues to be configured. Queues are always allocated in pairs, so this number should be even. A minimal Stream contains four queues (two for the Stream head, two for the driver). Each module pushed on a Stream requires an additional two queues. A typical configuration value is 8*NSTREAM.

NSTREAM The number of "Stream-head" (stdata) structures to be configured. One is needed for each Stream opened, including both Streams currently open from user processes and Streams linked under multiplexers. The recommended configuration value is highly application-dependent, but a value of 32-40 usually suffices on a *Series 32000* Computer for running a single transport provider with moderate traffic.

NSTRPUSH The maximum number of modules that may be pushed onto a Stream. This is used to prevent an errant user process from consuming all of the available queues on a single Stream. By default this value is 9, but in practice, existing applications have pushed at most four modules on a Stream.

NSTREVENT The initial number of Stream event cells to be configured. Stream event cells are used for recording process-specific information in the **poll**(2) system call. They are also used in the implementation of the STREAMS I_SETSIG **ioctl** and in the kernel **bufcall**() mechanism. A rough minimum value to configure would be the expected number of processes to be simultaneously using **poll**(2) times the expected number of Streams being polled per process, plus the expected number of processes expected to be using STREAMS concurrently. The default is 256. Note that this number is not necessarily a hard upper limit on the number of event cells that will be available on the system (see MAXSEPGCNT).

MAXSEPGCNT The number of additional pages of memory that can be dynamically allocated for event cells. If this value is 0, only the allocation defined by NSTREVENT is available for use. If the value is not 0 and if the kernel runs out of event cells, it will under some circumstances attempt to allocate an extra page of memory from which new event cells can be created. MAXSEPGCNT places a limit on the number of pages that can be allocated for this purpose. On a *Series 32000* Computer, each new page can provide 166 event cells. Once a page has been allocated for event cells, however, it cannot be recovered later for use elsewhere. It is recommended that the NSTREVENT value be set to accommodate most load conditions, and that MAXSEPGCNT be set to 1 to handle exceptional load cases should they arise.

NMUXLINK The maximum number of multiplexer links to be configured. One link structure is required for each active multiplexor link (STREAMS I_LINK **ioctl**). This number is application dependent; the default allocation equal to the number of Streams (NSTREAM) guarantees availability of links.

STRMSGSZ The maximum allowable size of the data portion of any STREAMS message. This should usually be set just large enough to accommodate the maximum packet size restrictions of the configured STREAMS modules. If it is larger than necessary, a single **write**(2) or **putmsg**(2) can consume an inordinate number of message blocks. The recommend value of 4096 is sufficient for existing applications.

STRCTLSZ The maximum allowable size of the control portion of any STREAMS message. The control portion of a **putmsg**(2) message is not subject to the constraints of the min/max

packet size, so the value entered here is the only way of providing a limit for the control part of a message. The recommended value of 1024 is more than sufficient for existing applications.

NBLK*n* — The number of STREAMS data blocks and buffers to be allocated for each size class. Message block headers are also allocated based on these numbers: the number of message blocks is 1.25 times the total of all data block allocations. This provides a message block for each data block, plus some extras for duplicating messages (kernel functions **dupb**(), **dupmsg**()). The optimal configuration depends on both the amount of primary memory available and the intended application. The default values provided in the NSU package are intended to support a moderately loaded configuration using RFS and UUCP/CU over STARLAN.

STRLOFRAC — The percentage of data blocks of a given class at which low-priority block allocation requests are automatically failed. For example, if STRLOFRAC is 80 and there are 48 256-byte blocks, a low-priority allocation request will fail when more than 38 256-byte blocks are already allocated. The parameter is used to help prevent deadlock situations by starving out low-priority activity. The recommended value of 80 works well for current applications. STRLOFRAC must always be in the range 0 <= STRLOFRAC <= STRMEDFRAC.

STRMEDFRAC — The percentage cutoff at which medium priority block allocations are failed (see STRLOFRAC discussion above). The recommended value of 90 works well for current applications. STRMEDFRAC must always be in the range STRLOFRAC <= STRMEDFRAC <= 100.

NOTE: There is no cutoff fraction for high-priority allocation requests; it is effectively 100.

7.6.4 Log Driver Parameters

The configurable parameters for the log driver are found in the file **/etc/master.d/log**. They are:

NLOG — The number of minor devices to be configured for the log driver; the active minor devices will be 0 through (NLOG—1). The recommended value of 3 services an error logger (**strerr**(1M)) and a trace command (**strace**(1M)), with one left over for miscellaneous usage. If only an error logger

and a tracer are to be supported, this number can be set to 2. If there are several daemons for an application that may be submitting log messages, this number can be increased to accommodate the extra users.

BSIZE This number controls the number of log messages that will be kept in the log driver after being sent upstream to the error or trace logging process. Duplicates or only the last BSIZE messages are kept at the log driver; older messages are freed as new ones are submitted. The intent of this feature is to hold onto the last few log messages in case a system crash prevents their being written to the log file. A system dump analysis tool can then be used to look at these messages to determine if they point to the cause of the crash. The recommended number of 20 is more than sufficient for most burst-traffic situations.

7.6.5 Message Parameters

The following tunable parameters are associated with inter-process communication messages. These parameters are defined in the **/etc/master.d/msg** file. The order in which they are described follows the order in which they are defined in the output of the **/etc/sysdef** command.

MSGMAP Specifies the size of the control map used to manage message segments. Default value is 100. Each entry contains 8 bytes.

MSGMAX Specifies the maximum size of a message. The default value is 2048. The maximum size is 64 kilobytes —1.

MSGMNB Specifies the maximum length of a message queue. The default value is 4096.

MSGMNI Specifies the maximum number of message queues system-wide (id structure). The default value is 50.

MSGSSZ Specifies the size, in bytes, of a message segment. Messages consist of a contiguous set of message segments large enough to fit the text. The default value is 8. The value of MSGSSZ times the value of MSGSEG must be less than or equal to 131,072 bytes (128 kilobytes).

MSGTQL Specifies the number of message headers in the system and, thus, the number of outstanding messages. The default value is 40. Each entry contains 12 bytes.

MSGSEG Specifies the number of message segments in the system. The default value is 1024. The value of MSGSSZ times the value of MSGSEG must be less than or equal to 131,072 bytes (128 kilobytes).

7.6.6 Semaphore Parameters

The following tunable parameters are associated with inter-process communication semaphores. These parameters are defined in the **/etc/master.d/sem** file. The order in which they are described follows the order in which they are defined in the output of the **/etc/sysdef** command.

SEMMAP Specifies the size of the control map used to manage semaphore sets. The default value is 10. Each entry contains 8 bytes.

SEMMNI Specifies the number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time. The default value is 10. Each entry contains 32 bytes.

SEMMNS Specifies the number of semaphores in the system. The default value is 60. Each entry contains 8 bytes.

SEMMNU Specifies the number of undo structures in the system. The default value is 30. The size is equal to 8 x (SEMUME + 2) bytes.

SEMMSL Specifies the maximum number of semaphores per semaphore identifier. The default value is 25.

SEMOPM Specifies the maximum number of semaphore operations that can be executed per **semop**(2) system call. The default value is 10. Each entry contains 8 bytes.

SEMUME Specifies the maximum number of undo entries per undo structure. The default value is 10. The size is equal to 8*(SEMMNU) bytes.

SEMVMX Specifies the maximum value a semaphore can have. The default value is 32767. The default value is the maximum value for this parameter.

SEMAEM Specifies the adjustment on exit for maximum value, alias **semadj**. This value is used when a semaphore value becomes greater than or equal to the absolute value of **semop**(2), unless the program has set its own value. The default value is 16384. The default value is the maximum value for this parameter.

7.6.7 Shared Memory Parameters

The following tunable parameters are associated with inter-process communication shared memory. These parameters are defined in the **/etc/master.d/shm** file. The order in which they are described follows the order in which they are defined in the output of the **/etc/sysdef** command.

SHMMAX Specifies the maximum shared memory segment size. The default value is 131072.

SHMMIN Specifies the minimum shared memory segment size. The default value is 1.

SHMMNI Specifies the maximum number of shared memory identifiers system wide. The default value is 100. Each entry contains 52 bytes.

SHMSEG Specifies the number of attached shared memory segments per process. The default value is 6. The maximum value is 15.

SHMALL Specifies the maximum number of in-use shared memory text segments. The default value is 512.

7.6.8 Remote File Sharing Parameters

RFS parameters are discussed in Section 11.4.7. NSRMOUNT is included in Table 7-1 because it is located in the **/etc/master.d/kernel** file. The values associated with NSRMOUNT are 0 unless the RFS package is installed. The rest of the RFS parameters are located in **/etc/master.d/du**.

Chapter 8

LINE PRINTER ADMINISTRATION

8.1 INTRODUCTION

This chapter tells you about

- How the LP Spooling system works
- How to find instructions for installing the LP Spooler
- The commands used to administer the system
- Printer interface programs
- LP Spooler files and directories

Error messages issued by the LP Spooler are listed in Appendix C.

8.1.1 How the LP Spooling System Works

The term **spool** is an acronym for "simultaneous peripheral output on-line." LP spooling means that you can send a file to be printed (LP originally stood for Line Printer, but has come to include many other types of printing devices), while you continue with other work. The LP Spooling system is software that

- handles the task of receiving files users want printed
- schedules the work of one or more printers
- starts programs that interface with the printer(s)
- keeps track of the status of jobs
- issues error messages when problems arise

The LP Spooler has five user commands. These are shown in Table 8-1.

In addition to being able to send requests to the LP Spooling system, check the status of requests, and cancel requests, users are given the ability to disable and enable a printer. The idea is that if a user finds a printer is malfunctioning in some way, it should not be necessary to call the administrator to turn the printer off.

TABLE 8-1. USER COMMANDS FOR THE LP SPOOLING SYSTEM

COMMAND	DESCRIPTION
enable(1)	Activate the named printer(s).
cancel(1)	Cancel a request for a file to be printed.
disable(1)	Deactivate the named printer(s).
lp(1)	Send a file or files to a printer.
lpstat(1)	Print the status of the LP system.

8.2 ADMINISTRATIVE COMMANDS

A separate set of commands available for the LP administrator is shown in Table 8-2. These commands are found in the **/usr/lib** directory. If you expect to use them frequently, you might find it convenient to include that directory in your PATH variable. To use the administrative commands, you must be logged in either as **root** or as **lp. Lp** is a system login (see Chapter 2 or Procedure 1.4 for a description of how to set up a password for a system login).

In Table 8-2 the administrative commands are listed in the order in which they occur in the *Administrator's Reference Manual.* In the section that follows, we describe the commands in the order in which they are usually used.

8.2.1 Command Descriptions and Examples

8.2.1.1 /usr/lib/lpadmin

The **lpadmin**(1M) command is used to add a new printer to the system, assign classes of printers, name or remove a default destination, and specify interface programs to be used. **Lpadmin** may not be used when the LP scheduler, **lpsched**(1M), is running, except when the **—d** option is specified.

TABLE 8-2. ADMINISTRATIVE COMMANDS FOR THE LP SPOOLING SYSTEM

COMMAND	DESCRIPTION
/usr/lib/accept(1M)	Permit job requests to be queued for a specified destination.
/usr/lib/reject(1M)	Prevent jobs from being queued for a specified destination. Described on the same manual page as **accept**(1M).
/usr/lib/lpadmin(1M)	Set up or change the LP configuration.
/usr/lib/lpmove(1M)	Move output requests from one destination to another. Described on the same manual page as **lpsched**(1M).
/usr/lib/lpsched(1M)	Start the LP scheduler.
/usr/lib/lpshut(1M)	Stop the LP scheduler. Described on the same manual page as **lpsched**(1M).

One (and only one) of the following three options must always be included on the command line when you execute **lpadmin:**

—**d**[*dest*]
—**x***dest*
—**p***printer*

No other options are allowed with —**d** and —**x**. However, many arguments are allowed with the —**p***printer* option, and at least one argument must always be present. The —**p** option names a printer to which the other argument(s) applies. If *printer* does not exist, it is created. The arguments that can be used with —**p** are as follows:

—**c***class* — This argument assigns the printer specified in the —**p** option to the specified *class*.

—**e***printer* — This argument allows you to use an existing interface program for a new printer that you are adding to the LP system. When you select this argument, the interface program for the printer specified in this argument is copied for the printer specified in the —**p** option.

—**h** — When adding a new printer, this argument means that the printer is hard-wired to the *Series 32000* Computer.

—i*interface*	Use this argument to specify a new interface program for the printer specified in the —**p** option. *Interface* is the path name of the new program.
—**l**	When adding a new printer, this argument means that the device associated with the printer is a login terminal.
—**m***model*	Several "model" interface programs are supplied with the LP Spooling Utilities. These model interface programs support some of the common printers that may be used with the *Series 32000* Computer. Use this argument to select the model interface program that you want to use with the printer you are adding to the LP system.
—**r***class*	Use this argument to remove a printer from a class.
—**v***device*	This argument must be used when you add a new printer to the LP system. It associates the printer with the GENIX V.3 system file specified by *device*. The complete path name must be given for the file.

The —**d**[*dest*] option is used to define an existing system destination as the new system default destination. If *dest* is not specified, there is no default destination. The LP scheduler may be running when you use this option. The default destination is used to determine where a file named in a user's **lp** command is sent (assuming the user does not specify a destination on the command line and the environment variable LPDEST is null). The destination (*dest*) must already exist.

To remove a destination (*dest*), the —**x***dest* option is used with **lpadmin**. This option cannot be invoked when the scheduler is running. If it is running, you must issue the **lpshut** command before **lpadmin**.

8.2.1.2 Command Examples

NOTE: In examples 2 through 7, it is assumed that the LP scheduler has already been stopped. This is done through the **lpshut**(1M) command. Example 1 does not require the scheduler to be stopped since only the —**d** option to **lpadmin** is used.

Example 1: Make printer dqp10_1 the system default destination.

```
# lpadmin —ddqp10_1
#
```

Example 2: Add a new printer called dqp10_2 and associate it with device **/dev/tty11**. Use the dqp10 model interface program.

```
# lpadmin —pdqp10_2 —v/dev/tty11 —mdqp10
#
```

When you add a new printer, it is left in a disabled state and does not accept requests.

Example 3: Create a hardwired printer called lp1 on device **/dev/tty13**. Add lp1 to a new class called cl1, and use the same interface program that is used with printer lqp40_1.

```
# lpadmin —plp1 —v/dev/tty13 —elqp40_1 —ccl1
#
```

Example 4: Change the interface program for printer lp1 to model interface program dqp10.

```
# lpadmin —plp1 —mdqp10
#
```

Example 5: Add printer dqp10_2 to class cl1:

```
# lpadmin —pdqp10_2 —ccl1<CR>
#
```

Printers that are added to a class are ordered according to the sequence in which they are added. For example, assume that class cl1, in the example above, already had printers lp1 and lp2 as members. After adding printer dqp10_2, the order of the printers would be lp1, lp2, and dqp10_2. If all three printers are available, and a request is routed to class cl1, the request will be serviced by lp1. If all three printers are busy, the request will be serviced by the first available printer.

Example 6: Remove printers lp1 and lp2 from class cl1:

```
# lpadmin —plp1 —rcl1
# lpadmin —plp2 —rcl1
#
```

Example 7: No destination (class or printer) may be removed if it has pending requests. The pending requests must either be cancelled using the **cancel**(1) command or moved to other destinations using the **lpmove**(1M) command before the destination can be removed.

Removing the last remaining member of a class causes the class to be deleted. If the destination removed is the system default destination, the system will no longer have a default destination. However, the removal of a class does not imply the removal of printers that were assigned to that class.

```
# lpadmin —xlp3
#
```

8.2.1.3 /usr/lib/lpsched

The **lpsched**(1M) command starts the LP scheduler. The LP scheduler takes the top job request off the queue and "hands" it to the appropriate interface program to be printed on a printer. The LP scheduler keeps track of the job progress, and as soon as the job is completed, it takes the next job request off the queue and repeats the process. As long as the LP scheduler is running, jobs requested by **lp** will be printed. If the scheduler is not running, jobs will not be printed.

The LP scheduler is started automatically each time the system is turned on. This is done through an executable file called **lp** in the **/etc/rc.d** directory. The **lp** file is created when the LP Spooling system is installed.

Every time the scheduler is started, **lpsched** creates a file called SCHEDLOCK in the **/usr/spool/lp** directory. As long as the SCHEDLOCK file is present, the system will not allow another scheduler to run. When the scheduler is stopped under normal conditions, either with **lpshut**(1M) or as part of the normal shutdown procedure, the SCHEDLOCK file is removed. However, if the system comes down abnormally, there is a possibility that the SCHEDLOCK file may not get removed. To ensure that the SCHEDLOCK file does not exist, **/etc/rc.d/lp** contains a command line to remove SCHEDLOCK first before it attempts to start the scheduler.

The command is entered without arguments.

```
# lpsched
#
```

Notice in the example that the command shows no response to let you know that the scheduler is running. To verify that the scheduler is running, use the **lpstat**(1M) command with the **—r** option.

```
# lpstat —r
scheduler is running
#
```

NOTE: If many job requests are queued, there may be a delay before the **lpstat** command reports that the scheduler is running.

8.2.1.4 /usr/lib/lpshut

Two of the three **lpadmin** command options (**—x** and **—p**) cannot be executed unless the LP scheduler is stopped. The **lpshut** command stops the LP scheduler and terminates all printing activity. All requests that were in the middle of printing will be reprinted in their entirety when the scheduler is restarted. The command is entered without arguments.

```
# lpshut
scheduler stopped
#
```

8.2.1.5 /usr/lib/lpmove

Occasionally, you may find it necessary to move output requests from one destination to another. For example, if you have a printer that was removed for repairs, you will want to move all the pending job requests to a destination with a working printer. This is done using the **lpmove** command. Be aware that job requests routed to a destination without a printer are automatically rejected.

Another use of the **lpmove** command is to move specific requests from one destination to another. When this is done, **lp** will no longer accept requests for the original destination (this is the same effect as a **reject** command). **Lpmove** refuses, however, to move requests while the LP scheduler is running. The general format of the **lpmove** command is as follows:

lpmove *requests dest*

Requests are the request identification numbers (request IDs) of jobs waiting to be printed, and *dest* is the destination to which the requests are to be moved. The destination can be a printer or a class of printers.

8.2.1.6 Command Examples

Example 1 Move all the requests for printer lp1 to printer lp2. Moving the requests renames the request IDs from lp1-nnn to lp2-nnn. After the requests are moved, **lp** will no longer accept requests for lp1 (this is the same effect as a **reject lp1** command issued after the **lpmove**).

```
# lpmove lp1 lp2
#
```

Example 2 Move requests lp1-54 and lp2-55 to printer dqp10_1:

```
# lpmove lp1-54 lp2-55 dqp10_1
total of 2 requests moved to dqp10_1
#
```

NOTE: The two requests are now renamed dqp10_1—54 and dqp10_1—55.

8.2.1.7 /usr/lib/accept

The **accept**(1M) command allows job requests to be placed in a queue at the named destination(s), destination being the name of a printer or class of printers. The general format of the **accept** command is as follows:

accept *destination(s)*

8.2.1.8 Command Example

The sample command line allows printer dqp10_1 to start receiving requests.

```
# /usr/lib/accept dqp10_1
destination "dqp10_1" now accepting requests
#
```

8.2.1.9 /usr/lib/reject

Sometimes it is necessary to stop **lp** from routing requests to a destination. For example, if a printer has been removed for repairs, or if too many requests are building at a destination, you may want to prevent new jobs from being queued at this destination. The **reject**(1M) command performs this function.

Requests in the queue when the **reject** command is invoked will be printed as long as the printer is enabled. After the condition that led to denying requests has been corrected, use the **accept** command to allow requests to be received again. The general format of the **reject** command is as follows:

reject [—r[*reason*]] *destinations*

The —r option enables you to let users know why requests are being rejected by the specified destination. *reason* is a brief explanation of the purpose for rejecting requests. If the reason consists of more than one word, enclose it in double quotes ("). The *destinations* are the printers that are not to accept requests any longer.

8.2.1.10 Command Example

The example given here is for a printer, lqp40_1, that is being repaired. While lqp40_1 is out of service you want to prevent **lp** from routing requests to it.

```
# reject —r"printer lqp40_1 under repair" lqp40_1
destination "lqp40_1" is no longer accepting requests
#
```

Users who try to route a job to lqp40_1 will receive the following message:

```
$ lp —dlqp40_1 filename
lp: can't accept requests for destination "lqp40_1" -
        printer lqp40_1 under repair
$
```

8.3 PRINTER INTERFACE PROGRAMS

Printers that are used as LP Spooling printers must have a printer interface program. Every print request made with the **lp** command is routed through the appropriate printer interface program before the request is printed on a line printer. The printer interface program to use is specified by the **lpadmin**(1M) command.

8.3.1 Model Interface Programs

Each type of printer requires its own interface program. Several, referred to as "model" interface programs, are furnished with the LP Spooling Utilities. The model interface programs support the DQP-10 printer, the LQP-40 printer, and several other popular printers. The model interface programs are written as shell procedures, but they can be written as C programs or any other executable program. They are located in the **/usr/spool/lp/model** directory.

8.3.2 Writing Interface Programs

If you have a printer that is not supported by one of the model programs, you will have to furnish an interface program for it. The shell script for a "dumb" printer interface program (a model program) is shown in Figure 8-1. This program may be used as a guide if you have to provide one of your own.

When the LP scheduler routes an output request to a printer, the interface program for the printer is invoked in the directory **/usr/spool/lp** as follows:

interface/*P id user title copies options file ...*

```
# lp interface for dumb line printer
#
x="XXXX
```

Arguments for the interface program are:

P	printer name
id	request id returned by lp
user	logname of user who made the request
title	optional title specified by the user
copies	number of copies requested by user
options	blank-separated list of class or printer-dependent options specified by user
file	full path name of a file to be printed

When the interface program is invoked, its standard input comes from **/dev/null** and both the standard output and standard error output are directed to the printing device. Interface programs format their output based on the command line arguments. You want to make sure that the interface program has the proper stty modes (terminal characteristics such as baud rate, output options). You can do this by adding **stty**(1) command lines of the form:

stty *mode options* <*&1*

This command line takes the standard input for the **stty** command from the device. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown below.

stty —parenb —parodd 1200 cs8 cread clocal ixon 0<&1

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by **lpsched** as follows:

Code	Meaning to lpsched
0	The print job has completed successfully.
1 to 127	A problem was encountered in printing this particular request (for example, too many nonprintable characters). This problem will not affect future print jobs. The **lpsched** command notifies users by **mail**(1) that there was an error in printing the request.
greater than 127	These codes are reserved for internal use by **lpsched.** Interface programs must not exit with codes in this range.

When problems occur that may affect future print jobs—for example, a device filter program is missing—it is wise to have your interface program disable printers so that print requests are not lost. When an active printer is disabled, the interface program can be halted with signal 15 (see **kill**(1) and **signal**(2)).

8.4 FILES AND DIRECTORIES

This section describes the files and directories in the LP Spooling structure.

8.4.1 /usr/spool/lp/FIF0

FIFO is a special file that all the commands use to send messages to **lpsched.** Any of the LP commands may write to **FIFO,** but only **lpsched** may read it.

8.4.2 /usr/spool/lp/default

This file contains the name of the system default destination. If this file does not exist or if it is empty, the LP system has no default destination.

8.4.3 /usr/spool/lp/log

The purpose of the **log** file is to keep a record of all the printing activity that has taken place since the LP scheduler was last started. This file contains the **logname** of the user who made the request, the request id, the name of the printer that the request was printed on, and the date and time that printing started. Any **lpsched** error messages that occur are also recorded. The first line of the log file shows the time that the LP scheduler was started.

8.4.4 /usr/spool/lp/oldlog

The **oldlog** file contains a record of what was in the **log** file. When the scheduler is stopped, the **log** file is closed. When the scheduler is restarted, all the information that had accumulated in the **log** file is copied to the **oldlog** file, and a new **log** file is started. Any information that had been in the **oldlog** file is overwritten. The first line of the file tells the time that the scheduler was turned on, and the last line tells the time the scheduler was turned off.

8.4.5 /usr/spool/lp/outputq

When an output request is made by the **lp** command, an entry is made in this binary file. The LP scheduler takes the job request and hands it to the appropriate interface program to be printed. After the job is completed, the job request is removed, and the scheduler takes the next job request from this file and has it printed. Only those requests made since the last time the LP scheduler was started are contained in this file (that is, starting the LP scheduler clears this file).

Entries in **outputq** may be modified by the **lpmove, disable,** and **lpsched** command. The **cancel, disable,** and **lpsched** commands can mark entries in this file "deleted." If a job request is deleted before the job is completed, the entry will remain in the file.

8.4.6 /usr/spool/lp/pstatus

The binary file **pstatus** contains status information for each printer. Entries are added and removed from this file by the **lpadmin** command, and they are modified by the **cancel, enable, disable,** and **lpsched** commands. When the **lpstat** command is invoked with the **—p** option, printer status information is obtained from this file.

8.4.7 /usr/spool/lp/qstatus

This binary file keeps track of whether a destination is accepting or rejecting requests. Entries are added or removed from this file by the **lpadmin** command and modified by the **accept** and **reject** commands. When the **lpstat** command is invoked with the —o option, the request status is obtained from this file.

8.4.8 /usr/spool/lp/seqfile

The **seqfile** file contains the sequence number of the last request id that was assigned by the **lp** command. The sequence number is incremented by **lp** for each request. When the number 9999 is reached, the sequence number is reset to 1.

8.4.9 /usr/spool/lp/class

This is a directory that contains one file for each LP class that has been identified. (The name of the file is the same as the name of the class.) The file identifies each member, in this case an LP printer, that is assigned to the class. Class files are created, modified, and deleted by the **lpadmin** command. Every class file must always have at least one member.

8.4.10 /usr/spool/lp/interface

The interface directory contains one executable interface program for each printer that is in the LP system. The filename of the interface program is the same as the printer name. The interface program is invoked with its standard error and standard output directed to the printer. Interface programs may be shell procedures or compiled C programs.

8.4.11 /usr/spool/lp/member

The member directory contains one file for each LP printer. The filename is the same as the printer name. Each file contains the pathname of the device to which the member is connected.

8.4.12 /usr/spool/lp/model

This is a directory that contains the printer interface programs that are distributed with the LP Spooling Utilities.

8.4.13 /usr/spool/lp/request

This directory contains a subdirectory for each destination in the LP system. The name of the subdirectory is the same as the name of the destination. When an **lp** request is made, a **request** file (or **r** file) and, in most cases, a **data** file (or **d** file) are created in the subdirectory of the destination to which the request is going. The **data** file stores the file to be printed until the scheduler is ready to print it. A **data** file is not created if the file to be printed cannot be linked to the request subdirectory.

The name of the request file is derived from the request identification number and is of the form **r-seqno**. The name of the data file is of the form **dn-seqno**, where **n** is a non-negative integer.

The request and data files are deleted by the **cancel** and **lpsched** commands. They may be moved from one subdirectory to another by the **lpmove** command.

8.4.14 Lock Files

To guarantee LP commands exclusive access to data files, several "lock" files are maintained in the LP system. They are binary files that contain the process ID of the locking process. The lock files and their associated data files are:

LOCK FILE	DATA FILE
OUTQLOCK	outputq
PSTATLOCK	pstatus
QSTATLOCK	qstatus
SEQLOCK	seqfile

Lock files "expire" after a given time and may be unlinked by any LP process. Thus, commands that lock a data file for longer than this interval must update the modification time on the lock file. The creation, updating, and unlinking of lock files is handled automatically by the LP low-level file access routines.

Another lock file, SCHEDLOCK, is present while the LP scheduler is running to ensure that only one invocation of **lpsched** is active. Unlike other lock files, SCHEDLOCK has no expiration time.

8.4.15 Cleaning Out Log Files

As described above, when the scheduler is stopped, the **log** file is closed. When the scheduler is restarted, the **log** file is copied to **/usr/spool/lp/oldlog**, and a new **log** file is started.

If the scheduler is not stopped for long periods of time and if you have a large number of LP requests, the **log** file can grow to be a large file. You can manually remove the contents of this file, or you can let the system do it for you on a scheduled basis (see Section 6.5.6).

To have the system clean out the log file, put an entry in a file in the **/usr/spool/cron/crontabs** directory. One way to do this is to log in as **root** and use the **crontab**(1) command. The other way is to edit a **crontabs** directory file.

The example below shows some typical **crontab** command lines. **crontab** adds these command lines to the **root** file in the **/usr/spool/cron/crontabs** directory. Every Friday at 11:00 PM **cron**(1M) executes the commands. First, the contents of the **log** file are copied to the **oldlog** file, and then the **log** file is cleaned out.

```
# crontab —l
0 23 * * 5 /bin/su lp —c "cp /usr/spool/lp/log
/usr/spool/lp/oldlog" 1 23 * * 5 /bin/su lp —c ">/usr/spool/lp/log"
#
```

Chapter 9

TTY MANAGEMENT

9.1 INTRODUCTION

This chapter covers the following topics:

- The terms used in discussing TTY management
- How the TTY system works
- How to tell what line settings are defined
- How to create new line settings and hunt sequences
- How to modify TTY line characteristics
- How to set terminal options

9.1.1 Definition of Terms

The following terms are used in this chapter:

TTY	Derived from the near-classic abbreviation for teletypewriter, the term covers the whole area of access between the GENIX V.3 system and peripheral devices, including the system console. It shows up in commands such as **getty**(1M) and **stty**(1), in the names of device special files such as **/dev/tty01**, and in the names of files such as **/etc/gettydefs**, which is used by **getty**.
TTY line	The physical equipment through which access to the computer is made.
port	A synonym for TTY line.
line settings	A set of line characteristics.
baud rate	The speed at which data is transmitted over the line. A part of line settings.
mode	The characteristics of the terminal interface. A part of line settings. The TTY line and the terminal must be working in the same mode before communication can take place. Described in **termio**(7).
hunt sequence	A circular series of line settings such as different baud rates. During the login sequence, a user looking for a compatible connection to the computer can go from one setting to the next by sending a BREAK signal.
terminal options	Selectable settings that define the way a given terminal operates. Described in **termio**(7).

9.2 THE TTY SYSTEM

The remaining sections in this chapter describe how the TTY system operates, and how you can administer it.

9.2.1 How the TTY System Works

A series of four processes (**init**(1M), **getty**(1M), **login**(1), **sh**(1)) connects a user to the GENIX V.3 system. **Init** is a general process spawner that is invoked as the last step in the boot procedure. It spawns a **getty** process for each line that a user may log in on, guided by instructions in **/etc/inittab.** An argument required by the **getty** command is **line.** The TTY line argument is the name of a special file in the **/dev** directory. For a description of other arguments that may be used with **getty** see the *Administrator's Reference Manual.*

A user attempting to make a connection generates a request-to-send signal that is routed by the hardware to the **getty** process for one of the TTY line files in **/dev.** (We're omitting how the signal gets from the user's terminal to the *Series 32000* Computer.) **getty** responds by sending an entry from file **/etc/gettydefs** down the line. The **gettydefs** entry used depends on the **speed** argument used with the **getty** command. (In the SYNOPSIS of the **getty**(1M) command the argument name is **speed,** but it is really a pointer to the **label** field of a **gettydefs** entry.) If no **speed** argument is provided, **getty** uses the first entry in **gettydefs.** Among the fields in the **gettydefs** entry (described later in this chapter) is the login prompt.

On receiving the login prompt, the user enters a login name. **Getty** starts **login,** using the login name as an argument. **Login** issues the prompt for a password, evaluates the user's response, and assuming the password is acceptable, calls in the user's shell as listed in the **/etc/passwd** entry for the login name. If no shell is named, **/bin/sh** is furnished by default. **Login** also executes **/etc/profile.**

/bin/sh executes the user's **.profile,** if it exists. **.profile** often contains **stty** commands that reset terminal options that differ from the defaults. The connection between the user and the GENIX V.3 system has now been made.

9.2.2 How to Tell What Line Settings Are Defined

You have two ways to check line settings.

1. Through the System Administration Menus, specifically the **sysadm**(1) **lineset** subcommand. **Sysadm lineset** first shows the full range of line settings, then gives you the chance to examine a line in detail (see Procedure 8.1).
2. By looking directly in **/etc/gettydefs.**

The **/etc/gettydefs** file contains information used by the **getty**(1M) command to establish the speed and terminal settings for a line. The general format of the **gettydefs** file is:

label# initial-flags # final-flags #login-prompt #next-label

Figure 9-1 shows a few lines from a **gettydefs** file.

The entries shown in Figure 9-1 form a single, circular hunt sequence; the last field on each line is the label of the next line. The next-label field for the last line shown points back to the first

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600
9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800
4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400
2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200
1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300
300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
```

Figure 9-1. Gettydefs Entries

line in the sequence. The object of the hunt sequence is to link a range of line speeds. If you see garbage characters instead of a clear login prompt, entering a BREAK causes **getty** to step to the next entry in the sequence. The hunt continues until the baud rate of the line matches the speed of the user's terminal. The flag fields shown have the following meanings:

B300-B19200	The baud rate of the line.
HUPCL	Hang up on close.
SANE	A composite flag that stands for a set of normal line characteristics.
IXANY	Allow any character to restart output. If this flag is not specified, only DC1 (CTL-Q) will restart output.
TAB3	Send tabs to the terminal as spaces.

For a description of all **getty** flags, see **termio**(7).

9.2.3 How to Create New Line Settings and Hunt Sequences

You have two ways to do this.

1. Use the System Administration Menus, specifically the **sysadm mklineset**(1) subcommand. **Sysadm mklineset** leads you through a series of prompts. Your responses make up the information for a new **gettydefs** entry (see Procedure 8.2).
2. By using **ed**(1) or **vi**(1) to edit **/etc/gettydefs**.

Create new lines for the **gettydefs** file by following the example shown above. Each entry in the file is followed by a blank line. After editing the file run the command:

 # /etc/getty —c /etc/gettydefs

This causes **getty** to scan the file and print the results on your terminal. If there are any unrecognized modes or improperly constructed entries, they are reported.

9.2.4 How to Modify TTY Line Characteristics

You have two ways to do modify TTY line characteristics.

1. Use the System Administration Menus, specifically the **sysadm modtty**(1) subcommand. **Sysadm modtty** leads you through a series of prompts. Your responses edit a "getty" entry in **/etc/inittab** (see Procedure 8.3).
2. By using **ed**(1) or **vi**(1) to edit **/etc/inittab.**

The **/etc/inittab** file contains instructions for the **/etc/init**(1M) command. The general format of a line entry in the **/etc/inittab** file is as follows.

identification:level:action:process

The four colon-separated fields are as follows.

identification	A unique one- or two-character identifier for the line entry.
level	The run-level in which the entry is to be performed.
action	How **/etc/init** treats the process field (refer to the **inittab**(4) manual page for complete information).
process	The shell command to be executed.

/etc/inittab contains several entries that spawn **getty** processes. Figure 9-2 is a selection of such entries **grep**'ped from an **/etc/inittab** on a *Series 32000* Computer.

There are at least three things you might want to do to an **inittab** entry for a TTY line:

1. Change the action. Two actions that apply to TTY lines are "respawn" and "off" (see the **inittab**(4) manual page for complete information on this field).
2. Add or change arguments to **/etc/getty** in the process field. A frequently used argument is **—t***nn*. This tells **getty** to hang up if nothing is received within *nn* seconds. It's good practice to use the **—t** argument on dial-up lines.
3. Add or change comments. Comments can be inserted after a semi-colon (**;**) to end the command, and a pound sign (**#**) to start the comments.

```
co:1234:respawn:/etc/getty console console
ct:2:off:/etc/getty contty contty;#line not in use
21:2:respawn:/etc/getty tty21 9600
22:2:respawn:/etc/getty tty22 9600
23:2:respawn:/etc/getty tty23 9600
24:2:off:/etc/getty tty24 9600;#line not in use
25:2:off:/etc/getty tty25 9600;#line not in use
```

Figure 9-2. Getty Entries from /etc/inittab

9.2.5 How to Set Terminal Options

The TTY system described thus far establishes a basic style of communication between the user's terminal and the GENIX V.3 operating system. Once the user has successfully logged in, there may be terminal options that would be preferable to ones in the default set.

The command that is used to control terminal options is **stty**(1). Many users add an **stty** command to their **.profile** so the options they want are automatically set as part of the **login** process. Here is an example of a simple **stty** command.

```
$ stty cr0 nl0 echoe —tabs erase ^H
```

The options in the example mean:

cr0 nl0	No delay for carriage return or new line. Delays are not used on a video display terminal, but are necessary on some printing terminals to allow time for the mechanical parts of the equipment to move.
echoe	Erases characters as you backspace.
—tabs	Expand tabs to spaces when printing.
erase ^H	Change the character-delete character to a ^H. The default character-delete character is the pound sign (#). Most terminals transmit a ^H when the **backspace** key is pressed. Specifying this option makes the **backspace** key useful.

Chapter 10

BASIC NETWORKING

10.1 INTRODUCTION

The Basic Networking Utilities let computers using the GENIX V.3 operating system communicate with each other, other UNIX oeprating systems, and with remote terminals. These utilities range from those used to copy files between computers (**uucp** and **uuto**) to those used for remote login and command execution (**cu**, **ct**, and **uux**).

As an administrator, you need to be familiar with the administrative tools, logs, and data base files used by the Basic Networking Utilities. Procedure 9, Basic Networking, presents instructions to install and maintain these utilities; this chapter goes into greater detail about the Basic Networking Utilities files, directories, daemons, and commands.

10.2 NETWORKING 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 local area networks.

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), your computer can communicate with other computers over standard phone lines. The ACU dials the telephone number requested by the networking utilities. The remote computer must have a telephone modem capable of answering incoming calls.
Local Area Network	A Local Area Network (LAN) can be the communication medium for basic networking. Once your computer is established as a node on a LAN, it will be able to contact any other computer connected to the LAN.

10.3 NETWORKING COMMANDS

Basic networking programs can be divided into two categories: user programs and administrative programs. The following paragraphs describe the programs in each category.

10.3.1 User Programs

The user programs for basic networking are in **/usr/bin.** No special permission is needed to use these programs. These commands are all described in the *User's 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.

ct Connects your computer to a remote terminal so the user of the remote terminal can log in. The user of a remote terminal can call the computer and request that the computer call it back. In this case, the computer drops the initial link so that the remote terminal's modem will be available when it is called back.

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 (**/usr/spool/uucppublic/receive**). Unlike **uucp**, which lets you copy a file to any accessible directory on the remote computer, **uuto** places the file in an appropriate spool directory and tells the remote user to pick it up with **uupick.**

uupick Retrieves the files placed under **/usr/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.

10.3.2 Administrative Programs

Most of the administrative programs are in **/usr/lib/uucp,** along with basic networking data base files and shell scripts. The only exception is **uulog,** which is in **/usr/bin.** These commands are described in the *Administrator's Reference Manual.*

You should use the **uucp** login ID when you administer the Basic Networking Utilities because it owns the basic networking and spooled data files. The home directory of the **uucp** login ID is **/usr/lib/uucp.** (The other basic networking login ID is **nuucp,** used by remote computers to access your computer. Calls from **nuucp** are answered by **uucico.**)

uulog Displays the contents of a specified computer's log files. Log files are created for each remote computer with which your computer communicates. 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.**

Uutry 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.

uucheck Checks for the presence of basic networking directories, programs, and support files. It can also check certain parts of the **Permissions** file for obvious syntactic errors.

10.4 DAEMONS

There are three daemons in the Basic Networking Utilities. A daemon is a routine 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 **Uutry** 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 executed by a shell script called **uudemon.hour**, which is started by **cron.**

10.4.1 Internal Programs

uugetty This program is very similar to the **getty** program except it permits a line (port) to be usable for concurrent dialin and dialout operation. A **uugetty** will be assigned to a port in the **/etc/inittab** file if bidirectional is chosen when you modify a port using the **sysadm**(1) **portmgmt** command. **Uugetty** is executed as a function of the **init** program and is described in the *Administrator's Reference Manual.*

10.5 SUPPORTING DATA BASE

The Basic Networking Utilities support files are in the **/usr/lib/uucp** directory. Most changes to these files can be made using the System Administration Menu commands described in Procedure 9, Basic Networking. The descriptions below, however, 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 network devices (automatic calling devices, including non 801-type dialers) 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.

Devconfig This file is used to configure utilities for the Basic Networking Utilities on an NPACK NETWORK or some other transport provider that conforms to the Transport Interface.

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 briefly in Procedure 9, Basic Networking.

10.5.1 Devices File

The **Devices** file (**/usr/lib/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.

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.

Each entry in the **Devices** file has the following format:

Type Line Line2 Class Dialer-Token-Pairs

Each of these fields is defined in the following section.

Type This field may contain one of two keywords (**Direct** or **ACU**), the name of a Local Area Network switch, or a system name.

Direct This keyword indicates a Direct Link to another computer or a switch (for **cu** connections only).

ACU This keyword indicates that the link to a remote computer is made through an automatic call unit (Automatic Dial Modem). This modem may be connected either directly to your computer or indirectly through a Local Area Network (LAN) switch.

LAN_Switch
This value can be replaced by the name of a LAN switch. **micom** and **develcon** are the only ones for which there are caller scripts in the **Dialers** file. You can add your own LAN switch entries to the **Dialers** file. If you are adding a Transport Interface-compatible network, such as NPACK NETWORK, you would use the special dialer types **TLI** or **TLIS.**

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:

Devices: ACU tty11 - 1200 penril

**Systems: eagle Any ACU 1200 3251 ogin: nuucp **
ssword: Oakgrass

You can designate a protocol to use for a device within this field. See Section 9.5.1.1.

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/tty11** line, the name entered in this field would be **tty11.**

Line2 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 ACUs do not contain a modem. Therefore, a separate modem is required and would 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 will not normally use this configuration, the *Line2* field will be ignored by them, but it must still contain a hyphen (-) as a placeholder.

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 (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:

Devices: ACU tty11 - D1200 penril

**Systems: eagle Any ACU D1200 3251 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 **Systems** file entry. If this field is **Any** and the **Systems** file *Class* field is **Any**, the speed defaults to 1200 baud.

Dialer-Token-Pairs:
This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a LAN switch, 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 one of several special dialer types. These special dialer types are compiled into the software and are therefore available without having entries in the **Dialers** file.

801 - Bell 801 auto dialer
TLI - Transport Level Interface Network (without STREAMS)
TLIS - Transport Level Interface Network (with STREAMS)

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:

   Devices: ACU tty11 - 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. (\T 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 local network switch as your computer, your computer must first access the switch and the switch can make the connection to the other computer. In this type of entry, there is only one pair. The *dialer* portion is used to match a **Dialers** file entry as shown below:

   Devices: develcon tty13 - 1200 develcon \D

   Dialers: develcon "" "" \pr\ps\c est:\007 \E\D\e \007

As shown, the *token* portion is left blank, 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:

   Devices: ACU tty14 - 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 LAN switch 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).

10.5.1.1 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 specify it in the form *Type,Protocol* (for example, **NPACK,e**). 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 error-free transmission.

For reliable local area networks, you should use the **e** protocol. Here is an example of adding a protocol designation to a device entry:

NPACK,e npack - - TLIS \D

This says, for device **Npack**, use **e** protocol.

10.5.2 Dialers File

The **Dialers** file (**/usr/lib/uucp/Dialers**) specifies the initial conversation that must take place on a line before it can be made available for transferring data. 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 (such as the Automatic Dial Modem).

As shown in the above examples, the fifth field in a **Devices** file entry is an index into the **Dialers** file or a special dialer type (801, TLI, or TLIS). 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:

dialer substitutions expect-send ...

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 character strings distributed with the Basic Networking Utilities in the **Dialers** file.

```
penril =W-P "" \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!
hayes  =,-, "" \dAT\r\c OK\r \EATDT\T\r\c CONNECT
rixon  =&-% "" \d\r\r\c $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadiac =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
direct
att2212c       =+-,    "" \r\c :--: ato12=y,T\T\r\c red
att4000 =,-,   "" \033\r\r\c DEM: \033s0401\c \006 \033s0901\c \
       \006 \033s1001\c \006 \033s1102\c \006 \033dT\T\r\c \006
att2224 =+-,   "" \r\c :--: T\T\r\c red
nls    ""      "" NLPS:000:001:1\N\c
```

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. First, the phone number argument is translated, replacing any = with a **W** (wait for dialtone) and replacing 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 item.)
\d	Delay for 2 seconds.
>	Wait for a >.
s\p9\c	Send an **s**, pause for ½ second, send a **9**, send no terminating new-line
)-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 new-line, and then wait for the).
y\c	Send a **y**.
:	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**.
>	Wait for a >.
9\c	Send a **9** without a new-line.
OK	Waiting for the string **OK**.

10.5.3 Systems File

The **Systems** file (**/usr/lib/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 basic networking software can be configured to prevent any computer that does not appear in this file from logging in on your computer (refer to Section 10.5.9 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. The management of this file is supported by the System Administration Menu subcommand **systemmgmt**.

Using the **Sysfiles**, you can define several files to be used as "Systems" files. See the description of the **Sysfiles** file, Section 10.5.8, for details. Each entry in the **Systems** file has the following format:

System-Name Time Type Class Phone Login

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 Th Fr Sa
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).

Here is an example:

Wk 1700-0800, Sa, Su

This example allows calls from 5:00 p.m. to 8:00 am, Monday through Thursday, 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. An optional subfield, *retry*, is available to specify the minimum time (in minutes) before a retry, following a failed attempt. The default wait is 60 minutes. 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.

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:

```
Systems: eagle Any ACU,g D1200 3251 ogin: nuucp \
         ssword: Oakgrass

Devices: ACU tty11 - D1200 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 in Section 10.5.1.1 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:

**Systems: eagle Any ACU D1200 NY3251 ogin: nuucp **
ssword: Oakgrass

Devices: ACU tty11 - D1200 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 (LAN switches). The phone number is made up of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the **Dialcodes** file. For example:

**Systems: eagle Any ACU D1200 NY3251 ogin: nuucp **
ssword: Oakgrass

Dialcodes: NY 9=1212555

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 LAN switch, you may access other computers that are connected to that switch. The **Systems** file entries for these computers will not have a phone number in the *Phone* field. Instead, this field 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 **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 new line, 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 new-line 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: 6
uucpx word: xyzzy

This example says send a carriage return and wait for **ogin:** (for **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:), send the password **xyzzy**.

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 NULL).
\b	Send or expect a backspace character.
\c	If at the end of a string, suppress the new-line that is normally sent. Ignored otherwise.
\d	Delay two seconds before sending or reading more characters.
\p	Pause for approximately ¼ to ½ 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 or expect EOT new-line twice.
BREAK	Send or expect a break character.
\K	Same as BREAK.
\ddd	Collapse the octal digits (ddd) into a single character.

10.5.4 Dialcodes File

The **Dialcodes** file (**/usr/lib/uucp/Dialcodes**) contains the dial-code abbreviations that can be used in the *Phone* field of the **Systems** file. Each entry has the format:

abb dial-seq

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.**

10.5.5 Permissions File

The **Permissions** file (**/usr/lib/uucp/Permissions**) specifies the permissions that remote computers have with respect to login, file access, and command execution. There are options that restrict the remote computer's ability to request files and its ability to receive files queued by the local site. Another option is available that specifies the commands that a remote site can execute on the local computer.

10.5.5.1 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

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.

MACHINE Specifies permissions that take effect when your computer logs in on (calls) a remote computer.

LOGNAME entries will contain a LOGNAME option and MACHINE entries will contain a MACHINE option.

10.5.5.2 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:

    1. Local send and receive requests will be executed.
    2. The remote computer can send files to your computer's **/usr/spool/uucppublic** directory.
    3. The commands sent by the remote computer for execution on your computer must be one of the default commands; usually **rmail**.

10.5.5.3 Options

This section describes each option, specifies how they are used, and lists their default values.

REQUEST 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. A note on security: 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.

SENDFILES 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.

READ and WRITE

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=/usr/spool/uucppublic WRITE=/usr/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=/usr/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

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings

READ=/ NOREAD=/etc WRITE=/usr/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 **/usr/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 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 COMMAND option is

CALLBACK=no

The CALLBACK option is very rarely used. Note that if two sites have this option set for each other, a conversation will never get started.

COMMANDS 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 a command 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 that COMMANDS is not used in a LOGNAME entry; COMMANDS in MACHINE entries define command permissions whether we call the remote system or it calls us.

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/lbin/rnews:/usr/local/lp

specifies that command **rmail** uses the default path. The default paths for your computer are **/bin, /usr/bin,** and **/usr/lbin.** When the remote computer specifies **rnews** or **/usr/lbin/rnews** for the command to be executed, **/usr/lbin/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.

The string

COMMANDS=/usr/lbin/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 The VALIDATE option is used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your computer's security. It is used to provide a certain degree of verification of the caller's identity. 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. (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/lbin/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/lbin/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 Systems

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/lbin/Photo:/usr/lbin/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 Entries

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=/
```

10.5.6 Poll File

The **Poll** file (**/usr/lib/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 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.**

10.5.7 Devconfig File

The **/usr/lib/uucp/Devconfig** file is used when your computer communicates over an NPACK network or some other Streams-based transport provider that conforms to the Transport Interface (TI).

Devconfig entries define the STREAMS modules that are used for a particular TI device. Entries in the **Devconfig** file have the format:

service=*x* **device=***y* **push=***z*[:*z* ...]

where *x* can be **cu, uucico,** or both separated by a colon; *y* is the name of a TI network and must match an entry in the **Devices** file; and *z* is replaced by the names of STREAMS modules in the order that they are to be pushed onto the Stream. Different modules and devices can be defined for **cu** and **uucp** services.

The following entries should most commonly be used in the file:

```
service=cu      device=NPACK    push=ntty:tirdwr:ld0
service=uucico  device=NPACK    push=ntty:tirdwr:ld0
```

This example pushes **ntty,** then **tirdwr,** then **ld0.** The **Devconfig** file cannot be modified with the System Administration Menus command **sysadm.** If you want to change the contents of this file, you must use one of the GENIX V.3 system text editors.

10.5.8 Sysfiles File

The **/usr/lib/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 **/usr/lib/uucp** directory, unless a full path is given. A backslash-carriage return (\<**CR**>) 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 **/usr/lib/uucp/Sysfiles**, then both **uucico** and **cu** will first look in **/usr/lib/uucp/Systems**. If the system they're trying to call doesn't have an entry in that file, or if the entries in the file fail, then they'll look in **/usr/lib/uucp/Local_Systems**.

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 **uuname** command or the **cu** list using the **uuname** —**c** command.

10.5.9 Other Networking Files

There are three other files that impact the use of basic networking facilities. In most cases, the default values are fine and no changes are needed. If you want to change them, however, use any standard GENIX V.3 system text editor (**ed** or **vi**).

Maxuuxqts This file defines the maximum number of **uuxqt** programs that can run at once.

Maxuuscheds This file defines the maximum number of **uusched** programs that can run at once.

remote.unknown This file is a shell script that 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.

10.6 ADMINISTRATIVE FILES

The basic networking 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 Basic Networking processes under the spool directory (*i.e.*, **/usr/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.***sysnxxxx* 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 **/usr/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.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work, and *xxxx* is the four digit job sequence number assigned by **uucp.** Work files contain the following information:

- Full pathname of the file to be sent or requested
- Full pathname of the destination or user filename
- User login name
- List of options
- Name of associated data file in the spool directory. If the **uucp —c** or **uuto —p** option was specified, a dummy name (**D.0**) is used
- Mode bits of the source file
- Remote user's login name to be notified upon completion of the transfer

D. (Data file)

Data files are created when it is specified in the command line to copy the source file to the spool directory. The names of data files have the following format:

D.*systmxxxxyyy*

where *systm* is the first five characters in the name of the remote computer, *xxxx* is a four-digit job sequence number assigned by **uucp.** The four digit job sequence number may be followed by a sub-sequence number, *yyy* that is used when there are several **D.** files created for a work (**C.**) file.

X. (Execute file) Execute files are created in the spool directory prior to remote command executions. The names of execute files have the following format:

X.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is a four digit sequence number assigned by **uucp**. Execute files contain the following information:

- Requester's login and computer name.
- Name of file(s) required for execution.
- Input to be used as the standard input to the command string.
- Computer and file name to receive standard output from the command execution.
- Command string.
- Option lines for return status requests.

10.7 DIRECT LINKS

10.7.1 General

Direct links are beneficial only when:

- It is not possible to link the computers together through a Local Area Network (LAN).
- Two computers transfer large amounts of data on a regular basis.
- Two computers are located no more than several hundred cable feet apart.

The distance between two directly linked computers is dependent on the environment in which the cable is run. The standard for RS-232 connections is 50 feet or less with transmission rates as high as 19200 bits per second (bps). As the cable length is increased, noise on the lines may become a problem, which means that the transmission rate must be decreased or limited distance modems be placed on each end of the line.

Do not use more than 1000 cable feet to connect the two computers or communications will be unreliable. This link should operate comfortably at 9600 bps in a clean (noise free) environment.

10.7.2 BNU Software and Direct Links

Ideally, systems that have a direct link should be running the same release of the GENIX V.3 system to have the full set of capabilities available. However, lack of a common version of the GENIX V.3 operating system will not prevent you from using the Basic Networking Utilities.

This section describes the software files that must be modified on your *Series 32000*-based system in order to accommodate a direct link connection.

The following support files must be updated to reflect the presence of a Direct Link:

- **/usr/lib/uucp/Devices**
- **/etc/inittab**
- **/usr/lib/uucp/Systems.**

All necessary additions/modifications to these three files can be accomplished using the System Administration Menus subcommand **uucpmgmt.**

10.7.2.1 Making Devices File Entries

The **Devices** file contains the information concerning the location (line) and transmission rate of the link. Entries can be added to the **Devices** file using the System Administration Menus subcommand **uucpmgmt devicemgmt** and selecting the **add** operation. You will be prompted for the following information:

- port name (for /dev/tty21 use **tty21**)
- device type to call on (**direct**)
- speed at which you want to call (**9600** or **19200**).

To access the System Administration Menus subcommand **devicemgmt,** enter:

 # sysadm devicemgmt<CR>

You should see the following output on your screen:

```
Running subcommand 'devicemgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

        Note: After a brief introduction to the devicemgmt
        subcommand, the procedure interactively prompts you for
        the information listed above.
```

10.7.2.2 Making Changes to the /etc/inittab File

There are two versions of the Basic Networking Utilities. The differences between them are reflected in the **/etc/inittab** file. The newest version allows for bidirectional login capability by respawning **uugetty** instead of **getty.** This means that if two computers (both using **uugetty**) were connected via a direct link, either of these computers could request communication with the other. This would not be true if only one computer was capable of respawning **uugetty.**

If the direct link is connecting your *Series 32000*-based system with a computer that has the new version of basic networking, the **/etc/inittab** files on both computers should be set up to allow "bidirectional" traffic on the associated lines. This means that the lines used must respawn **uugetty** on each end of the link. This would allow either computer to request communication with (call) the other.

If the direct link is connecting your *Series 32000*-based system with a computer that does not have the new version of basic networking, the **/etc/inittab** file would be set up differently on each system. The **/etc/inittab** file on each computer would be set up to allow either "incoming" or "outgoing" traffic on its line. If one computer allows incoming traffic, the other must allow only outgoing traffic. A **uugetty** could not be used on either computer in this case.

A computer's **/etc/inittab** entry would be respawning **getty** for incoming traffic, or have respawn turned off for outgoing traffic. In order for this type of link to work, one of the computers must be set up to poll the other. If the remote computer is allowing only incoming traffic, you must set up your *Series 32000*-based system to poll the remote computer. (You can use the System Administration Menus subcommand **uucpmgmt pollmgmt** to do this). If the remote computer is allowing only outgoing traffic on the link, it must be set up to poll your *Series 32000* Computer.

Entries in the **/etc/inittab** file can be changed using the System Administration Menus subcommand **uucpmgmt portmgmt,** and selecting the **modify** operation. The **modify** operation will prompt you for the following information:

- port name you want to modify (for /dev/tty21 use **tty21**)
- direction of traffic on port (**bidirectional, incoming,** or **outgoing**)
- transmission speed of the link (**9600** or **19200**).

The modify procedure will display the ports that are currently dedicated for use by **uucp** (listed in **Devices** file). The port name to be modified must be one that is listed.

To access the **portmgmt** subcommand and modify entries in the **/etc/inittab** file, enter:

 # sysadm portmgmt <CR>

You should see the following output on your terminal screen:

Running subcommand 'portmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

Note: After a brief introduction to the portmgmt subcommand, the procedure interactively prompts you for the information listed above.

10.7.2.3 Making Systems File Entries

An entry must be made into the **Systems** file for the computer associated with the direct link. This can be done using the System Administration Menus subcommand **systemmgmt** and selecting the add operation. The add operation will prompt you for the following information:

- node name of system
- type of device to call on (**direct**)
- transmission speed of link (**9600** or **19200**)
- device port used with link (for /dev/tty21 use **tty21**)
- login ID used to login on system (**nuucp** or some other login you set up)
- password used by above login

To prevent possible problems logging on when operating at high speeds, you can insert pauses (**\p**) between the characters being sent out for the login ID and the password. For instance, instead of **nuucp**, you should enter **n\pu\pu\pc\p\pp** when prompted for the login ID. Do **not** select the default by pressing RETURN. The same applies for the password assigned.

To access the **systemmgmt** subcommand and add entries in the **Systems** file, enter:

 # sysadm systemmgmt<CR>

You should see the following output on your terminal screen:

Running subcommand 'systemmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

Note: After a brief introduction to the systemmgmt subcommand, the procedure interactively prompts you for the information listed above.

Upon completion of the **add** operation, a new entry is added to the **Systems** file for the remote computer. You must also make an additional entry in the **Devices** file. When you use **devicemgmt** to create an entry for the link, it creates an entry similar to the one shown below.

Direct tty21 - 9600 direct

Chapter 11

REMOTE FILE SHARING

11.1 OVERVIEW

Remote File Sharing (RFS) allows computers running GENIX V.3 to selectively share resources (files, directories, devices, and named pipes) across a network. As an administrator for a host computer on an RFS network, you can choose directories on your system you want to share and add them to a list of available resources on the network. From this list, you can choose resources from remote hosts that you would like to use on your computer.

Each host computer on a Remote File Sharing system can be grouped with others in a domain or can operate as an independent domain. The domain can provide a central point for administering a group of hosts.

Unlike other distributed file systems used with the GENIX V.3 operating system, Remote File Sharing is built into the operating system itself. This approach has several advantages:

Compatibility	Once you mount a remote resource on your system it will look to your users as though it is part of the local system. You will be able to use most standard GENIX V.3 system features on the resource. Standard commands and system calls, as well as features like File and Record Locking, work the same on remote resources as they do locally. Applications should be able to work on remote resources without modification.
Security	Standard GENIX V.3 system file security measures will be available to protect your resources. Also, special means for verifying host access to your resources and restricting remote users' permissions have been added for Remote File Sharing.
Flexibility	Because you can mount a remote resource on any directory on your system, you have a lot of freedom to set up your computer's view of the world. You don't have to open up all your files to every host on the network. Likewise, you don't have to make all files on the network available to your computer's users.

11.1.1 Resource Sharing

Sharing a resource on a Remote File Sharing system begins with a path name to a GENIX V.3 system directory. If there is a directory you want to share, you assign it a resource name and "advertise" it to other hosts, using the **adv**(1M) command. The resource name is how other hosts reference that directory. Hosts that pass the security checks you have set up can then mount your resource as they would mount a file system locally.

Figure 11-1 shows how two hosts can share resources. In this example, the administrator of a host named **fie** on a Remote File Sharing system wants to share all files and directories under **/fsf** on its file system tree. The administrator advertises **/fsf** as a resource called **FSLOGS.**

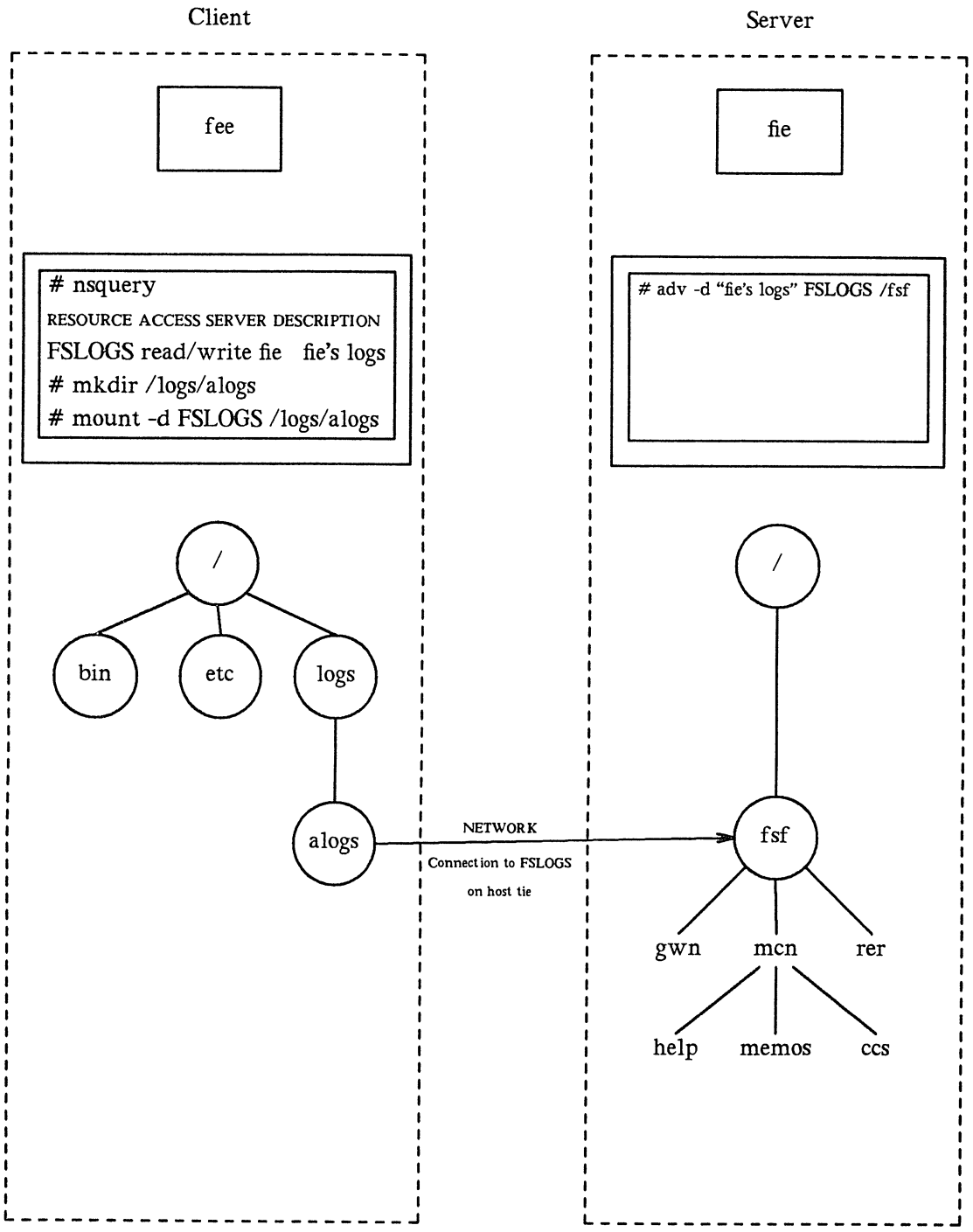


Figure 11-1. Example - Sharing Resources

Another machine in **fie**'s domain is called **fee**. The administrator from **fee** uses the **nsquery** command to see that **FSLOGS** is available on **fie**. **Fee**'s administrator then creates a directory called **/logs/alogs** on **fee** (**mkdir** command) and **mounts FSLOGS** on **/logs/alogs**.

Files or subdirectories from **/fsf** are now accessible to users on **fee**. Users can **cd** to the remote directory, list the contents, and run a remote program locally. If the resource contained the **/dev** directory, users could direct output to a remote device as though the device were on the local host.

11.1.2 Domains

A "domain" in Remote File Sharing provides a means of logically grouping hosts for administrative and addressing purposes. Designated domain name servers for a domain keep track of available resources and password information for the hosts they serve.

Every host in a Remote File Sharing system must belong to a domain. The size of a domain can range from one host to as many as will fit on the network. When you set up a domain, you define a primary and zero or more secondary domain name servers. If the primary fails, domain name service functions are picked up by one of the secondary domain name servers. A domain provides the following services to each of its hosts:

domain name server
: The domain name server keeps track of all available hosts and resources in the domain. It provides transparent access to resources and ensures unique names within a domain. Host passwords, host user lists, and network addresses of other domains are also kept by the domain name server.

reliable service
: Primary and secondary domain name servers ensure that failure of one host does not bring down a Remote File Sharing system. Even if the primary fails, the secondary will take over promptly. If a host that is not a name server fails, that host's resources are no longer available, but all other mounts on the network remain intact.

central point of access
: As Remote File Sharing networks grow, grouping hosts into domains simplifies communications by providing a central point (the primary domain name server) through which several hosts can be located. Once your domain knows another domain name server's address, you have the potential to access resources from any host in that domain.

  With this central point for information, a host administrator doesn't need to check with every host to find what it needs to share resources. All information needed to locate resources, set up security, and communicate with other hosts is contained on the host's domain name server.

11.1.3 Administration

Remote File Sharing administration should be viewed on two levels—host and domain. As a host administrator you need to do the following.

1. Start up Remote File Sharing. This connects you to the Remote File Sharing service on the network.
2. Choose what resources you want to share and who you want to share them with. When you advertise resources they are listed with your domain name server.
3. Choose the resources you want to use from other hosts and mount them on your local file system.
4. Optional: Define how group IDs and user IDs from remote hosts are mapped into your host. You can also provide lists of your users and groups so they can be mapped into other hosts.

All domain administration is done from the primary domain name server. These activities include:

- defining the names and network addresses of the primary and secondary domain name servers, including those of name servers for any other accessible domains
- maintaining a list of all hosts in the domain
- (optional) maintaining password and user mapping information for all accessible hosts within the domain

If the primary goes down, a secondary will take over as domain name server. However, an administrator on a secondary cannot add or remove hosts or change host passwords. You can only pass name server responsibility back to the primary when it comes up.

A primary name server can serve several domains. However, the secondaries must be the same for all domains the primary serves. This simplifies the process of passing name server responsibilities back to the primary after it has gone down and come back up.

The tools you use for Remote File Sharing extend, rather than change, the standard GENIX V.3 system interface. You can use the same commands and system calls on remote resources as you did on local resources. However, some new options and commands have been added to address special Remote File Sharing needs.

For example, the **sar —u** command still produces system activity reports for your computer, but **sar —Du** adds Remote File Sharing records to the report. Likewise, your GENIX V.3 system file structure looks the same, but it is extended by mounting remote resources (**mount —d** command).

11.1.4 Security

As a host administrator, you can maintain strict control of your resources. No files, directories, or devices in an unshared file system can be accessed by other hosts.

Direct access to your computer is controlled because local users still have to login as they always have. As for remote accessibility, you can set up security to allow only certain remote hosts to access your resources.

Figure 11-2 illustrates the security checks that occur when a host tries to mount a remote resource, including the user mapping that occurs. The following sections explain some of these issues.

11.1.5 Connect Security

When a remote host tries to mount a resource from your host, and no other resources are mounted, it tries to set up a connection (virtual circuit) across the network to your machine. Once this virtual circuit is set up, the host can mount any resource you have made available to it. This virtual circuit is closed when the last resource is unmounted.

Before this virtual circuit is created, the host must pass the security you set up.

- any host can connect

  You could indicate that any host that tries to connect to your host may do so (see the **rfstart** command).

- some hosts can connect

  Only those hosts for which your host has a matching name and password can connect (see the **—v** option to the **rfstart** command). This information is in the domain password file(s) you can maintain. You can tailor each of these files if you only want a subset of hosts in each domain to be allowed to connect.

11.1.5.1 Mount Security

Once a remote host has established a connection to your host, the resources it can mount from your host depend on how you advertised each resource. These are your choices:

- any host can mount

  You may have advertised the resource so that any host that can connect to your host can mount it.

- some hosts can mount

  You restricted access to the resource to certain hosts; the remote host trying to mount it must be one of those hosts (see the *clients...* option of the **adv** command).

You also may have advertised the resource as read-only (see **adv —r**). In that case, the remote host can only mount the resource read-only instead of read/write (default).

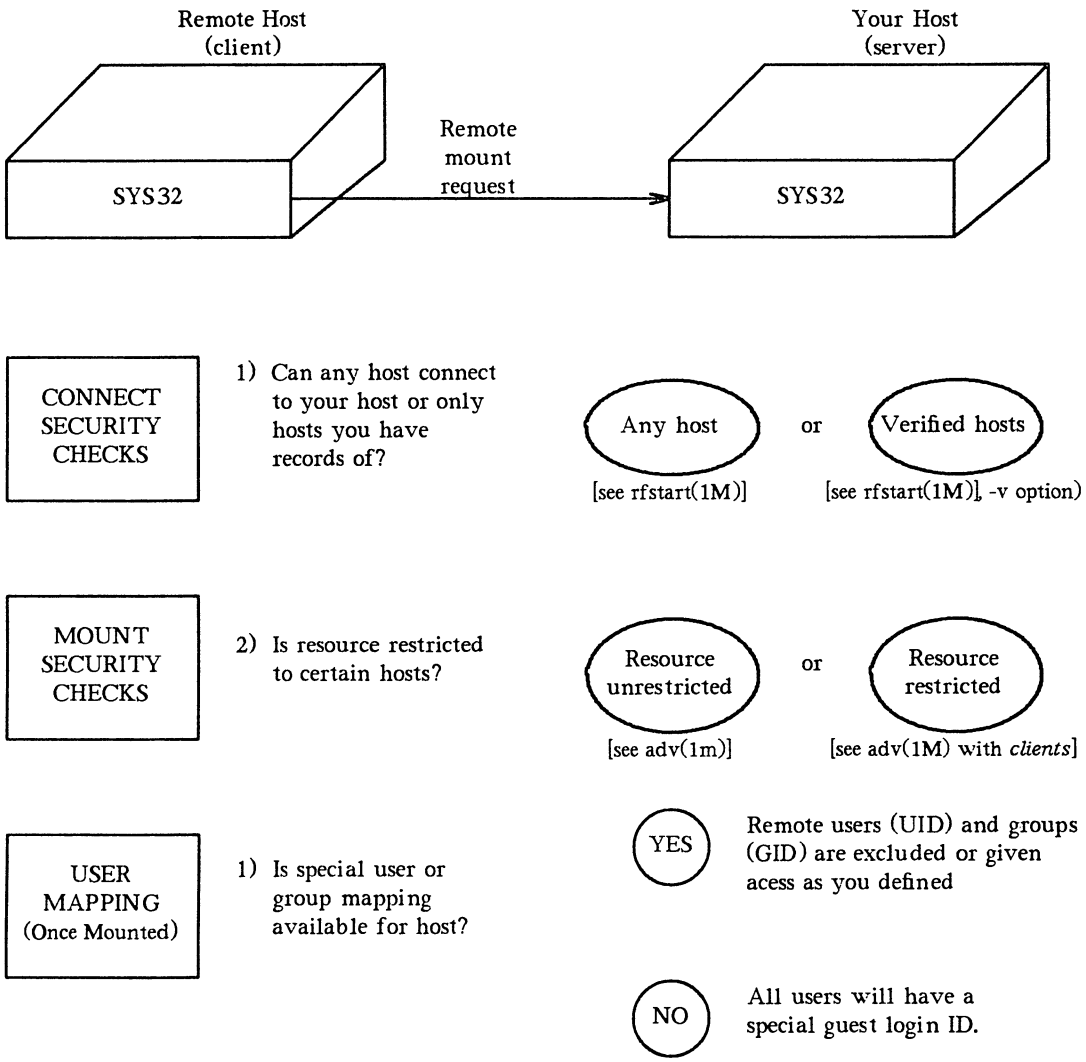


Figure 11-2. Example - Remote File Sharing Security

11.1.5.2 User and Group Security

Remote users' permissions can be defined to provide another layer of security for a mounted resource. Remote users and groups can be mapped into your host's user and group list to set permissions they will have to your resources.

You can set these mapping rules on a global or host basis. The global rules set user and group permissions for all hosts that do not have explicit host mapping rules.

Here are the ways you can map remote hosts' users into your host. These rules apply to both **global** and **host** mapping.

- no mapping

  If you don't set any special mapping for any remote host, all users are mapped into your host 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

  You can set default mapping so that all remote users are mapped into one of these permissions:

  — the local user ID number that matches each remote user's ID (default transparent)

  — a single local ID number

  — a single local ID name

  — the local user name that matches each remote user's name (**map all**)

  Group permissions can be mapped 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 they only have special guest permissions (for example, **exclude 0**).

- specific mapping

  You can map any user or group from any remote machine into a specific user or group on your machine. This can be done by user name or numeric ID. (You can't use remote names in global blocks.)

Using these mapping techniques and standard methods for setting file permissions, you can keep strict controls over your resources, even after they are remotely mounted. (See Section 11.3 more details.)

11.2 SHARING RESOURCES

This section describes how to share your resources with other hosts on a Remote File Sharing system and how to use the resources other hosts have made available. To share resources, you will use the following commands:

adv(1M)	(with options) Makes your resources available to other hosts.
mount(1M)	(with options) Attaches a remote resource to your host.

To print information on shared resources, you will use the following commands:

adv(1M)	(no options) Lists your advertised resources.
nsquery(1M)	Lists other hosts' advertised resources.
mount(1M)	(no options) Lists the remote resources, as well as local file systems you have mounted.
rmntstat(1M)	Lists your resources that other hosts have mounted.
fusage(1M)	Reports the extent of disk usage by remote hosts. (See Section 11.4.6.)
df(1M)	Reports disk space availability on remote file systems as well as localfile systems that you have mounted. (See Section 11.4.6.)

To remove remote resources, you will use these commands:

unadv(1M)	Unadvertises one of your advertised resources.
umount(1M)	Unmounts a remote resource you mounted on your host.
fuser(1M)	Checks what local users are currently accessing a remote resource. (You can use this before you unmount the resource.)
fumount(1M)	Forces an unmount of one of your resources from all clients that have mounted it.

11.2.1 Advertise Resources

The **adv** command is used to "advertise" a directory so it is accessible to other hosts. When you advertise a directory you must assign it a resource name. This resource must have a unique name within your domain.

When **adv** is performed with the options listed below, the resource is registered with your domain name server. Any host that has access to your domain can find a listing of it from your domain's advertise table (**nsquery** command). A remote host will not know the exact location of the resource on your host. All the host will know is its resource name, the short description you assign, that it resides on your host, and the read/write permissions.

You can set up your system so resources are advertised automatically when you enter **init 3.** To do this place, the entire command line for each advertised resource in the **/etc/rstab** file.

The syntax of the **adv** command to advertise a resource is:

adv [—**r**] [—**d** "*description*"] *resource pathname* [*clients...*]

The syntax of **adv** to modify an advertised resource entry is either:

adv —**m** *resource* —**d** "*description*" [*clients* ...]

or

adv —**m** *resource* [—**d** "*description*"] *clients* ...

The options are as follows:

—r	The **—r** option, for read-only, is used to advertise the resource with read-only access. If it is not used, the remote hosts will have read/write access.
—d	The **—d** option indicates that the next argument (*description*) is a description of the resource. The description can be from zero to 32 characters.
resource	This is the resource name you assign. The name is limited to 14 printable ASCII characters; slash (/), period (.), and white space may not be used.
pathname	This is the full pathname to the directory you want to share. The directory must be on your local system, 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 clients are not included, the resource will be accessible to any Remote File Sharing host that can connect with your host. You can also define aliases so a single name can represent a group of hosts and/or domains. (See Section 11.2.1.1.)
—m *resource*	This option is used to modify the *description* or *client* fields for an advertised resource listing. It cannot be used to change the read/write permissions of a resource.

11.2.1.1 Aliases

The **adv** command reads the **/etc/host.alias** file to find the definitions of any aliases in the *clients* field. The format of the file is:

alias *name client1 client2 client3 ...*

where *name* is replaced by the character string you want to represent the list of clients. Each client can be a host name, domain name, or an alias name previously defined in the file. The three fields must be separated 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.

11.2.1.2 Resource Security

These are the levels of security that protect your resource once you have advertised it:

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 (see **rfstart —v**).
mount security	You may have advertised your resource so only selected hosts can mount it (see the *client* option of the **adv** command). If

so, a host must have proven it was one of those hosts when it first connected to your host.

user security — The permissions remote users will have to your resources is 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 that host mounts.

GENIX V.3 system security — Normal GENIX V.3 System access security, governing read, write, and execute permissions, will apply to any advertised resource.

11.2.1.3 Local Advertise Table

All advertised resources for your host are contained in your host's local advertise table. Any user can use the **adv** command with no options to display the local advertise table. The output will be a listing like the one that follows:

```
$ adv
CUSTOMER  /usr/bin/cust  read-only   "Atlanta customers"    fee fie doc.tick

SCCS      /sccs          read/write  "Project Y source"     unrestricted

CALENDAR  /usr/bin/cal   read-only   "GENIX V.3 System calendar" graph. hostgrp
$
```

The information will match what you entered using the **adv** command, with appropriate options, for each resource. Some of the information shown was implied when the resource was advertised. For example, access is read/write if **—r** is not specified and clients are not limited to certain hosts (**unrestricted**) when no clients are identified.

Clients listed in the last field can be:

- host names in your domain (**fee**)
- host names in other domains (**doc.tick**)
- domain names (**graph.**)
- aliases listed in **/etc/host.alias** (**hostgrp**)
- **unrestricted** if the resource is not restricted to certain hosts

11.2.1.4 Domain Advertise Table

All advertised resources for your domain are in the domain advertise table on your domain name server host. The **nsquery** command is available to all users to list any or all of the advertised resources in a domain. The syntax of the command is:

nsquery [—**h**] [*name*]

where the —**h** option can be used to suppress printing of the heading line and the *name* option can be replaced by one of the following:

host name	To list the resources advertised by a particular host in your domain.
domain. name	To list all resources advertised by all hosts in a domain. (A period at the end of a name causes it to be interpreted as a domain name.)
domain.host name	To list all resources advertised by a particular host in a domain outside your own domain.

If the *name* option is not used, **nsquery** will print a list of all advertised resources in its domain. Here is an example of output from an **nsquery** command:

```
# nsquery graph.fee

RESOURCE    ACCESS      SERVER     DESCRIPTION
GRAPHICS    read/write  graph.fee  Domain files
CALENDAR    read/write  graph.fee  Monthly meetings
USERHELP    read-only   graph.fee  System help information

#
```

For each available resource, **nsquery** will list the resource name (RESOURCE), the permissions (ACCESS), the host that owns the resource (SERVER), and the description of the resource.

NOTE: The output from **nsquery** does not indicate whether you have permission to access the resource.

11.2.1.5 Unadvertise

You can unadvertise any of your host's resources using the **unadv** command. It will remove the resource from the advertise tables on your host and the domain name server. The domain administrator can use **unadv** to unadvertise any resource within the domain. (You should only use **unadv** when the host has gone down, otherwise the domain and host advertise tables will be out of sync.)

Unadvertising does not remove a currently-mounted resource from a remote host (see **umount**(1M)). It does, however, prevent other hosts from mounting the resource. There are two reasons you may want to use this command.

1. Before you can unmount (**umount**) one of your file systems containing an advertised directory, it must be unadvertised.
2. If you want to restrict a previously-shared directory to only local access, you will want to unadvertise it.

Because advertise commands can be set up to run automatically in **init 3**, you may have to remove them if you want them permanently unadvertised. (See Section 11.2.1 for information on how to modify the **/etc/rstab** file.)

The syntax of the command is:

unadv *resource*

or

unadv *domain.resource*

where *resource* is used to unadvertise one of your host's resources, and *domain.resource* is used by a domain name server to unadvertise any resource in its domain. In the second case, the resource name is prefixed by a period (.) and the domain name in which it resides.

11.2.2 Mount Remote Resources

You can attach another host's advertised resource to your system using the standard **mount** command. You simply choose an existing directory, preferably empty, or create a directory to use as a mount point and **mount** the resource, using the —**d** option.

When you try to **mount** a remote resource, a request is sent to the host that advertised the resource. If you have permission to mount the resource, the resource will be added to your mount table and connected to the mount point you specified. You can list the remote resources, as well as local file systems, mounted on your host using the **mount** command without options. You can see what hosts have mounted your advertised resources using the **rmntstat** command.

The form of the **mount** command to mount a remote resource is as follows:

mount [—**r**] —**d** *resource directory*

The options are as follows:

—**r** The —**r** option indicates that the resource should be mounted read-only. If it is not used, the resource is mounted with read/write permissions. (A remote resource can only be mounted read/write if it was advertised that way.)

—**d** *resource directory*

This option is used to indicate that you are mounting a remote resource. You must substitute the resource name for *resource* and the mount point directory for *directory*.

11.2.2.1 Automatic Remote Mounts

You can set up your **mount** commands to run automatically when your host enters the **init 3** state. You do this by adding mount information to the **/etc/fstab** file. The format of **/etc/fstab** for remote mounts is as follows:

resource directory —**d[r]**

resource is replaced by the resource name, *directory* is replaced by the directory where the resource will be mounted, and —**d** says this is a remote mount. You can use —**dr** instead of —**d** if you want to mount the remote resource read-only.

11.2.2.2 Mounting Guidelines

Below are some guidelines that apply to mounting remote resources.

- Once you have advertised a resource from your host, you can:
  - — Mount a local file system on the directory or its subdirectories. The new file system will become part of the advertised resource.
  - — Mount a remote resource on subdirectories of your advertised resource. The remote resource you mount will not become part of the resource, however. Only your local users will be able to access it. (Remote users will be able to see this mount point directory, but will get a "multihop" error message if they try to **stat** the directory.)

NOTE: You cannot mount a remote resource directly on an advertised directory.

- If a resource was advertised with read-only permissions, you must mount it read-only. If it was advertised read/write, you have a choice of mounting it read-only or read/write.

11.2.2.3 Mounting Rules

There are some rules you must follow to avoid unexpected results when mounting remote resources.

Rule #1 Mounting over basic directories

A directory containing files that define your local machine should not be used as a mount point on a remote machine. This will result in essential local files being 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 inaccessible (**/dev/console**). As another example, if you mounted an **/etc** directory on your **etc** directory, you would cover your local **inittab, passwd**, and **mnttab** files, to name a few.

Some other directories that fall into this category are: **/**, **/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.

Rule #2 Mounting spool and work directories

Like Rule #1, Rule #2 has to do with mounting a directory from one computer on the same directory on another. In this case the problem is spool files and workspace directories. 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 **/usr/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.

Rule #3 File systems on devices

When a remote machine advertises a directory containing a device and that device contains a file system, you would not have access to that file system by simply mounting this device. To access the file system on the device, the remote machine would have to mount the device locally, then advertise that mount point.

Rule #4 Mounting sticky bit programs

Mounting remote resources that contain executable files with the sticky bit on can improve performance of those files. When executed on your machine, the text portion of the sticky bit program will remain in main memory on your machine, thereby reducing the network overhead on future executions. From your perspective as a client, you should be careful not to mount too many sticky bit programs or you could unknowingly gobble up a lot of memory.

If you're a server sharing your sticky bit files, you should be aware that they are treated differently from strictly local sticky bit files. Once clients have run your sticky bit program, you cannot remove that program without unmounting the resource from each of those clients. You should do this to prevent out-of-date text for recompiled or deleted files from remaining in memory on client machines. Before removing sticky bit programs from an advertised resource, you must unmount the resource from all client machines (**fumount**(1M)), remove the program, then readvertise the resource. (See Chapter 7 for a discussion of local sticky bit use.)

11.2.2.4 Host Mount Table

You can list the remote resources that are mounted on your host as you would list local file systems: the **mount** command with no options. Remote resource output from this command will appear in the form:

<directory> on *<resource>* *<permission>* on *<date>*

where *directory* is the name of the directory where the remote *resource* is mounted, the *permission* is read only/remote or read/write/remote, and *date* is the time and date the resource was mounted.

The following is an example of output from the mount command, with no options. The last two entries in this example are remote resource mounts.

```
$ mount
/ on /dev/dsk/c1d0s0 read/write on Thu Jan 16 09:07:19 1986
/usr2 on /dev/dsk/c1d0s8 read/write on Thu Jan 16 09:07:32 1986
/usr on /dev/dsk/c1d1s2 read/write on Thu Jan 16 09:07:33 1986
/s/codes on LCODE  read/write/remote on Thu Jan 16 09:10:13 1986
/s/timing on TEMPO read only/remote on Thu Jan 16 09:10:27 1986
$
```

11.2.2.5 Domain Mount Table

You can use the **rmntstat** command to find out what remote hosts have mounted your advertised resources. This command can print output for all your resources or the one you choose. The syntax is as follows:

rmntstat [—**h**] [*resource*]

where —**h** will print the output without the heading and *resource* can be used to only output information for a particular resource. Here is an example of the output from **rmntstat:**

```
$ rmntstat
RESOURCE    PATHNAME    HOSTNAME
CUSTOMER    /usr/cust   doc.host1
CALENDAR    /usr/cal    fee fie foe
SPECIAL     unknown     fie
$
```

The output shows the resources the host has advertised, where those resources are located on the local machine, and the hosts that have mounted the resource.

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.

11.2.2.6 Unmounting

You can unmount any remote resource you have mounted with the **umount** command. The syntax for using **umount** to unmount a remote resource from your host is as follows:

umount —**d** *resource*

where *resource* is replaced by the name of the resource you are unmounting.

Before you run **umount**, you can make sure none of your users are using the resource with the **fuser** command. When the **fuser** command is run, it lists the processes on your host that are accessing a mounted remote resource. It can then be used to kill all processes relating to a resource.

The form of **fuser** for reporting on remote resources is:

fuser [—**ku**] *resource ...*

where —**u** will produce a longer report of processes that have files open in any directory or subdirectory relating to the resource, and —**k** will kill all processes that have files open in any directory or subdirectory relating to the resource.

11.2.2.7 Forced Unmount

You cannot unmount a local file system using the **umount** command if any part of that file system is mounted remotely. Normally, you should tell each administrator whose host has mounted such a resource to unmount it. In this way, a resource can be removed in an orderly fashion.

When you have to unmount a local file system immediately; however, you can use the **fumount** command to remove a remotely-mounted resource from all hosts that have mounted it. You should only do this in cases where it is urgent that the resource be removed, because you may be cutting off remote processes that are accessing the resource.

The syntax of the **fumount** command is:

fumount [—**w** *sec*] *resource*

where the —**w** option says to wait *sec* seconds before remotely unmounting the resource and *resource* is replaced by the resource name.

When you execute **fumount**, this is what happens:

1. The resource is unadvertised.
2. If the **fumount** command is executed with a grace period of several seconds, this shell script is run on all hosts currently using the resource.

   /usr/bin/rfuadmin fumount *resource*

   By default, this shell script will write to all terminals:

   resource **is being removed from the system in # seconds.**

   (You can edit **rfuadmin** to tailor the action taken in response to **fumount**.)

3. After the grace period of *sec* seconds, the resource is removed from all remote hosts it is mounted on.

4. By default, **rfuadmin** will be started up again on each remote host to send this message to all terminals:

   resource **has been disconnected from the system.**

5. **Rfuadmin** then executes **rmount**. Then **rmount** will try to remount the resource every 60 seconds until it succeeds.

See **rfuadmin**(1M) and **rmount**(1M) for further information on the processing of these commands.

11.3 MAPPING REMOTE USERS

You can define the permissions remote users will have to your host's resources by mapping remote user IDs or names into user IDs or names that are defined on your host. Likewise, you can map remote groups into groups on your host.

If you don't want to map remote users, you don't have to. The default treats all remote users as a special guest ID number, which is MAXUID plus one. MAXUID is the maximum ID number defined for the system, so MAXUID+1 is always guaranteed not to overlap with any current or future users (by default, MAXUID+1 is 60001).

11.3.1 How Mapping Works

When you set up your remote user and group mapping for a remote host, you define how requests from user and groups will be handled. This mapping has an impact on the remote users' access to files and directories on your resources, as well as each remote user's view of ownership.

For example, say you map user ID **101** from machine **abc** into user ID **115** on your machine. When **101** from **abc** tries to create a file in a directory of one of your advertised resources, your machine will translate the request from **101** into a request from **115**. If **115** has permissions to create a file in that directory, then the file will be created.

If you tried to **stat** the file on your machine (**ls —l**, for example) you would see that user ID **115** was the owner. However, if a **stat** comes from machine **abc**, your machine would do inverse mapping. Therefore, the user from **abc** would see the file as being owned by **101** user ID.

Inverse mapping from the machine that owns the resource (the server) provides the most consistent file system view to a remote user. It could potentially cause confusion, though. Continuing with the example, say that instead of just mapping **101** into **115**, you also mapped **102** from **abc** into **115** on your machine. A file created by **102** would correctly create the file as owned by **115** on your machine. However, when a user from **abc stats** the file, it would always show ownership by the smaller numeric value: **101** user ID.

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.

11.3.2 Mapping Components

You must use the **idload**(1M) command to do the user and group mappings. This command reads the user and group mapping rules you create, reads your host's **/etc/passwd** and **/etc/group** files, if needed, and maps the remote users into your users' permissions. If you are using remote user and group names to map into your host, you must get user and group lists from the remote hosts, so **idload** can read the files and translate those names into the appropriate numeric ID numbers.

11.3.2.1 Rules Files

The rules files you create will tell **idload** how to map remote users. Both files are in **/usr/nserve/auth.info** under the names **uid.rules,** for user rules, and **gid.rules** for group rules.

The example below shows how the user rules file can be structured. The format of the group rules files is exactly the same. 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
```

Legend:

local_name = a local user name
local_id = a local user ID number
remote_id = a remote user ID number
remote_name = a remote user name
local = a local name or ID number
remote = a remote name or ID number

A rules file is divided into blocks of information. Each block is either a global or host block. There is only one global block per file, but there can be one host block for each host mapped.

global

This line starts the block of global information. Each line of definitions after **global** and before the first **host** line will be applied to all hosts that are not explicitly defined in host blocks. You can use **default, exclude,** and **map** 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 be applied 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 host names 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 name or an ID should only be mapped once in each block. If one is mapped more than once, the first reference is in effect.

1. **default** *local* | **transparent**

   One "default" line can be put in each block to indicate how to handle remote users and groups that aren't explicitly mapped or excluded.

   Transparent means use the same numeric ID on your machine that the user had on the remote machine for undefined users. So if a request comes from remote uid **101,** that request will have the permissions of local uid **101.**

   Local is replaced by a local user name or ID number. By default, all remote users will be mapped into the permissions of the local user indicated by name or ID. If a default line does not appear in a block, MAXUID+1 permission will be assigned.

2. **exclude** [*remote_id-remote_id*] | [*remote_id*] | [*remote_name*]

   Optional "exclude" lines can go into a block to exclude certain users from the default mapping. Zero or more ranges of ID numbers (*remote_id-remote_id*), single *remote_names*, or single *remote_id* numbers can be excluded. (*Remote_name* is not available in the global block.)

   A user who is excluded will still have access to your resources but will only have permissions of the MAXUID+1 user. All **exclude** lines must go before any map lines in a block.

3. **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. 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. For example:

   map mcn

   would give the remote **mcn** the same permissions of the local user **mcn.**

   The literal entry **all** maps all users by user name into the permissions of users with the same name on your host.

   NOTE: **Map all** and mapping by name are not allowed in a global block. **Map all** will usually produce error messages, since multiple administrative logins will have uid 0 and **idload** will try to map each one 0 to 0. There is no harm in this.

11.3.2.2 Idload Command

Once the rules files are created, the **idload** command reads your rules files and maps the users into your host. Since this command is run automatically on both hosts when a **mount** is done between two hosts, you will often run **idload** with the **—n** option. The **—n** option will print out what your mapping will be without actually updating the tables. This option is useful for making sure you have done your mapping correctly.

The syntax of **idload** is:

idload [**—n**] [**—g** *g_rules*] [**—u** *u_rules*] [*directory*]

The options are as follows:

—n	This is the "no update mode" option. When it is used, **idload** will print its results at your terminal (stdout) 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.

Each time you set up or change your rules files, first run **idload** with the **—n** option. The results will show you the mapping that will occur when the command is run to actually load the IDs.

11.3.2.3 Host Password and Group Files

If you are mapping remote users by name, you will need lists of these users from each remote host. These lists should be copies of the **/etc/passwd** and **/etc/group** files from each host. (See Procedure 10.1, Domain Setup, and Procedure 10.2, Host Setup, for information on how to distribute these files).

If **idload** finds a request for a remote user name in a *domain.host* information block, it will check the directory for that host for **passwd** and **group** files. The host's directory will be in **/usr/nserve/auth.info**/*domain/host* on your system, where *domain* and *host* are replaced by the remote host's domain and remote host's name, respectively.

NOTE: 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 will avoid the need to coordinate distributing and updating remote **passwd** and **group** files and reruning **idload.**

11.3.3 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 host.

11.3.3.1 No Mapping

If you don't run **idload** to map users, all remote users will have the permissions of the user ID number MAXUID +1, which is the maximum ID number defined on your system, plus one. Because there are no users on your system with that user ID number, remote users will only have permissions to files created by your users that are open to all users.

11.3.3.2 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 of mapping that only involve remote ID numbers.

```
global
default transparent
exclude 0
```

In this example, all remote user IDs will be mapped into the same user ID permissions on your host, except for **root** (ID number **0**), which would only have special guest permissions. This would apply to all remote hosts.

CAUTION

The **exclude 0** line is strongly recommended to prevent possible security breaches from **root** users on other systems.

```
global
default transparent
exclude 0-100
map 732:106
```

In the example above, users have the same permissions as in the previous example, except user IDs **0** through **100** will have MAXUID +1 permissions, and any user ID **732** would have the same permission as local user ID **106.**

```
global
default transparent
exclude 0-100
map 732:106

host graph.fee
default mpg
exclude 0-50
```

Here, the users from host **fee** in domain **graph** will not be mapped by the global rules. Instead, all users will have the permissions of local user **mpg** and only user IDs 0 through 50 will have MAXUID +1 permissions.

11.3.3.3 Mapping Remote Names

If you want to map specific user names into your local users you will need to have access to **passwd** and **group** files from those hosts installed on your system. Below are some examples of ways you can map remote user names.

11.3.3.4 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 graph.fee
exclude mary 0 uucp
map all
```

In the above example, each user name from host **fee** in domain **graph** will have the same permissions as the same user name on your host. The only exceptions will be users **mary, root,** and **uucp,** who will have MAXUID +1 permissions.

11.3.3.5 Map Name:Name

You can also map particular remote user names into local user names or user IDs on your host. Here is an example:

```
global
default transparent
exclude 0

host graph.fee
default transparent
exclude 0
map mcn:jcb ral gwn:103
```

Here all users from the host will be mapped into their same user ID with the following exceptions. Remote user **mcn** will have the permission of local user **jcb,** remote user **ral** will have permissions of local user **ral,** and remote user **gwn** will have permissions of local user ID **103.**

Following is the result of **idload —n** used for the example shown above. (The **gid.rules** file simply has the global block set at **default transparent.**)

```
# idload —n

TYPE    MACHINE     REM_ID      REM_NAME    LOC_ID      LOC_NAME
USR     GLOBAL      DEFAULT     n/a         transparent n/a
USR     GLOBAL      0           n/a         60001       guest_id
USR     graph.fee   DEFAULT     n/a         transparent n/a
USR     graph.fee   0           n/a         60001       guest_id
USR     graph.fee   100         mcn         105         jcb
USR     graph.fee   102         gwn         103         n/a
USR     graph.fee   191         ral         101         ral
GRP     GLOBAL      DEFAULT     n/a         transparent n/a
```

11.4 MAINTENANCE

11.4.1 Introduction

Remote File Sharing maintenance tasks are split into three groups: host administration, domain administration, and monitoring. The following host and domain administration sections describe how to maintain your Remote File Sharing host and domain. The section on monitoring describes how to check Remote File Sharing activities and tune your host to share resources efficiently.

11.4.1.1 Host Administration

You will need to learn about the following commands for administration on a Remote File Sharing host.

- **rfstart** starts up Remote File Sharing.
- **rfstop** shuts down Remote File Sharing.
- **idload** prints and updates remote user permissions to your resources.
- **rfpasswd** changes your host passwd.

- **dname** prints and changes your host's domain name or network.
- **rfadmin** lists the host that is the current name server.

Once you set up Remote File Sharing, you rarely need to use the administrative commands directly. You should use **init 3** to automatically start Remote File Sharing (**rfstart**), advertise your resources, and mount remote resources on your machine. **Init 2** will automatically stop Remote File Sharing (**rfstop**), unadvertise your resources, and unmount remote resources. Other commands—**idload, rfpasswd,** and **dname**—are used to alter the existing Remote File Sharing set up.

11.4.1.2 Domain Administration

If you are a domain administrator, you will also need to use some of the following commands to maintain the domain while logged on to the primary domain name server.

- **rfadmin —a** adds a new host to a domain.
- **rfadmin —r** removes a host from a domain.
- **rfadmin —p** resets the current name server.

Besides the commands described above, you will have to maintain the **rfmaster** file. You may also need to make sure user and group information is up-to-date for hosts in your domain.

11.4.1.3 Monitoring and Tuning

Commands used for monitoring Remote File Sharing activities are as follows.

- **sar —Du** adds information on the percent CPU time used for handling Remote File Sharing requests from other hosts to the information printed on the **sar —u** report.
- **sar —Dc** adds information on how many system calls were used for Remote File Sharing activity, including how many characters were read and written by the **read** and **write** system calls, to the information printed on the **sar —c** report.
- **sar —S** prints information on processes that handle requests from other hosts to use your resources (server processes) and the request queue.
- **fusage** prints the kilobytes of data transferred, read, and written to disk by remote hosts as well as by local users.
- **df** 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.

11.4.2 Host Maintenance

After your host is connected to a Remote File Sharing system, little maintenance is required. All you need to know is how to start up and stop Remote File Sharing. If your host is using some of the more advanced security or user mapping facilities, there are some files and tasks you should become familiar with. Below is a list of files and directories you may be interested in as a Remote File Sharing host.

NOTE: The format and contents of these files are subject to change in later releases. Where possible, use only the interfaces provided with this package to modify these files. Also, don't put any additional files in the **/usr/nserve** directory.

/etc/master.d/du This file contains the Remote File Sharing tunable parameters. (See Section 11.4.7.)

/etc/init.d/rfs This shell script is linked to a file in the **/etc/rc3.d** directory. It starts Remote File Sharing, advertises resources, and mounts resources when you enter Remote File Sharing mode (**init 3**). When you leave **init 3**, this script unadvertises, unmounts, and stops Remote File Sharing.

/usr/bin/rfuadmin This shell script defines the actions to be taken on your system when links to remote resources are dropped. You can modify this file to tailor the responses.

/usr/nserve/domain This file contains your host's domain name. It is created using the **dname**(1M) command. You should leave this file alone.

/usr/nserve/netspec This file contains the name of your host's Remote File Sharing network. It is created using the **dname**(1M) command. You should leave this file alone.

/usr/nserve/rfmaster This file contains the names and network addresses of primary and secondary domain name servers. It is created manually on the primary domain name server and transported automatically to all hosts in the domain.

/usr/nserve/loc.passwd This file contains your host's authentication password. The **rfstart**(1M) command creates this file.

/usr/nserve/auth.info/uid.rules
This file contains the rules you set up for mapping remote users.

/usr/nserve/auth.info/gid.rules
This file contains the rules you set up for mapping remote groups.

/usr/nserve/auth.info/*domain***/passwd**
This file(s) can be used for host verification. There can be a **passwd** file for the *domain* of each host you share resources with. Each entry in these files consists of a host name and its encrypted password. If your host is the primary name server

for the *domain*, this file will contain a listing of the hosts in your domain. (See **rfstart —v** for information on password validation and **rfadmin** for information on adding host passwords.)

/usr/nserve/auth.info/*domain/host*/**passwd**
This file(s) is used for user mapping. There can be one **passwd** file for every *host* in every *domain* whose users you want to map specifically by name into your system.

/usr/nserve/auth.info/*domain/host*/**group**
This file(s) is used for group mapping. There can be one **group** file for every *host* in every *domain* whose groups you want to map specifically by name into your system.

/usr/net/servers/rfsetup
This file is set up automatically by Remote File Sharing. It is used to create server processes that are used by the network listener. (This file must be owned by **root** and the group must be **adm** for the listener to work properly.)

Here is a quick way to familiarize yourself with your Remote File Sharing environment.

domain name	Type **dname —a** to see the name of your host's domain and the type of network it is running on.
current name server	Type **rfadmin** to print the host name of the current name server.
mapping rules	Type **idload —n** to see the user mapping that is done based on the **uid.rules** and **gid.rules** files in **/usr/nserve/auth.info.** They are optional, however, and therefore they may not exist.

Below are the Remote File Sharing tasks you may need to perform on your host.

11.4.2.1 Start Up Host

The **rfstart** command starts up Remote File Sharing activities on your host and registers it with the domain name server. Usually, this command is run automatically when your host enters **init 3** state. It must be done manually from run level 2 (multi-user state) the first time to start Remote File Sharing. If Remote File Sharing will not start, here are some things you can check.

- **rfmaster**

  Your host's **/usr/nserve/rfmaster** file may not be right. Use **rfstart —p** *nsaddress* to explicitly enter the primary domain name server's network address.

- domain name server

  The domain name server for your domain (primary or secondary acting as primary) must be up and running.

- added to domain

  You must be added to the domain by the domain administrator before **rfstart** will work (see **rfadmin —a**).

- listener

  The host must have a listener running for the network, and it must be configured for Remote File Sharing, for **rfstart** to work (see **nlsadmin**(1M)).

- network

  The underlying network must be running properly. This network must be some Transport Interface-compatible network, such as NPACK NETWORK.

- super-user

  You must be logged in as **root.**

11.4.2.2 Passwords and Verification

You will be prompted for a host password the first time you run **rfstart.** This password will be compared against the password entered for your host at the primary domain name server (**rfadmin —a**). Any time you run **rfstart** and your password doesn't match the one on the current domain name server, you will receive a warning, but **rfstart** will NOT fail.

Though Remote File Sharing will run, you may have a problem if the *domain*/**passwd** file from the primary domain name server is shared with other hosts to use for verification. In that case, your remote **mount** requests will fail if the passwords don't match. For this reason, it is recommended that host passwords always be kept in sync on the host and the primary name server. If passwords aren't important to you, you can simply enter a carriage return for the passwords on the host and the primary.

If you do get warnings that your password is out of sync with the current domain name server, you should handle the problem differently if the primary is the current domain name server than if the secondary has temporarily taken over.

- secondary is the current name server

  If the primary went down and a secondary took over as domain name server, the secondary may not have a *domain*/**passwd** 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 as domain name server again.

- primary is the current name server

  Try to correct your password by re-entering it with the **rfpasswd** command. If that doesn't work, follow the sequence shown below.

From primary:

rfadmin —**r** *domain.nodename*
rfadmin —**a** *domain.nodename*
Enter password for *nodename: type password*

From host:

rm /usr/nserve/loc.passwd
rfstart
rfstart: Please enter machine password: *type password*

You should then make sure that any host that verifies your host's password copies the new *domain*/**passwd** file from the primary.

11.4.2.3 Rfstart Options

There are two options you can use with **rfstart.**

—**v** Verify all hosts. If you use this option, any host that tries to mount your resources must match a name and password you have in the **passwd** file in the **/usr/nserve/auth.info/***domain* directory on your machine, where *domain* is replaced by the name of the remote host's domain. If the remote host is not listed in this **passwd** file, if it is listed and the password doesn't match, or if no **passwd** file exists, the remote mount will fail.

If you don't use the —**v** option, the following validation occurs. If a *domain*/**passwd** exists for the remote host's domain on your host and the remote host is listed, but the password doesn't match, the mount request will fail. If the host is not listed in the file or if the *domain*/**passwd** file doesn't exist, the host will be allowed to mount your resources without validation.

—**p** *primary_addr* Specify name server. This option is only needed the first time you start Remote File Sharing or if your host's **rfmaster** cannot

provide the name to the correct domain name server. *primary_addr* is the network address of the primary domain name server.

You can expect to see these active processes if Remote File Sharing is running properly. Type **ps —e** to list the processes.

```
listen
rfdaemon
nserve
rfudaemon
server
```

There may be multiple processes of some of these names running.

11.4.2.4 Shut Down Host

The **rfstop** command disconnects your host from the Remote File Sharing network. Before you can use **rfstop**, you must:

- unadvertise all your resources (**unadv**)
- unmount everything you have mounted from remote hosts (**umount**)
- make sure all your advertised resources are unmounted from remote hosts (can be forced by using **fumount**)

These steps will happen automatically when you leave init state 3.

11.4.2.5 Change Host Password

The **rfpasswd** command changes your host password, both on your host and on the primary domain name server. Processing of the new password follows the same criteria as **passwd**(1).

Since changing passwords requires communication with the primary domain name server, Remote File Sharing must be running on both your host and the primary domain name server. You cannot change your host password if the primary is down and a secondary is the current domain name server.

CAUTION

When you change your password, hosts that are authenticating your host may not automatically receive the change. If communication fails with a host after you change your password, check that they have copied the latest version of your domain's **passwd** file from your primary domain name server.

11.4.2.6 Update User and Group Information

The **idload** command updates remote user permissions to your resources. As new users and groups are added to a host, you may want to change your mapping rules if you are mapping remote names or specific user IDs.

You should regularly check to see if a host whose users you map by name has updated its **passwd** and **group** files on the domain name server. If so, you can use **idload** to re-map that information into your host. **Idload —n** is useful for checking the mapping that is currently active. For a detailed description of **idload**, see Section 11.3 or the **idload**(1M) manual page.

11.4.3 Domain Maintenance

If you are a domain administrator, you should become familiar with the basic setup of the name server hosts and a few simple maintenance tasks. You should become familiar with the files described in Section 11.4.2.

Your domain will have one primary and probably one or more secondary name server hosts. The primary will keep definitive records for the domain.

If the primary name server goes down, a secondary domain name server will quickly take over. A secondary only has limited name server responsibilities. Hosts cannot be added or removed on a secondary, for example. Domain name server responsibilities should be passed back to the primary as soon as it comes up.

Here is a quick overview of what you should know about your domain and your host.

domain name	Type **dname —a** to see the name of your host's domain and network type.
name servers	Look at the **rfmaster** file in **/usr/nserve**; This file will list all primary and secondary name servers in the domain(s) your host serves; it is maintained on the primary for each domain. If your domain can contact other domains, this file will also contain the names and addresses of those domains' name servers.
current name server	Type **rfadmin** to print the host name of the current name server. (Make sure this is your host before you do any domain administration tasks.)

Below are the basic domain maintenance tasks.

11.4.3.1 Add New Hosts

To add a new host to a Remote File Sharing domain you must be on the primary name server. It is done as follows:

rfadmin —a *name*

where *name* is replaced by the host's domain name and node name, separated by a period (for example: *dom1.host1*).

You will be prompted for an initial password for the host. It must match the password the host administrator enters when first starting up Remote File Sharing on that host (**rfstart** command). A null password (hit **<CR>**) is acceptable. After the password is entered, the system adds the host name and password to the **/usr/nserve/auth.info/***domain*/**passwd** file, where *domain* is the name of your domain.

11.4.3.2 Remove a Host

To permanently remove a host from a Remote File Sharing domain you must be on the primary. It is done as follows:

rfadmin —r *name*

where *name* is replaced by the host's domain name and node name, separated by a period (for example, *dom1.host1*). If the host is not listed in the **rfmaster** file as a primary or secondary name server, **rfadmin** will remove the entry from the *domain*/**passwd** file. Otherwise, you will get an error message, and the host will not be removed.

11.4.3.3 Change Current Domain Name Server

If your host is the current domain name server, you can pass that responsibility to another host, as follows:

rfadmin —p

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 primary, you may also want to do this if you are bringing down your system. The **rfadmin —p** command will pass domain name server information to the next host listed in the domain's **rfmaster** file. That host will then act as the current domain name server.

NOTE: Because a secondary is only intended to take over domain 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! Most domain maintenance (adding new hosts or changing passwords) cannot be done while the secondary is acting domain name server.

11.4.3.4 Reassign Domain Name Server

If you want to reassign domain name server responsibilities on a more permanent basis, you will have to change the **rfmaster** file in **/usr/nserve** as follows:

- Stop Remote File Sharing on all primary and secondary domain name servers for the domain (**rfstop** or **init 2**).
- Change the **rfmaster** file on the primary domain name server.
- Start the primary again (**rfstart** or **init 3**).
- Start the secondaries again (**rfstart** or **init 3**).

Once changed on the name servers, each individual host will pick up the change the next time it starts Remote File Sharing. See Procedure 10 for more information on the setup of the **rfmaster** file.

11.4.3.5 Add New Domains

You may want to communicate with other domains in your local area network. To do this, get the names and network addresses of the name servers for each domain and add them to the **rfmaster** file in **/usr/nserve**. See the procedures section for more information on the setup of the **rfmaster** file.

11.4.3.6 Update User and Group Files

If hosts in your domain map remote users by name, the primary name server can provide a central point for gathering hosts' password and group information. Each host will be responsible for providing this information to you. When you receive updated **passwd** and **group** files, install them in the directory for that host. The **passwd** and **group** files will go into the proper **/usr/nserve/auth.info/domain/***domain/host* directory. (*Domain* is replaced by the domain name, and *host* is replaced by the host name.)

11.4.4 Remote File Sharing Mode (init 3)

There are several steps involved in starting up Remote File Sharing and sharing resources. To simplify this procedure, a new Remote File Sharing run level has been defined: run level 3.

When you enter run level 3 using the **init 3** command, remote file sharing is automatically started via **/etc/rc3** from shell scripts in your computer's **/etc/rc3.d** directory. These scripts start Remote File Sharing, advertise local resources, and mount remote resources. When you leave run level 3 (**shutdown** or **init 2**), Remote File Sharing processes will be stopped.

You can add your own shell scripts to those that start run level 3. You can also tailor the run level 3 shell scripts to suit the way you use Remote File Sharing.

See Section 4.2 for details on the run levels. This section will describe those shell scripts used in run level 3 and suggest how to modify or add to them.

NOTE: Before you can enter Remote File Sharing mode, you must have already installed and configured Remote File Sharing. See Procedure 10.1 for information on setting up Remote File Sharing.

11.4.4.1 Entering Run Level 3

You can go into **init** level 3 in one of three ways:

- from single-user mode (run level **s**)

  Remote File Sharing mode is also a multi-user mode. Therefore, when you type **init 3** from single user mode, all multi-user processes (**gettys, cron,** etc.) will be started, followed by Remote File Sharing mode processes.

- from multi-user mode (run level 2)

  When **init 3** is run from run level 2, init checks that all multi-user processes are running, then starts the Remote File Sharing mode processes. (**Init 3** will not spawn another process for a level 2 script that is already running.)

- at boot time By default, your system will enter run level 2 at boot time. You can change that to have run level 3 start automatically at boot time by changing the value for **initdefault** in the **/etc/inittab** file so it reads as follows:

  is:3:initdefault:

11.4.4.2 Init 3 Processing

When **init 3** is run, all entries in the **inittab** file that indicate level 3 are started, including **/etc/rc3. /etc/rc3** executes all shell scripts in **/etc/rc3.d** that begin with **S.**

RFS places only one file in **/etc/rc3.d: S21rfs.** This file is linked to the **rfs** file in **/etc/init.d.** Also, **rfs** is linked to **K50rfs** in **/etc/rc2.d** and **K65rfs** in **/etc/rc0.d.**

/etc/rc3 executes **S21rfs** with the **start** option upon entering run level 3. **S21rfs** then does the following:

- Validates that the domain name has been defined for your machine.
- Validates that the **rfmaster** file has been created. (This is created automatically the first time you run **rfstart —p.** The latest copy is then sent to your machine from the primary domain name server.)
- Executes the **rfstart** command continuously, with 60 second sleep intervals, until it succeeds or returns a fatal error.
- Executes **/etc/init.d/adv** to advertise all system resources you set up in your **/etc/rstab** file. (This file contains an entire **adv** command line for each advertised resource.)
- Executes **/etc/rmountall /etc/fstab** to mount all remote resources you listed in your **/etc/fstab** file. Any remote mount that does not succeed will be tried continuously until it does via **/etc/rmount**(1M). (See Section 11.2.2 for the format of **/etc/fstab.**)

When you leave run level 3 via **init 1** or **2, rfs** is run with the **stop** option. This will execute **rfstop.**

NOTE: If for some reason **rfs** fails to terminate the **rfstart** daemon, Remote File Sharing may continue to run in the lower run state. You can surely bring down Remote File Sharing by running **unadv** and **fumount** for each advertised resource, **umount** for each mounted remote resource, and **rfstop.**

11.4.4.3 Changing Init 3 Processing

Going into init state 3 makes some assumptions about how you use your Remote File Sharing system. Here is a description of how to change some of the processing that takes place.

- verification

  If you want to use host verification to restrict access to your host, you can edit the **/etc/rc3.d/S21rfs** file to automatically run **rfstart** with the **—v** option. You must add the **—v** to about line 61 of this file, after the **rfstart** command, as shown below.

```
trap 'rm -f /usr/tmp/rfs$$;exit' 0 1 2 3 15
stat=1
retries=0
while [ ${stat} -eq 1 ]
do
 /usr/bin/rfstart -v </dev/console >/dev/console 2>/usr/tmp/rfs$$
 stat=$?
 case ${stat} in
```

- retry **rfstart**

  By default, **init 3** will keep trying to start Remote File Sharing rfstart) until it succeeds. If you want it to try a limited number of times, you must edit the **/etc/rc3.d/S21rfs** file. Find the line **retries=0** and change the number **0** to the number of times you want it to retry.

- retry mounts

  When you enter **init 3,** the system tries separately, every 60 seconds, to mount each resource listed in **/etc/fstab** until it succeeds or you leave state 3. To change this behavior you can edit **/etc/rmount.** Find the line **RETRIES=0** and change **0** (try forever) to the number of times you want to attempt to mount each resource. Find the line **TIME=60** and change 60 to the number of seconds you want it to wait between retries.

11.4.4.4 Adding RFS Mode Scripts

All files in **/etc/rc3.d** and other **/etc/rc?.d** directories are shell scripts, so you can read them to see what they do. You can modify the existing files, though it is preferable to add your own since the delivered scripts may change in future releases. To create your own scripts you should follow these rules:

- Place the file in **/etc/init.d.**
- Link the file to files in appropriate run level directories using the naming convention described below.
- Have the file accept the **start** and/or **stop** options.

You should name the files using the following conventions:

S*00name*

or

K*00name*

The filenames can be split into three parts:

S or **K**	The first letter of each file defines whether the process should be started (**S**) or killed (**K**) upon entering the new run level.
00	The next two characters represent a number from 00 to 99. These numbers indicate the order in which the files will be started (S00, S01, S02, etc.) or stopped (K00, K01, K02, etc).
name	The rest of the filename is the **/etc/init.d** filename this file is linked to.

For example, the **init.d** file **netdaemon** is linked to **rc2.d** file **S68npack** and **rc0.d** file **K67npack.** When you enter **init 2,** this file is executed with the **start** option: **sh S68npack start.** When you enter **init 0,** this file is executed with the **stop** option: **sh K67npack stop.**

11.4.5 Recovery

As a node on a Remote File Sharing network, your computer relies on other machines on the network for name service and resource sharing. When communication is cut between machines, either because a machine went down or withdrew its services, Remote File Sharing is designed to recover quickly. The following sections describe Remote File Sharing events that can occur and the recovery mechanisms designed to handle them.

11.4.5.1 Primary Goes Down

All essential domain records are maintained on the primary domain name server. This machine regularly distributes the most critical of these records to secondary domain name servers. (These records do not include files and directories under **/usr/nserve/auth.info**. See Procedure 10.1 for information on how to share this information.)

If the primary goes down, domain 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. The reason is that a secondary has limited name service capabilities. This is done to maintain the definitive domain records on the primary. Changing the name server does not affect any currently mounted resources.

While a secondary is acting domain name server, these functions cannot be done:

- maintaining host lists

  Hosts cannot be added or deleted from domain lists while a secondary is acting domain name server.

- changing host passwords

  Neither the secondary nor another host can change host authentication passwords while a secondary is acting domain name server.

The secondary will maintain lists of advertised resources for the domain and continue basic name server functions so Remote File Sharing activities can continue. In most cases, the hosts in the domain shouldn't be aware the primary is down. When the primary comes back up, the secondary should pass name server responsibilities back to the primary using the **rfadmin —p** command.

NOTE: When a primary crashes without properly shutting down Remote File Sharing and passing name server responsibilities to a secondary in an orderly fashion, the advertise table on the secondary may contain some errors. Resources from the primary may still be listed as available and recently advertised resources from other hosts may not appear on the list. You can fix the domain advertise table using **unadv** and **adv —m** commands from the domain name server.

11.4.5.2 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. The problem is that when the primary comes back up, each host will still think its resources are advertised but the primary will have no record of these advertised resources.

As soon as the primary is running, each host can make sure its advertised resources are in sync with those listed on the primary in one of two ways:

- restart Remote File Sharing

  You can bring down remote files sharing, bring it back up then readvertise your resources. This can be done automatically by going from **init 3** to **init 2** to **init 3.**

- readvertise with **—m**

  This is a less drastic way to update the advertise tables on the primary. Re-advertise each resource using the **adv —m** command from the host where the resource resides. This command will get the primary and host advertise tables back in sync.

11.4.5.3 Server Goes Down

Any machine that shares its resources with another is defined as a "server" machine. When a server goes down, machines that have mounted resources from the server will lose contact with those resources. A 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.

11.4.5.4 Rfudaemon

The **rfudaemon** process is run automatically when Remote File Sharing is started (**rfstart**) and continues to run until it is stopped (**rfstop**). The **rfudaemon** process waits for one of the following events to occur, then passes that information to the **rfuadmin** administrative shell script.

disconnect	When a link is cut to a remote resource, **rfudaemon** sends a disconnect message and the resource name to the **rfuadmin** shell script.
fumount	When a resource is unmounted (**fumount**) by the server, the **rfudaemon** sends a fumount message and the resource name to the **rfuadmin** shell script.
fuwarn	When a server sends a message that a resource is about to be unmounted (**fumount**), the **rfudaemon** sends a fuwarn message, the resource name, and the number of seconds before the resource will be unmounted to the **rfuadmin** shell script.

11.4.5.5 Rfuadmin

When links to resources are disconnected, the response to the disconnect is handled at the user level by the **rfuadmin** shell script. 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**(1): *Resource* **has been disconnected from the system.** Then it executes **fuser**(1M) to kill all processes using the resource, unmounts the resource [**umount**(1M)] to notify the kernel, and starts **rmount** to try to remount the resource.

The assumption is that the link was either broken by mistake or that as soon as the 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.

NOTE: When a resource is disconnected, **rfuadmin** tries to remount the resource using **/usr/bin/rmount**. This command retries the remount every 60 seconds until it succeeds. To change this behavior, you must either edit **/etc/rfuadmin** so it no longer does an **rmount**, or edit **rmount** so it retries a limited number of times. [See **rmount**(1M).]

11.4.6 Monitoring

This section describes the commands used to monitor Remote File Sharing activity, the reports they produce, and how to correct potential problems based on the data from these reports. In general, these reports can help you decide if you want to:

- change parameter settings to better match the way your system is used
- move resources to machines on the network that access the resource most frequently
- use sticky bit programs across the network
  (See Section 11.2.2.3 for special rules relating to sharing sticky bit programs.)

A description of all Remote File Sharing tunable parameters and suggested initial settings appear at the end of this section.

11.4.6.1 Remote System Calls (sar —Dc)

Your host collects data each time a system call sends a message across a Remote File Sharing network to access a remote file. You can print this information using **sar —Dc.**

The report produced by **sar —Dc** contains the average system calls per second, average read and write system calls per second, including average characters read and written per second, and average exec's per second, averaged over a set time period.

Information is divided into three categories: incoming requests (another host's request for your resources), outgoing requests (your host's request for a remote resource), or strictly local system calls. See Figure 11-3.

```
$ sar —Dc
fee fee 3.0 2 Series 32000  02/14/86

00:00:04    scall/s   sread/s   swrit/s   fork/s   exec/s   rchar/s   wchar/s
01:00:04
  in              4         1         2             0.00       350       220
  out             3         2         1             0.00       240       300
  local         133        30        12     0.73    1.33     11202      3813
02:00:04
  in              4         1         2             0.00       350       220
  out             3         2         1             0.00       240       300
  local         133        30        12     0.73    1.33     11202      3813
03:00:02
  in              4         1         2             0.00       350       220
  out             3         2         1             0.00       240       300
  local         133        30        12     0.73    1.33     11202      3813
04:00:02
  in              4         1         2             0.00       350       220
  out             3         2         1             0.00       240       300
  local         133        30        12     0.73    1.33     11202      3813

Average
  in              4         1         2             0.00       350       220
  out             3         2         1             0.00       240       300
  local         133        30        12     0.73    1.33     11202      3813

$
```

Figure 11-3. Output from Sar —Dc

The following paragraphs describe how information from the **sar —Dc** report can be useful to you. If performance is poor, you can see how efficiently system read and write calls to and from your host are using the Remote File Sharing network. For incoming (in) and outgoing (out) system calls, divide the characters read or written by the reads and writes, respectively.

If the average characters read and written per system call are less than 1 kilobyte per call, the network is not being used efficiently. This may be improved by using Standard I/O, or doing I/O in larger chunk sizes.

If your host is attempting more than about 30 remote system calls per second (in and out scall/s), you are probably nearing host capacity. Performance problems will probably result from this much demand. Remote execs also put a heavy demand on a host. Selective use of sticky bit programs can help improve performance.

You may want to consider moving resources to machines where they are most in demand. (See **fusage** command to see what resources are being used most heavily.)

11.4.6.2 CPU Time (sar —Du)

You can list the percent of total central processing unit (CPU) time spent on system calls from remote hosts (%sys remote) with the **sar —Du** command. See Figure 11-4.

If the percent of CPU time spent servicing remote system calls is high, your local users may be suffering. (However, if the host is a server machine, you would expect %sys remote to be high.)

To reduce the time spent servicing remote requests, you may want to place the resource(s) in demand on another host (see **fusage**(1M) command) or limit resource access by changing some of the tunable parameters. (See Section 11.4.7.) You may also want to make sure clients are doing I/O in an efficient way (see **sar —Dc**).

11.4.6.3 Server Processes (sar —S)

Every request from a remote host to access your resources is handled by a server process. When there are too many requests for the servers to handle, they are delayed and placed on the request queue. Requests leave the request queue when servers are available. Information on server availability and requests awaiting service are listed with **sar —S**. See Figure 11-5.

As an administrator you can set the number of server processes available to service remote system calls (see Section 11.4.7). There are two server variables you can set: MINSERVE and MAXSERVE. MINSERVE is the number of servers that are initially running to service remote requests.

MAXSERVE is the maximum number of servers that may ever exist. If demand goes beyond what the MINSERVE servers can handle, extra servers can be dynamically allocated; so the total number of servers can be as high as the value of MAXSERVE. These processes disappear when they are no longer needed.

Information from **sar —S** can be used to tune your server parameters as shown below.

11.4.6.4 Too Few Servers

If the receive queue is almost always busy (request %busy), you may want to raise the number of servers. Here is how to decide the parameter to raise:

- Raise the MAXSERVE if the total average servers is high.
- Raise the MINSERVE if the total average servers is low.

```
$ sar —Du
fee fee 3.0 2 Series 32000  02/14/86

00:00:04    %usr    %sys    %sys    %wio    %idle
                    local   remote
01:00:04      7      21      10      28      44
02:00:04     11       9      10       4      76
03:00:02      8      18      10      17      57
04:00:02      2       4      10       1      93
05:00:03      1       4      10       1      93
06:00:02      2       5      10       2      91
07:00:02      1       4      10       1      94
08:00:02      2       5      10       2      91
08:20:02     26      16      10      11      48
08:40:02     18      11      10       9      62
09:00:17     25      21      10      13      41
09:20:18     23      21      10      11      45
09:40:20     21      24      10      15      39
10:00:09     21      29      10      17      33
10:20:14     29      28      10      13      31
10:40:18     19      20      10       7      54

Average       9      12      10       8      71

$
```

Figure 11-4. Output from Sar —Du

```
$ sar —S

fee fee 3.0 2 Series 32000    02/14/86

00:00:04    serv/lo-hi    request    request    server    server
                3 - 6      %busy    avg lgth    %avail    avg avail
01:00:04        3           0          0         100         3
02:00:04        3           0          0         100         3
03:00:04        4          80          8          20         2
04:00:04        6         100         25           0         0

Average        15          50         15          70         2

$
```

Figure 11-5. Output from Sar —S

11.4.6.5 Too Many Servers

If servers are available nearly 100% of the time (server %avail), you may have allocated too many servers. To decide the parameter to lower:

- Check the number of total servers. If this number is near the MINSERVE value, you can lower MINSERVE. Try reducing it by 50% or by the number of idle servers.
- Check the total servers that are idle. If this number is near the MAXSERVE value, you can lower MAXSERVE. Try reducing it by 50%.

11.4.6.6 Resource Usage (fusage)

You can find out how extensively hosts are using your resources with the **fusage** command. It reports how many kilobytes were read and written from your resources, broken down by hosts who have access to the resources. The form for **fusage** for reporting on a resource you have advertised is:

fusage *advertised-directory*

where *advertised-directory* is the full path name to one of the directories you have advertised. **Fusage** with no options produces a full report of data usage for all disks and advertised directories on your system, see Figure 11-6.

```
# fusage
FILE USAGE REPORT FOR sf111

        /dev/dsk/c1d0s0     /

                    /
                                    sf111       649 Kb
                                  Clients         0 Kb
                                   TOTAL        649 Kb

        /dev/dsk/c1d0s8     /usr2

                    /usr2
                                    sf111       563 Kb
                                  Clients         0 Kb
                                   TOTAL        563 Kb
```

Figure 11-6. Output from Fusage

Below are some ways you may want to use **fusage.** If a remote host's requests for your resources are high, it may be causing performance problems on your host. With the output from **fusage,** you can see what resources are being particularly hard hit. You may then decide to move the resource. You may want to move or copy a resource to a host that is constantly accessing it.

11.4.6.7 Remote Disk Space (df)

You can use the standard **df** command with a remote resource name to see the space left on the disk on which the remote resource resides. The form of the command to report on a remote resource is:

df *resource*

where *resource* is the name of a remote resource mounted on your machine. See Figure 11-7.

```
# df USERsrc USERmail

/usr/src    (USERsrc    ):  5436 blocks    2202 i-nodes
/usr/mail   (USERmail   ):  5436 blocks*   2202 i-nodes
#
```

Figure 11-7. Output from Df

NOTE: When multiple remote resources are reported that reside on the same disk, all listings of space on that disk, after the first, will be noted with an asterisk.

If you have write permission to a resource, you have as much access to file system space as a user on the system who owns a resource. This command will tell you the potential disk space available for you to write in.

11.4.7 Parameter Tuning

There are several parameters you can tune to best suit the way you use Remote File Sharing. Remote File Sharing parameters control the amount of resources you devote to Remote File Sharing service. Each network transport provider may also have some tunable parameters that may affect performance characteristics of that particular network. See the network documentation for your network for more details.

All parameters have set default values that should work well for an average system. The sections below describe these parameters and cases where you may want to change them.

11.4.7.1 RFS Parameters

RFS parameters define the extent remote hosts can use your resources, but they also control your own remote access to remote resources. If the values are too small, you may not be providing enough resources to properly handle your Remote File Sharing load. Requests for mounts, advertises, or even a file could fail if either of those values reach the maximum number allowed for your machine. If these parameters are too large, it could result in excessive drain on your resources by remote hosts.

The parameters described below are in **/etc/master.d/du**, except for **NSRMOUNT**, which is in **/etc/master.d/kernel**. Read these files for default values. If you change any of these values, see Chapter 7 for information on how to make the updated parameters take effect.

NRCVD (maximum number of receive descriptors)

Your system creates one receive descriptor for each file or directory being referenced by remote users and one for each process on your machine awaiting response to a remote request. If you limit the number of receive descriptors, you limit the number of remote files and directories that can be accessed at a time. The result would be failed user commands. The default value is 150.

NSNDD (maximum number of send descriptors)

For each remote resource (file or directory) your users reference, your system creates a send descriptor. A send descriptor is also allocated for each server process and each message waiting on the receive queue. You can change this value to limit how many remote files and directories your host can access at a time. This would, in effect, limit the amount of remote file sharing activities your users can perform. The result would be failed user commands. The default value is 150.

NSRMOUNT (server mount table entries)

Each time a remote host mounts one of your resources, an entry is added to your server mount table. Changing this number would limit the total number of your resources that can be mounted at a time by remote hosts. The default value is 50.

NADVERTISE (advertise table)

An entry is placed in your advertise table for each resource you advertise. This parameter sets the maximum resources you can advertise. The default value is 25.

MAXGDP (virtual circuits)

There are up to two connections (virtual circuits) set up on the network between you and every host you are currently sharing resources with. There is one for each host whose resources you mount and one for each host that mounts your resources. A virtual circuit is created when a host first mounts a resource from another, and is taken down when the

last resource is unmounted.

This parameter limits the number of that Remote File Sharing virtual circuits your host can have open on the network at a time. It limits how many hosts you can share resources with at a time. Note that a given network may have a limited number of circuits on any one host, so this parameter influences the maximum percentage of those that might be used for Remote File Sharing. The default value is 24.

MINSERVE (minimum server processes)

Your system uses server processes to handle remote requests for your resources. This parameter sets how many processes are always active on your host. (See the **sar —S** command for information on monitoring server processes.) The default value is 3.

MAXSERVE (maximum server processes)

When there are more remote requests for your resources than can be handled by the minimum servers, your host can temporarily create more. This parameter sets the maximum total server processes your system can create. The default value is 6.

NRDUSER

This value specifies the number of receive descriptor **user** entries to allocate. Each entry represents a client's use of one of your resources. These entries are used during recovery when the network or a client goes down. The default value is 225.

RFHEAP

This value specifies the size in bytes of an area of memory set aside for RFS information. It contains the following information:

- The user and group ID mapping tables and the domain name of each machine currently sharing a resource(s) with this machine.
- Lists of machine names supplied as a client list on the **adv** command.

The appropriate size for **RFHEAP** depends on:

- uid/gid tables (size and number).

  There will always be two global tables, one uid and one gid. Also, any machine with a **host** entry in **uid.rules** or **gid.rules** files will have a table corresponding to each of these entries while it is connected to this machine. Machines that do not have separate entries in one of these files do not take any extra space.

  To estimate the size on an individual table, type **idload —n**. There will be one four-byte table entry per line of

output from **idload**, plus up to 24 bytes of overhead per table.

- adv client lists (size and number)

  Each advertise may have a list of authorized clients attached to it. This list is stored in this area, with its size unchanged, until the resource is unadvertised.

- currently connected resources

  Each connection will use a maximum of 64 bytes to store the name of the connected resource. This memory is allocated dynamically, so some additional space is required to account for possible fragmentation as space is allocated and de-allocated. Since the total size is likely to be relatively small, 1 to 4 kilobytes, it is best to allow too much rather than too little space. The default value is 3072.

Table 11-1 lists the Remote File Sharing parameters and recommended values for different uses of Remote File Sharing.

TABLE 11-1. RFS TUNABLE PARAMETER SETTINGS

(file /etc/master.d/du)											
Parameter	Client Only			Server Only			Client+Server			Default Value	Size per Entry in Bytes
	2M	3M	4M	2M	3M	4M	2M	3M	4M		
NSRMOUNT*	0	0	0	50	50	50	50	50	50	50	24
MAXGDP	10	15	20	24	32	32	24	32	32	24	104
NADVERTISE	0	0	0	25	25	25	25	25	25	25	32
NRCVD	40	60	80	300	400	500	150	250	350	150	44
NRDUSER	0	0	0	450	600	700	225	375	525	225	20
NSNDD	150	250	350	30	30	30	150	250	300	150	20
MINSERVE	0	0	0	3	3	3	3	3	3	3	9K
MAXSERVE	0	0	0	6	6	6	6	6	6	6	-
RFHEAP	2048	2048	2048	3072	3072	3072	3072	3072	3072	3072	1

* This tunable is found in **/etc/master.d/kernel.**

Procedure 1

SYSTEM IDENTIFICATION AND SECURITY PROCEDURES

The following procedures are covered in these sections:

Section 1.1 **Check Console Terminal Configuration** (Procedure 1.1)
To assure that the system console terminal is properly configured.

Section 1.2 **Set Time and Date** (Procedure 1.2)
To set the time and date of the internal system clock.

Section 1.3 **Establish or Change System Node Name** (Procedure 1.3)
To define the formal system name, especially for the computer to be a node in a network.

Section 1.4 **Assign Special Administrative Passwords** (Procedure 1.4)
To assign special passwords to administrative and system logins.

1.1 CHECK CONSOLE TERMINAL CONFIGURATION (Procedure 1.1)

Purpose	To assure that communication with the system is maintained.
References	The system manual for your console terminal.

All system administration functions are performed at the console terminal, which should be set at a 9600 baud rate. If the console port is operated at another data rate, you may lose communication with the system when you shutdown to the firmware mode (see Chapter 4 for an explanation of firmware mode and other operating modes).

Here are some things you might want to do to make sure your console terminal is configured properly. Use the system manual for your terminal to learn how to make these equipment checks.

- Set the input/output terminal speed option to 9600.
- Set the interface to 8-bit ASCII, Full duplex, with a parity of "none" or "space" depending on the terminal you have.
- If you lose communication with the system, check to see if the terminal is still plugged in.
- If you lose communication with the system when going to the firmware mode, make sure the input/output terminal speed option is set to 9600 and hit **RETURN.**

A printer should be part of the console equipment configuration because it provides a record of exactly what was done and how the system responded. It is essentially the system log, and it is especially advantageous when you run diagnostics. If your console terminal has this capability, the best method is to hook the printer directly to the system console. (If you have a Teletype 5425, there is an auxiliary port on the rear of the terminal. Other manufacturers' brands offer the same feature.) The console printer should be independent of the LP Spooler system.

1.2 SET TIME AND DATE (Procedure 1.2)

Purpose	To synchronize system time with clock time or to reset the system time after it has been corrupted.
Starting Conditions	System state —multi-user, for synchronizing system time with current clock time —single-user, for setting the date ahead Login—**root**, to reset clock with **date**(1)
sysadm menu	SYSTEM SETUP
Commands	**sysadm datetime**(1) **date**(1)—requires logging in as **root**
Caution	Go to single-user mode (Procedure 3.3) if you are setting the date ahead.

Setting the date ahead by one or more days should be done in the single-user mode. Setting the date ahead while in the multi-user mode with **cron** running should be avoided. The **cron** program will try to "catch-up" for the time interval involved. All the processes that were scheduled to run in the time interval are started by **cron.**

When the GENIX V.3 operating system is booted, you may receive a reminder to set the time-of-day clock. This should happen infrequently. Setting the clock, however, is normally required when you first get the system.

Step 1: To set the time and the date whenever you are in the operating system, use the System Administration Menu **datetime**. For example:

```
# sysadm datetime

Running Subcommand 'datetime' from menu 'syssetup',
SYSTEM SETUP

Current time and time zone is: 04:59 EDT
Change the time zone? [y, n, ?, q] n
Current date and time: Tue. 08/28/85 05:00
Change the date and time: [y, n, ?, q] y

Month    default 08   (1-12): <CR>
                (Using <CR> to accept the default)
Day      default 28   (1-31): <CR>
Year     default 84   (70-99): <CR>
Hour     default 05   (0-23): <CR>
Minute   default 00   (0-59): 04

Date and time will be set to: 08/28/84 05:04. OK? [y, n, q] y
The date and time are now changed.
Cron aborted: SIGTERM
The cron has been restarted to pick up the new time and/or time zone.
```

Step 2: The clock also can be set using the **date** command when you are in the operating system. You must be logged in as **root** to use **date**, and you must first take the system to single-user mode. The arguments to the **date** command are in the sequence of month, day, hour, minute, and year.

```
# date 0216131686
Sun Feb 16 13:16:00 EST 1986
```

1.3 ESTABLISH OR CHANGE SYSTEM AND NODE NAMES (Procedure 1.3)

Purpose	To define the system and node names —for a new *Series 32000* Computer —after using the floppy key —when reconfiguring the system to include basic networking
Starting Conditions	System state—multi-user or single-user Login—authorized login or **root**
sysadm menu	SYSTEM SETUP
Commands	**sysadm nodename**(1) **uname**(1) —**S**(1)—need **root** login
Caution	Do not change system and node names arbitrarily. Changes must be coordinated with your networking connections.
References	Chapter 10, Basic Networking

The system and node names of the *Series 32000* Computer can be set by any of the following means.

- Using the **sysadm nodename** command.
- Using the **uname** command.
- Reconfiguring the operating system after changing the names of SYS and NODE tunable parameters (see Chapter 7, Performance Management, for more details).

Choose one of the following methods for establishing or changing the system and node name.

1.3.1 System Administration Menu—nodename

The following command line entries and system responses show the setting of the node name using the **sysadm nodename** command. The node name is then output using the **uname** command. The contents of the **/etc/rc.d/nodename** file is the result of the execution of **sysadm nodename**.

```
# sysadm nodename
Running subcommand 'nodename' from menu 'syssetup',
SYSTEM SETUP

This machine is currently called "unix".
Do you want to change it? [y, n, ?, q] y
What name do you want to give it? [q] ABcd5678
# uname —a
ABcd5678 ABcd5678 3.0 2 VR32 db332
# cat /etc/rc.d/nodename
#       Node name changed 02/01/86 14:55:49.
#
```

1.3.2 Command—uname

The following shows how to display and change the system and node names.

```
# uname —a
unix unix 3.0 VR32 db332
# uname —S abcdefghijk
uname: name must be <= 8 letters
# uname —S ABcd5678
# uname —a
ABcd5678 ABcd5678 3.0 2 VR32 db332
#
```

NOTE: Using **uname** to change the system and node names is not as permanent as using **sysadm nodename.** Whenever the system is rebooted, the system and node names assigned to the system are those last entered through **sysadm nodename** and residing in the file **/etc/rc.d/nodename.** This file is not set up or changed by **uname.** Thus it is a good practice to use the command **sysadm nodename** to change the system and node names.

1.4 ASSIGN SPECIAL ADMINISTRATIVE PASSWORDS (Procedure 1.4)

Purpose	To permit controlled access to various administrative and special system functions.
Starting Conditions	Multi-user or single-user state, any login.
sysadm menu	SYSTEM SETUP
Commands	**sysadm admpasswd**(1) **sysadm syspasswd**(1)
Reference	"Special Administrative Passwords," Section 2.3.4., in Chapter 2, System Security

After you have set up your *Series 32000* Computer you should assign passwords to the special administrative and system logins (see Chapter 2, System Security, for definitions of these logins). The administrative logins are: **setup, powerdown, sysadm, checkfsys, makefsys, mountfsys,** and **umountfsys.**

The system logins are: **root, sys, adm, bin, uucp, nuucp, lp, rje, daemon,** and **trouble.**

Step 1: The following command will step you through all the special administrative logins to see whether you want to assign or change any of the passwords.

```
# sysadm admpasswd
        NOTE: if you want to change the passwords for
        any of these logins, you may do it with this command.
        For example, the following is displayed:
Running subcommand 'admpasswd' from menu 'syssetup',
SYSTEM SETUP

Do you want to give passwords to administrative logins? [y, n, ?, q] y
        If you enter "y", you are prompted about each administrative login.
The login 'setup' already has a password.
Do you want to change the password, delete it, or skip it? [c, d, s, q, ?] s
The login 'powerdown' already has a password.
Do you want to change the password, delete it, or skip it?
[c, d, s, q, ?] q
#
```

Step 2: To assign a password to the special system logins, enter:

```
# sysadm syspasswd
        NOTE: if you want to change any of these passwords,
        you must either be logged in as root or as one
        of these logins, and then execute the passwd command.
        For example, the following is displayed:
Running subcommand 'syspasswd' from menu 'syssetup',
SYSTEM SETUP

Do you want to give passwords to system logins? [y, n, ?, q] y
Do you want to give the 'root' login a password? [y, n, ?, q] n
The following system logins still do not have passwords:
        daemon
```

This command will only let you assign passwords to those logins that have never received a password in the first place.

Procedure 2

USER SERVICES PROCEDURES

The following procedures are covered in these sections:

Section 2.1 **Add Users or Groups** (Procedure 2.1)
To add information about new users of the system, or to name groups.

Section 2.2 **Modify User Information** (Procedure 2.2)
To change information about users or groups.

Section 2.3 **Delete Users or Groups** (Procedure 2.3)
To remove users or groups from the system.

Section 2.4 **List Users or Groups** (Procedure 2.4)
To display information about users or groups.

Section 2.5 **Write to All Users** (Procedure 2.5)
To send a message to all users logged in.

2.1 ADD USERS OR GROUPS (Procedure 2.1)

Purpose	To identify new users or groups of users to the system.
Starting Conditions	System state—multi-user
sysadm menu	USER MANAGEMENT
Commands	**sysadm adduser**(1) **sysadm addgroup**(1)
Reference	"Login Administration," Section 3.2, in Chapter 3, User Services

Step 1: Enter one of the following commands:

```
$ sysadm adduser

    or

$ sysadm addgroup
```

Step 2: If you enter the command **sysadm adduser**, you are led through the following sequence:

```
Running subcommand 'adduser' from menu 'usermgmt',
USER MANAGEMENT

Anytime you want to quit, type "q".
If you are not sure how to answer any prompt, type "?" for help,
or see the Owner/Operator Manual.

If a default appears in the question, press <RETURN> for the default.

Enter user's full name [?, q]: John Q. Public
Enter user's login ID [?, q]: jqp
Enter user's ID number (default 100) [?, q]: <CR>
   (Accepting defaults by entering <CR>)
Enter group ID number or group name
(default 1) [?, q]: <CR>
Enter user's login (home) directory name.
(default '/usr/jqp') [?, q]:<CR>

This is the information for the new login:
   User's name:   John Q. Public
   login ID:      jqp
   user ID:       100
   group ID:      1
   home directory: /usr/jqp
Do you want to install, edit or skip this entry [i, e, s, q]? i
Login installed
Do you want to give the user a password? [y, n] y
New password:
                (Enter at least six characters, one of them a numeral.)
```

```
Re-enter new password:

Do you want to add another login?[y, n, q] n
```

Step 3: If you enter the command **sysadm addgroup**, this sequence appears:

```
Running subcommand 'addgroup' from menu 'usermgmt',
USER MANAGEMENT

Anytime you want to quit, type "q".
If you are not sure how to answer any prompt, type "?" for help,
or see the Owner/Operator Manual.

If a default appears in the question, press <RETURN> for the default.

Enter group name [?, q]: seventy7
Enter group ID number (default 401) [?, q]: <CR>
   (Accepting default by entering <CR>)
This is the information for the new group:
   Group name:    seventy7
   group ID:      401
Do you want to install, edit or skip this entry [i, e, s, q]: i
Group installed.
Do you want to add another group? [y, n, q] n
```

2.2 MODIFY USER INFORMATION (Procedure 2.2)

Purpose	To change stored information about users.
Starting Conditions	System state—multi-user
sysadm menu	USER MANAGEMENT
Commands	**sysadm modadduser**(1) **sysadm modgroup** **sysadm moduser**(1) **sysadm chgloginid**(1) **sysadm chgpasswd**(1) **sysadm chgshell**(1)
Reference	"Login Administration," Section 3.2, in Chapter 3, User Services

This procedure covers three separate functions:

- to change the default values that apply to the **adduser** sequence (**modadduser**)
- to change the name of a group (**modgroup**)
- to change three of the attributes of user information (**moduser**)

Function 1, Step 1: Enter the command:

```
$ sysadm modadduser
```

Function 1, Step 2: The **sysadm modadduser** command gives you the opportunity to change either or both of the default values for group ID and home (parent) directory that appear on the **adduser** form. The screen below shows an example of changing the default group number from **1** to **100.**

```
Running subcommand 'modadduser' from menu 'usermgmt'
USER MANAGEMENT

Anytime you want to quit, type "q".
If you are not sure how to answer any prompt, type "?" for help,
or see the Owner/Operator Manual.

Current defaults for adduser:
group ID            1     (other)
parent directory    /usr
Do you want to change the default group ID: [y, n, ?, q] y
Enter group ID number or group name [?, q] 100
Do you want to change the default parent directory? [y, n, ?, q] n

These will be the new defaults:
group ID:           100
parent directory:   /usr
Do you want to keep these values? [y, n, q] y
Defaults installed.
$
```

Function 2, Step 1: To change the value of a group ID name, enter the command:

```
$ sysadm modgroup
```

Function 3, Step 1: To change the values for an individual user's login, enter the command:

```
$ sysadm moduser
```

The following menu is displayed on your terminal:

```
                    MODIFY USER'S LOGIN

1 chgloginid          change a user's login
2 chgpassword         change a user's password
3 chgshell            change a user's login shell

Enter a number, a name, the initial part of a name, or
? or <number> ? for HELP, q to QUIT:
```

Function 3, Step 2: Selecting an item from this menu starts a prompt sequence that helps you make the required change.

When a user is first given a login ID, a default shell (**/bin/sh**) is assigned. The **chgshell** subcommand enables you to assign a different shell.

Function 3, Step 3: When you select Item 3 from the menu, or if you entered the command:

```
$ sysadm chgshell
```

the following sequence appears on your terminal:

```
Running subcommand 'chgshell' from menu 'moduser',
MODIFY USER'S LOGIN

Enter user's login ID [?, q]: jqp
The current shell is /bin/sh
Enter new shell command [q]: /bin/rsh
Do you want to change the login shell of another login? [y, n, q] q
```

The above sequence assigns a restricted shell to user **jqp.**

2.3 DELETE USERS OR GROUPS (Procedure 2.3)

Purpose	To clear the system of an inactive user. To eliminate a group name that is no longer needed.
Starting Conditions	System state—multi-user
sysadm menu	USER MANAGEMENT
Commands	**sysadm deluser**(1) **sysadm delgroup**(1)
Caution	When you delete a user's ID, all the files and directories owned by that ID are deleted too.
Reference	"Login Administration," Section 3.2, in Chapter 3, User Services

Step 1: Deleting a group ID is done with this command:

```
$ sysadm delgroup
```

Step 2: The prompt sequence is as follows:

```
Running subcommand 'delgroup' from menu 'usermgmt',
USER MANAGEMENT

Which group name do you wish to delete? [q] seventy7
Do you want to delete group name 'seventy7', group ID 401? [y, n, ?, q] y
seventy7 has been deleted
Do you want to delete any other groups? [y, n, q] q
```

NOTE: The **sysadm delgroup** command deletes only the specified group and not the user login(s) assigned to that group. The logins belonging to the group must be deleted separately using **sysadm deluser.**

Step 3: Deleting a user's login ID requires more persistence. The user's home directory and all the files in and below that directory are deleted as well. Here is the sequence:

```
$ sysadm deluser

Running subcommand 'deluser' from menu 'usermgmt',
USER MANAGEMENT

This function COMPLETELY REMOVES THE USER, their mail file, home directory
and all files below their home directory from the machine.
Once this is done, there is no way guaranteed to get them all back.
        BE SURE THIS IS WHAT YOU WANT TO DO!

Enter login ID you wish to remove[q]: jqp
        'jqp' belongs to 'John Q. Public'
        whose home directory is /usr/jqp
Do you want to remove login ID 'jqp'?[y, n, ?, q] y

/usr/jqp and all files under it have been removed.

Enter login ID you wish to remove [q]: q
```

2.4 LIST USERS OR GROUPS (Procedure 2.4)

Purpose	To see what users or groups are known to the system.
Starting Conditions	System state—multi-user
sysadm menu	USER MANAGEMENT
Commands	**sysadm lsuser**(1) **sysadm lsgroup**(1)

Step 1: The two **sysadm** subcommands in this procedure enable you to see what groups and what users are in the computer. The command to list groups is:

```
$ sysadm lsgroup
```

which produces a report with these column headings:

```
Running subcommand 'user' from menu 'usermgmt'.
USER MANAGEMENT

Groups currently in the computer:
(press <RETURN> to start printing each time you hear the bell)

group     group     logins permitted to become
name      number    members using newgrp
-----     ------    ----------------------------
adm       4         root,adm,daemon
bin       2         root,bin,daemon
daemon    12        root,daemon
mail      6         root
other     1
rje       8         rje,shqer
root      0         root
sys       3         root,bin,sys,adm
$
```

Step 2: If you enter the command:

$ sysadm lsuser

the following lines appear on your terminal:

```
Running subcommand 'user' from menu 'usermgmt',
USER MANAGEMENT

Users currently in the computer
(press <RETURN> to start printing each time you hear the bell)
```

When you press <RETURN> a list in the following form is displayed:

```
login name      user name
----------      ---------
adm             0000-Admin(0000)
bin             0000-Admin(0000)
checkfsys       check diskette file system
daemon          0000-Admin(0000)
listen          Network Admin
lp              0000-lp(0000)
makefsys        make diskette file system
mount fsys      mount diskette file system
nuucp           0000-uucp(0000)
powerdown       general system administration
rje             0000-rje(0000)
root            0000-Admin(0000)
setup           general system administration
sys             0000-Admin(0000)
sysadm          general system administration
trouble         trouble(0000)
umountfsys      unmount diskette file system
uucp            0000-uucp(0000)
$
```

2.5 WRITE TO ALL USERS (Procedure 2.5)

Purpose	To send urgent messages to all users logged in.
Starting Conditions	System state—multi-user Login—**root** required to prevent users from blocking messages.
Commands	**wall**(1M)
Reference	"Write to All Users," Section 3.4.3, in Chapter 3, User Services

Step 1: For times when it is necessary to communicate with all users on the system at once, the GENIX V.3 system **wall** command is used.

 # wall

The command reads whatever you type in at your terminal until it reads an end-of-file (indicated by typing in a **control-d**).

Step 2: The message you type in is sent immediately to the terminal of all users logged in. It is preceded by:

Broadcast Message from ...

A typical use of the **wall** command is to warn users that the system is about to be shutdown:

```
Broadcast Message from root: System coming down in
ten minutes. Please log off.
```

Procedure 3

PROCESSOR OPERATIONS PROCEDURES

The following procedures are covered in these sections:

Section 3.1 **Powerup** (Procedure 3.1)
To power up the system to the multi-user state.

Section 3.2 **Powerdown** (Procedure 3.2)
To halt the system and turn the power off.

Section 3.3 **Shutdown to Single-User** (Procedure 3.3)
To bring the system to the single-user state to do administrative tasks.

Section 3.4 **Return to Multi-User** (Procedure 3.4)
To return the system to the multi-user state after it was brought to another state for administrative purposes.

Section 3.5 **Recovery from System Trouble** (Procedure 3.5)
To handle system troubles due to hardware problems and software errors.

3.1 POWERUP (Procedure 3.1)

Purpose	To turn on the system and make it available for use.
Starting Conditions	System state—(power off).
Time	Approximately 90 seconds (depending on configuration).

To power up the *Series 32000* Computer from a halted state (power is off), perform the following procedure:

Step 1: Turn on the computer.

Step 2: After the monitor is automatically loaded, boot the GENIX V.3 operating system by entering **k** and returning.

Step 3: Log on to the system when the prompt **Console Login:** appears. You can log in with a system or user login.

3.2 POWERDOWN (Procedure 3.2)

Purpose	To halt the system and turn off the power.
Starting Conditions	System state—multi-user or single-user. You must **mount /usr** to run this procedure in single-user mode. Login—authorized user or root
sysadm menu	MACHINE MANAGEMENT
Commands	**sysadm whoson**(1) **sysadm powerdown**(1) **shutdown**(1M)—**root** login only
Caution	Do not pull the plug until the powerdown procedure is completely finished.

There are differences in the procedure depending on whether you are in multi-user or single-user state.

3.2.1 From Multi-User

The best way to turn off the computer while the system is in the multi-user state is to enter the **sysadm powerdown** command. Entering this command causes the system to flush the system buffers, close any open files, stop all user processes and daemons currently running, unmount file systems, and then remove power from the computer.

Step 1: Check who is logged in before taking any action that would affect a logged-in user. Enter:

```
$ sysadm whoson
```

A typical response might be:

```
Running subcommand 'whoson' from menu 'machinmgmt',
MACHINE MANAGEMENT

These users are currently logged in:

  ID      terminal number     sign-on time
root      console             Dec 10 15:53
jaf       tty22               Dec 10 15:53
```

Step 2: Notify any users that the system is shutting down via the **/etc/wall**(1M) command (see Procedure 2.5). For example:

```
# /etc/wall <CR>
Broadcast Message from root (console) on unix Wed Feb 26 07:30:27...
The system will be coming down in 5 minutes.
Please log off. <CTRL-D>
```

Step 3: Enter:

```
$ sysadm powerdown
```

Observe the following output:

```
Running subcommand 'powerdown' from 'machinemgmt',
MACHINE MANAGEMENT

Once started, a powerdown CANNOT BE STOPPED.
Do you want to start an express powerdown? [y, n, ?, q] n
Enter the number of seconds to allow
between the warning messages (default 60) : [?, q] 30

Shutdown started.       Thu May 16 17:10:57...

Broadcast Message from root (console) on "system name"  Thu May 16 17:10:59
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.

INIT: New run level: 0
The system is coming down. Please wait.
System services are now being stopped.

Stopping process accounting

The system is down.
```

To protect your system from unauthorized power removal (from a user terminal), assign a password both to the **sysadm** login as well as to the **powerdown**(1M) command.

3.2.2 From Single-User

If the system is in the single-user state, use the following command to powerdown the system.

Step 1: To remove power and guarantee file system integrity, use the **shutdown** command as follows:

shutdown —y —i0 —g0

The arguments have the following meanings:

—y	assume yes answers to all questions
—i0	go to state 0 (off)
—g0	allow grace period of 0 seconds

Observe the following on the console:

```
Shutdown started.

The system will be shutdown in 0 seconds.
Please log off now.
THE SYSTEM IS BEING SHUTDOWN NOW !!!
Log off now or risk your files being damaged.
The system is coming down. Please wait.

The system is down.
```

3.3 SHUTDOWN TO SINGLE-USER (Procedure 3.3)

Purpose	To perform administrative tasks that should be done when no other users are on the system, such as —software installation —file backup and restore —hard disk formatting —system reconfiguration
Starting Conditions	System state—multi-user. Login—root
Commands	**shutdown**(1M)
Reference	"Going to Single-User Mode," Section 4.2.4, in Chapter 4, Processor Operations

Shutting the system down to the single-user state should be done as much as possible during off-hours, since only the console has access to the system in the single-user state.

Step 1: Log in as **root** at the console.

Step 2: Enter:

 # shutdown -i1

By default, **shutdown** prompts you about the various broadcast messages, provides a 60-second grace period between each message, and brings the system to the single-user state. When you arrive in the single-user state, you will see the following:

INIT: SINGLE USER MODE

You may now proceed with your intended tasks.

3.4 RETURN TO MULTI-USER (Procedure 3.4)

Purpose	To make the system available to users after administrative duties have been performed in either the single-user state or the firmware mode.
Starting Conditions	System state—single-user or firmware. Login—**root**
Commands	**init**(1M) **/unix** (boot program)
Reference	"A Look at Entering the Multi-User State," Section 4.2.3, in Chapter 4, Processor Operations

There are three system states from which you can return the system to the multi-user state. You can bring the system back from the single-user state and the firmware mode, and you can cause the system to immediately halt and reboot.

3.4.1 From Single-User

After administrative tasks are finished, you can bring the system back to the multi-user state from the single-user state via the **init** command.

Step 1: At the console, enter:

 # init 2

This causes **init** to inspect **/etc/inittab** and execute entries that will initialize the system to the multi-user state. The following is displayed:

```
INIT: New run level: 2

The system is coming up. Please wait.
```

The file systems are checked and the current system configuration is printed out. Finally:

```
The system is ready.

Console Login:
```

Now you can log in either as **root** or as a conventional user, since the system is in the multi-user state.

3.5 RECOVERY FROM SYSTEM TROUBLE (Procedure 3.5)

Purpose	To return the system to a usable state, after determining and recording what went wrong.
Starting Conditions	Variable —running, but throughput is degraded —running, after an automatic reboot —halted or in an unknown state Login—**root**
sysadm menu	MACHINE MANAGEMENT
Commands	**sysadm firmware**(1)—to go to firmware **/etc/shutdown**(1M)—to go to firmware **sysdump**(8)—to dump core image to diskettes **sysadm store**(1)—to copy **/unix** to diskette **/unix**—boot program
Media	One diskette for each 3/4 megabyte of memory.
Time	4 1/2 minutes per diskette

3.5.1 Perform a System Dump

Step 1: If the operating system is no longer running (it has crashed), it will place itself automatically in the stand-alone dump mode. A SYSTEM FAILURE message is displayed.

Step 2: The system will display a menu from which you can select the media to dump to. Currently, only floppy and tape dumps are supported.

Step 3: Gather the appropriate number of diskettes for the dump. Use this rule of thumb: you need one diskette for each 3/4 megabyte of memory in your configuration.

With...	Use...
1 Megabyte	2 Diskettes
2 Megabytes	3 Diskettes
3 Megabytes	4 Diskettes
4 Megabytes	6 Diskettes

Allow about 4 1/2 minutes per diskette used.

Step 4: Follow the displayed instructions, which prompt you to insert diskettes into the diskette drive.

Step 5: When the dump is finished, the program exits to the firmware monitor.

Step 6: You can return to the multi-user state by executing the boot program (see Procedure 3.4, Return to Multi-User).

In general, once you have performed the system dump, be sure to boot (execute) **/unix**.

Step 7: It is a good practice to make a copy of the current **/unix** (the version that was current at the time of the system crash) on an additional diskette. This will allow the dump to be analyzed properly by **/etc/crash**(1M).

 # sysadm store

This System Administration Menu command will prompt you for the file you want copied (**/unix**), to which media (diskette or tape).

You may want to contact your service representative for help in analyzing the system dump.

3.5.2 Example of sysdump

The following command line entries and system responses show how to execute **sysdump**. This example loads the 4 floppies produced via **sysdump**(8) or a panic crash on a machine equipped with 2 MB of memory.

```
$ ldsysdump /usr/tmp/cdump
Will extract kernel core dump from '/dev/diskette' to '/usr/tmp/cdump'
Insert kernel dump floppy #1 - Enter 'q' to quit or 'return' to continue:
EXTRACT 4096 (512 byte) blocks of kernel dump from 4 floppies
32 blocks/dot: ..........................................
Insert kernel dump floppy #2 - Enter 'q' to quit or 'return' to continue:
32 blocks/dot: ..........................................
Insert kernel dump floppy #3 - Enter 'q' to quit or 'return' to continue:
32 blocks/dot: ..........................................
Insert kernel dump floppy #4 - Enter 'q' to quit or 'return' to continue:
32 blocks/dot: ........
4 kernel dump floppies coalesced, 4096 (512 byte) blocks
$
```

Procedure 4

DISK MANAGEMENT PROCEDURES

The following procedures are covered in these sections:

Section 4.1 **Format Diskettes** (Procedure 4.1)
To prepare diskettes for use and verify they are usable.

Section 4.2 **Duplicate Diskettes** (Procedure 4.2)
To make exact copies of diskettes.

Section 4.3 **Check for Hard-Disk Errors**
To see if disk errors have been logged.

NOTE: Some variation in these procedures may occur depending on the configuration of your system. If you have a cartridge tape unit or a second diskette drive, the **sysadm** menus and prompts will reflect that.

4.1 FORMAT DISKETTES (Procedure 4.1)

Purpose	To format a diskette so it can be used by the system.
Starting Conditions	System state—multi-user or single-user You must **mount /usr** to run this procedure in single-user mode. You must be at the computer to insert and remove the diskettes or tapes. Login—an authorized login
sysadm menu	DISK MANAGEMENT
Commands	**sysadm format**(1)
Media	Unformatted diskettes
Time	Approximately 1 minutes per diskette.
Reference	"Formatting Disks," Section 5.4.1, in Chapter 5, Disk Management

Step 1: Enter the following command:

```
$ sysadm format
```

Step 2: You are prompted to select a drive or to insert the medium to be formatted, and to say whether the format should be verified; for example:

```
Running subcommand 'format' from menu 'diskmgmt',
DISK MANAGEMENT

Insert the medium in the diskette 1 drive. Presses <RETURN> when ready [q].

Formatting in progress.
```

Step 3: At the conclusion of the formatting process a message like the following one is displayed on your terminal:

The medium in the diskette 1 drive is now formatted; it may be removed.

At this point you can remove the diskette.

4.2 DUPLICATE DISKETTES (Procedure 4.2)

Purpose	To make a copy of the contents of a diskette.
Starting Conditions	System state—multi-user or single-user You must **mount /usr** to run this procedure in single-user mode. You must be at the computer to insert and remove diskettes. Login—an authorized login
sysadm menu	DISK MANAGEMENT
Commands	**sysadm cpdisk**(1)
Media	A new formatted diskette for each one to be copied.
Time	Approximately 12 minutes per copy (6 minutes to read in to hard disk; 6 minutes to read out to new diskette)
Reference	"Duplicating Disks," Section 5.5.1., in Chapter 5, Disk Management

With a single diskette drive, the technique for duplicating diskettes is to read the contents into a temporary file from the source diskette and then write it out to a new diskette. If your *Series 32000* Computer is equipped with two diskette drives, duplication is done drive-to-drive.

Step 1: From the console, enter the command:

```
$ sysadm cpdisk
```

Step 2: A prompt sequence like this appears on your terminal:

```
Running subcommand 'format' from menu 'diskmgmt',
DISK MANAGEMENT

        (If your system has two diskette drives...)
Select which drive to use:
  1 diskette   2 diskette1

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, q to QUIT: 1

Insert the ORIGINAL medium to be COPIED IN the diskette1 drive.
It is recommended that you write_protect the original.

Press <RETURN> when ready [q] <CR>
        (Accepting default by entering <CR>)

        (The sequence below is for a system with one diskette drive.)
The original is being copied in.
Copy in complete.
You may now remove the medium from diskette1 drive.
To make a copy of the original,
insert the medium TO BE WRITTEN into the diskette1 drive.
Press <RETURN> when ready. Type q to quite.
The original is being copied out onto the duplicate medium.
Copy out complete
You may now remove the medium from the diskette1 drive.
```

Step 3: At this point you may take the diskette out of the drive. You will receive the following prompt:

```
To make ANOTHER COPY of the original
insert the medium to be written into the diskette1 drive.
Press <RETURN> when ready. Type q to quit.
```

If you want another copy of what is already on the hard disk, you can insert a new diskette into the drive.

NOTE: You do not have to put the first diskette back in the drive. The copy that you put on the hard disk stays there until you quit this procedure. You can make as many copies to as many diskettes as you want, all from the one copy on the hard disk.

If you are through duplicating, you must type **q** to stop the process.

4.3 CHECK FOR HARD-DISK ERRORS (Procedure 4.3)

Purpose	To check hard disk error report
Starting Conditions	System state—multi-user or single-user You must **mount /usr** to run this procedure in single-user mode. Login—an authorized login
sysadm menu	SYSTEM DIAGNOSTICS
Commands	**sysadm diskreport**(1)
Reference	"A Bad Block Handling Scenario", Section 5.6.4, in Chapter 5, Disk Management

While new disk errors occur rarely, you should check the disk report regularly. The damage caused by uncorrected errors can be considerable.

Step 1: Enter the command:

```
$ sysadm diskreport
```

Step 2: The following report is displayed on your terminal:

```
Running subcommand 'diskreport' from menu 'diagnostics',
SYSTEM DIAGNOSTICS

WARNING: This report is provided to advise you if your machine
needs the built_in disks repaired. Only qualified repair
people should attempt to do repair. NOTE: If disk errors
are reported it probably means that files/or data have been
damaged. It may be necessary to restore the repaired disk
from backup copies.

Select "full" or "summary" disk error report [?, q]:
```

Procedure 5

FILE SYSTEM ADMINISTRATION PROCEDURES

The following procedures are covered in these sections:

Section 5.1 **Create File Systems on Diskette** (Procedure 5.1)
To define a file system on a diskette.

Section 5.2 **Create File Systems on Hard Disk** (Procedure 5.2)
To define additional file systems when more than one disk device is available.

Section 5.3 **Maintain File Systems** (Procedure 5.3)
To check and possibly repair file systems.
To monitor disk space usage.
To reorganize disk space.

Section 5.4 **File System Backup and Restore** (Procedure 5.4)
To provide a storage copy of active files.
To archive unneeded files.
To bring files and file systems back from storage.

NOTE: Some variation in these procedures may occur depending on the configuration of your system. If you have a second diskette drive, the **sysadm** menus and prompts will reflect that.

5.1 CREATE FILE SYSTEMS ON DISKETTE (Procedure 5.1)

Purpose	To define file systems that are removable for reasons of privacy or security. To write identifying labels on the magnetic medium so the system can know what is brought on line. To bring a file system under GENIX V.3 system control (**mount**), or to release it so it can be removed from the system (**unmount**).
Starting Conditions	System state—(multi-user) or (single-user) You must **mount /usr** to run this procedure in single-user mode. You must be at the computer to insert and remove diskettes. Login—an authorized login
sysadm menus	DISK MANAGEMENT
Commands	**sysadm diskmgmt**(1) **sysadm makefsys**(1) **makefsys**(1M) **sysadm mountfsys**(1) **sysadm umountfsys**(1)
Media	One or more formatted diskette for each file system to be created. Either medium with a file system that needs to be mounted.
Time	Approximately 3 1/2 minutes per diskette.
Reference	Chapter 6, File System Administration

Before doing this procedure, make sure that the diskettes you plan to use have been formatted and are not write-protected (see Procedure 4.1).

In this procedure you are prompted to name a directory that will be the mount point for a file system. Users should be informed that they should not keep other files in that directory.

Step 1: Enter one of these two commands:

```
$ sysadm makefsys

   or

$ makefsys
```

Step 2: Depending on the equipment on your system, you may be given the following prompt:

```
One subcommand or submenu name matches 'makefsys'
     1 diskmgmt/makefsys
Select one:[?, q] 1
```

Step 3: Assuming option 1 was selected, you will be prompted for further information as follows:

```
Select which drive to use:
    1. diskette1   2. diskette2
Enter a number, a name, the initial part of a name, or
? for HELP, q to QUIT: 1

Insert the medium in the diskette1 drive. Press <RETURN> when ready: [q] <CR>

Enter the label to be put on the medium [?, q] fsys01

    (This writes a label on the magnetic medium. The name should
    be six characters or less. A paper label with the same file
    system name should be affixed to the front of the diskette
    protective envelope.)

Enter the file system name [?, q] dirfs1

    (This makes a directory that will be used as the mount
    point for the file system. The name should be six characters
    or less.)

Enter the maximum number of files and directories on this medium
(default 180) [q]: <CR>

(The range is 1—711.)

Building 'dirfs1' file system on 'fsys01'.
Initializing 'dirfs01' file system.
Do you want to leave 'dirfs01' mounted? [y, n, ?, q]: y
/dirfs1 mounted. DO NOT REMOVE THE MEDIUM UNTIL IT IS UNMOUNTED!
```

The names and other responses used in the above scenario are arbitrary. Use names and responses appropriate for your situation.

Step 4: If you want to mount a file system that has already been created, you may select **mountfsys** from the **sysadm diskmgmt** menu. The procedure is as follows:

```
# sysadm mountfsys

Running subcommand "mountfsys" from menu "diskmgmt",
DISK MANAGEMENT

Insert the medium in the diskette1 drive. Press <RETURN> when ready, [q]
Disk 'fsys01', file system '/dirfs1', Mount it? [y, n, q] y
/dirfs1 mounted. DO NOT REMOVE THE MEDIUM UNTIL IT IS UNMOUNTED!
#
```

Step 5: If you want to unmount a file system that is mounted on the system, you may select **umountfsys** from the **sysadm diskmgmt** menu. Do not use **umount**(1M) if the file system was mounted with **mountfsys.**

5.2 CREATE FILE SYSTEMS ON HARD DISK (Procedure 5.2)

Purpose	To define additional file systems on a second hard disk device —to give a user group a dedicated portion of the disk space —to balance the distribution of data on the disk To write identifying labels on the magnetic medium so the system can know what is being brought on line. To bring a file system under GENIX V.3 system control (**mount**), or to release it so it can be removed from the system (**unmount**).
Starting Conditions	System state—(single-user) Login—**root**
Commands	**mkfs**(1M) **mkdir**(1) **mount**(1M) **umount**(1M)
Media	A second hard disk device.
Time	Approximately 3 minutes for **mkfs**.
Reference	"Using Mkfs," Section 6.4.2, in Chapter 6, File System Administration

Your *Series 32000* Computer is delivered with the file systems **root** and **/usr** already defined on the integral hard disk device.

If the circumstances do arise where you need to use this procedure, you will need the following information before you begin:

- The name of the *special* device file that is to contain the file system.
- The number of blocks (and optionally, i-nodes) the file system is to have.
- The interrecord gap and blocks per cylinder for the disk device.

You may package some of this information, plus details of directories and files to be copied to the new file system, in a prototype file. See **mkfs**(1M) in the *Administrator's Reference Manual* for details.

The procedure includes the **mount** and **umount** GENIX V.3 system commands, which are used to make file systems available for use or to remove them from use. Because most standard file systems are automatically mounted and unmounted during the startup and shutdown procedures, you seldom have to use these commands.

Step 1: Login as **root**.

Step 2: Make a directory to use as the mount point for the new file system.

```
# mkdir /usr2
      (/usr2 is the name of the directory where
      the new file system will be mounted.)
```

Step 3: Use the **mkfs** command like this:

```
# mkfs   /dev/dsk/c0d0s4  40000:10000  10  90

       (The arguments on the command line are:

       /dev/dsk/c0d0s4     (the name of the device on which
                            the file system resides)
       40000:10000         (the number of blocks and i-nodes)
       10                  (the rotational gap)
       90                  (the number of blocks per cylinder)
```

Step 4: When the command has been entered, you receive a prompt like this:

```
Mkfs: /dev/dsk/c0d0s4?
(DEL if wrong)
```

The command waits for 10 seconds before proceeding. If anything on the command line looks incorrect, you have a chance to cancel the command by hitting the DELETE key.

Step 5: **mkfs** next reports some of the attributes of the file system:

```
bytes per logical block = 1024
total logical blocks = 20000
total inodes = 9600
gap (physical blocks) = 9
cylinder size (physical blocks) = 90
mkfs: Available blocks = 19400
#  (root prompt)
```

The figures that appear on your screen may not correspond exactly to those shown.

Step 6: Mount the new file system with this command:

```
# mount /dev/dsk/c0d0s4 /usr2

   (The arguments on the command line are
   /dev/dsk/c0d0s4, the name of the device on which
   the file system resides, and /usr2, the mount point
   directory.)
```

Step 7: Make a directory in the new file system called **lost+found**; create a few files in the directory and remove them.

```
# cd /usr2
# mkdir lost+found
# cd lost+found
# > f1
# > f2
# > f3
# > f4
# rm f1 f2 f3 f4
```

The **lost+found** directory is used by the file system checking utility, **fsck**. Adding and removing files sets up some available i-nodes to which **fsck** can assign lost files. The recommended number of empty i-nodes for the **lost+found** directory is roughly one-quarter of the i-node count for the file system. A script like the following can be helpful:

```
#       pop -- populate (and remove)
#       usage: pop pfx [ number ]
#       makes 'number' (default = 5) files: pfx1, pfx2, ...
number=${2-5}
i=1
while test "$i" -le "$number"
do
        > $1$i
        i="expr $i + 1"
done
rm $1*
```

Step 8: The file system is now available for use. This procedure may also be used to define file systems on diskettes.

5.3 MAINTAIN FILE SYSTEMS (Procedure 5.3)

Purpose	To check and possibly repair removable file systems so the integrity of the file system is assured. To be informed on how disk space is being used so adequate resources can be provided to users.
When Performed	Before mounting the file system. On a schedule appropriate for your circumstances.
Starting Conditions	System state—(multi-user) or (single-user) For checking, the file system must NOT be mounted. For other maintainance, the file system must be mounted. To run this procedure in single-user mode, you must **mount /usr** You must be at the computer to insert and remove the media.
sysadm menus	FILE MANAGEMENT DISK MANAGEMENT
Commands	**sysadm checkfsys**(1) **sysadm diskuse**(1) **sysadm fileage**(1) **sysadm filesize**(1)
Media	Diskette that contains file system to be checked.
Time	Approximately 3 1/2 minutes per file system.
Reference	"Maintaining a File System," Section 6.5, in Chapter 6, File System Administration

5.3.1 File System Checking

Step 1: Insert the diskette or tape into its drive and close the latch.

Step 2: Enter the command:

```
$ sysadm checkfsys
```

If your system is configured with both diskette and cartridge tape drives, you will be asked to specify which **checkfsys** you want.

```
Running subcommand 'mountfsys' from menu 'mountfsys',
DISK MANAGEMENT
```

Step 3: After you have typed in your choice, you are prompted to select the appropriate drive and press **RETURN** when ready.

Step 4: You are then asked to choose a type of checking from this display:

```
Insert the medium in the diskette1 drive. Press <RETURN>'
When ready [q]

Disk 'fs1.v1', file system '/fsys01'
Select:

  check        (Any errors detected are reported, but not fixed.)
  interactive  (You are asked to approve(disapprove) fixes.)
  automatic    (Any errors detected are automatically fixed.)

[c, i, a, ?, q]:
```

Step 5: Following your response, file checking proceeds. If you selected **interactive**, and if an error is detected, you receive a message on your screen that describes the error and

asks for a **yes** or **no** response.

NOTE: The error messages are in Chapter 6, Section 6.7.5.

Step 6: When the check is completed, a message like this appears:

```
27 files 94 blocks 1266 free
You may now remove the medium from the diskette1 drive.
```

The operation is finished.

5.3.2 Monitoring Disk Usage

The second part of the file system maintenance procedure involves various ways of making sure that enough space is available on hard disk to accommodate the users' needs.

Step 1: Enter the command:

```
$ sysadm diskuse
```

Step 2: This display appears on your screen:

```
Running subcommand 'diskuse' from menu 'filemgmt'
FILE MANAGEMENT

FILE SYSTEM USAGE AS OF 05/04/86  13:23:30
File        Free    Total   Percent
System      Blocks  Blocks  Full
------      ------  ------  -------
/           1738    12510   86%
/usr        15448   43830   64%
```

Step 3: If you judge that a more detailed look at files is indicated, there are two commands you can use:

```
$ sysadm fileage

   and

$ sysadm filesize
```

Step 4: The **fileage** command causes you to be prompted for two pieces of information:

1. the full pathname of the directory to search

   It is important to be specific in your response. If you select a high-level directory, such as **/usr**, you will get a great deal more information than you want.

2. the number of days to go back

   The default is 90 days.

Step 5: **Filesize** displays information on the *n* largest files (default is 10) in a directory named by you.

The heading for the information displayed is:

```
Running subcommand 'filesize' from menu 'filemgmt',
FILE MANAGEMENT

            file size       date
owner       (characters)    last access    filename
```

The information in the display depends on your application.

5.4 FILE SYSTEM BACKUP AND RESTORE (Procedure 5.4)

Purpose	To store file systems or parts of file systems. —to guard against loss of data —to free up space on the disk
When Performed	On a schedule developed to fit the needs of your system.
Starting Conditions	System state—(single-user) You must **mount /usr** to run this procedure in single-user mode. You must be at the computer to insert and remove the media. Login—**root**
sysadm menu	FILE MANAGEMENT
Commands	**sysadm backup**(1) **sysadm store**(1) **sysadm restore**(1) **sysadm bupsched**(1) **mount**(1M)
Media	Formatted diskettes in enough quantity to hold the files or file systems you are backing up. Diskettes hold about 1400 512-byte blocks.

Time	Approximately 6 minutes per diskette.
Reference	"File System Backup and Restore", Section 6.6.9.6, in Chapter 6, File System Administration

5.4.1 Complete Backup

Step 1: Login as **root.**

Step 2: Take the system to the single-user mode (run-level S or 1; see Procedure 3.3, Shutdown to Single-User).

Step 3: The System Administration Menu package resides in **/usr** so you must **mount** that file system:

 # mount /dev/dsk/c0d0s2 /usr

Step 4: Enter the command:

 # sysadm backup

Because you logged in as **root**, you are not prompted for the **sysadm** password.

```
Running subcommand 'backup' from menu 'filemgmt',
FILE MANAGEMENT

Available file systems:
/ /usr /usr2 ALL
Enter file system(s) you want to backup [q, ?]: /usr2
Select complete or incremental backup [c, i, q, ?]: c
Print each file name as it is copied? [q,?] [y, n, q, ?]: y

Before inserting the first part into the drive, mark it as follows:

    Complete Backup of /usr2,
    Sat. 09/08/84, 05:08:31 AM
        part 1

Insert the medium in the diskette1 drive. Press <RETURN> when ready. [q] <CR>
.....(If you asked for file names to be printed, they will appear here.)
Reached end of medium on output.

Remove medium.

Before inserting the next part into the drive, mark it:

    Complete Backup of /usr,
    Sat. 09/08/84, 05:08:31 AM
        part 2

Insert the diskette. Press <RETURN> when ready. [q] <CR>
.....(If you asked for file names to be printed, they will appear here.)
Reached end of medium on output.
.
.
.
```

```
.
.
.
        (Process continues until all data are copied
        to the diskettes)
.
.
.
Reached end of medium on output.
Remove medium.

Before inserting the next part into the drive, mark it:

        Complete Backup of /usr,
        Sat. 09/08/84, 05:08:31 AM
                part n

Insert the diskette. Press <RETURN> when ready. [q] <CR>
.....(If you asked for file names to be printed, they will appear here.)
13089 blocks

Complete backup of /usr2 finished.
You may now remove the medium.
#
```

Step 5: Label each diskette or tape used for the backup. Include a sequence number as part of the label (Part 1, Part 2, etc).

Step 6: Return the system to the normal operating configuration (see Procedure 3.4, Return to Multi-User).

5.4.2 Incremental Backup

Incremental backups can be used after an initial full backup has been done. See the discussion on backup strategies in Chapter 6.

Step 1: Login as **root.**

Step 2: Take the system to the single-user mode (run-level S or 1).

Step 3: On a single-disk machine, the System Administration Menu package resides in **/usr** so you must **mount** that file system:

 # mount /dev/dsk/c0d0s2 /usr

Step 4: Execute the System Administration backup command (**sysadm backup**) and follow the displayed instructions.

```
# sysadm backup

Running subcommand 'backup' from menu 'filemgmt',
FILE MANAGEMENT

Available file systems:
/ /usr /usr2 ALL
Enter file system(s) you want to backup [?, q]: /usr2<CR>
Select complete or incremental backup [c, i, ?, q]: i<CR>
Print each file name as it is copied? [?, q] [y, n, ?, q]: y<CR>

Before inserting the tape into the drive, mark it as follows:

     Incremental Backup of /usr2,
     Sat. 09/08/84, 05:08:31 AM to
     Sat. 09/08/84, 07:04:21 AM
          Part 1
Insert the medium in the diskette1 drive. Press <RETURN> when ready. [q] <CR>
/usr2/abc/fil01
  .
  .
  .
83 blocks

Incremental backup of /usr2 finished.
You may now remove the medium.
#
```

Step 5: Label the medium as indicated in the displayed instructions.

Step 6: Return the system to the normal operating configuration (see Procedure 3.4, Return to Multi-User).

5.4.3 Selective Backup

Step 1: Login as **root.**

Step 2: Take the system to the single-user mode, run-level S or 1 (see Procedure 3.3).

Step 3: The System Administration Menu package resides in **/usr** so you must **mount** that file system:

 # mount /dev/dsk/c0d0s2 /usr

Label a formatted diskette to indicate the directory or file.

Step 4: Insert the formatted diskette in the diskette drive.

Step 5: Enter the command:

 # sysadm store

The display on your terminal is

```
Running subcommand 'store' from menu 'filemgmt',
FILE MANAGEMENT

  1. Select a single file for storing.
  2. Select all files under a directory for storing.
Enter a number: 1
Enter full path name of file to be stored [q]:
    /usr/abc/file1

  1. Select a single file for storing.
  2. Select all files under a directory for storing.
  3. List files selected so far.
  4. Store selected files.
Enter a number: 4

  1 files selected. [?, q]
```

Step 6: Each time you provide a file or directory name, you are prompted to enter more names, to review what has been entered, or to proceed with the storing process. When you have entered all the names of files to be stored, enter **4**. You will see the following display:

```
Print each filename as it is being stored? [y, n, ?, q]
Before inserting the first part into the diskette1 drive, mark it:

Files stored on:
Sat 05/04/85, 03:18:53 PM
   part 1
/usr/abc/file1
28 blocks
  .
  .
  .
Store complete.
Do you want to verify that your file(s) were stored properly? [y, n, ?, q] y <CR>
PLEASE NOTE:
To verify that the store worked properly, you must re-insert
all parts that were just written to, starting with "part 1".
Insert the medium in the diskette1 drive. Press <RETURN> when ready. <q>
Verification complete.
Should the stored files be removed from the built-in disk [y, n, ?, q]

You may remove the medium.
```

This last question gives you a chance to remove the files just written to diskette. You would probably elect to do that if you were in the process of freeing up storage space.

Step 7: Return the system to the normal operating configuration.

5.4.4 Restore

Step 1: Login as **root.**

Step 2: Take the system to the single-user mode (run-level S or 1).

Step 3: The System Administration Menu package resides in **/usr** so you must **mount** that file system:

 # mount /dev/dsk/c0d0s2 /usr

Step 4: Enter the command:

$ sysadm restore

and follow the displayed instructions. When restoring from a complete or incremental backup, all diskettes of that series must be loaded. Even if you intend to restore only a single file, all diskettes of the backup series must be loaded in sequence.

 # sysadm restore

Running subcommand 'restore' from menu 'filemgmt',
FILE MANAGEMENT

Select:
1. restore a single file
2. restore a directory of files
3. restore all files
4. list all the files

Enter a number [?, q]: **1**
Insert the medium in the diskette1 drive. Press <RETURN> when ready. [q]<**CR**>
Enter full path name of file(s) to be restored [q, ?]: **/usr/abc/file1**
Do you want to rename each file as it is copied in? [y, n, q] **y**
WARNING:

Be very careful when you rename a file. Files incorrectly named by typing errors are difficult to find and repair.
Remember that only the first 14 characters of each part of the file name (i.e. the characters between the "/"s) are significant.

(You will be asked to rename each file in turn. An empty response (<**CR**>*) skips that file. An answer of period (.) restores the file with its original name.)*

Rename </usr/abc/file1>
/usr/abc/datafile3
.
.
.
83 blocks

```
Restoration complete.
You may now remove the medium from the diskette1 drive.
Select:
        1. restore single files
        2. restore a directory of files
        3. restore all files
        4. list all the files
Enter a number [?, q]: q

#
```

Step 5: Return the system to the normal operating configuration (see Procedure 3.4, Return to Multi-User).

Step 6: Store the backup diskettes in a safe place.

5.4.5 Backup Schedule Reminders

You may do this procedure with any authorized login.

Step 1: Enter the command:

```
$ sysadm bupsched
```

Step 2: The following display will appear on your terminal:

```
                    BACKUP REMINDER SCHEDULING

1 schedcheck  schedule backup reminder checks
2 schedmsg    schedule backup reminder message

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, q to QUIT:
```

Step 3: To schedule a reminder message to be sent to the system console during **shutdown,** enter a **2.** The following message is displayed on your terminal.

Enter the command you wish to execute [p, r, a, m, w, q, ?]:

The choices for this prompt (and what is displayed if you enter a question mark) are:

```
p - Print lines of the file
r - Remove a line or group of lines
a - Add a line
m - Modify a line
w - Write the changes into the file
q - Quit - Leave bupsched
```

The file referred to contains reminder messages and the schedule for sending the messages to the system console.

Step 4: Enter **a** to add a line, and the following sequence of prompts occurs:

Enter time intervals in which backup reminder messages are to be printed [q, ?]: **16:00-18:00**

(The time interval must be entered with no blanks)

Enter time intervals in which backup reminder messages are to be printed [q, ?]: **q**

(Many prompts in this sequence are repeated to let you enter additional information. When you have entered as much as is appropriate for the particular prompt, a **q** *takes you to the next prompt.)*

Enter the day of the week [0-6, *, q, ?]: *****

(The asterisk (*) means you want to schedule the message to appear every day. Day 0 is Sunday.)

Enter the day of the month [1-31, *, q, ?]: **15**

(The **15** *means you want to schedule the message to appear on the 15th of the month. This modifies the Day-of-the-Week*

schedule to mean any day that is the 15th of the month.)

Enter the day of the month [1-31, *, q, ?]: **q**

(The prompt is repeated to let you enter additional days for this message to be scheduled. **q** *ends the sequence.)*

Enter the month [1-12, *, q, ?]: ***

(The asterisk (*****) means you want to schedule the message to appear every month.)

.
.
.

.
.
.

Enter the file system you wish to backup [q, ?]: **/usr2**

(The file system name must start with a slash (/). The prompt will re-appear until you enter **q**.)

Enter the command you wish to execute [p, r, a, m, w, q, ?]: **w**

(The **w** *means you want what you just entered added to the file of reminder messages.)*

Enter the command you wish to execute [p, r, a, m, w, q, ?]:

The sequence is over. You can exit with a **q** or choose another function.

Step 5: To see all the scheduled reminder messages, enter a **p.** The following display appears on your terminal:

NOTE: In this display the pound sign (#) means the line is interpreted as a comment when the file is executed.

```
#
#       Format of lines
#
#time    day     month     list
#
#       time    -  time(s) of day (24hr or am/pm)
#       day     -  day(s) of week (mon, tue, etc)
#                  day(s) of month (1,2,...first, last)
#       month   -  month(s) of the year (jan, feb,...)
#       list    -  list of file systems to be backed up
#                  or command to be executed ( !command line)
#
#       Example:
#
#4:00-18:00  mon  *  /usr
#
#       If ckbupscd is invoked between 4:00 and 6:00 in the
#       evening on mondays during any month of the year,
#       display /usr as the name of a file system that needs
#       to be backed up.
#
#================================================
#
#       Default backup schedule calls for daily backups of /usr
#       and monthly backups of root (/) on the 15th of each month.
#

 1    0:00-8:00,16:00-23.59  mon,tue,wed,thu,fri   *  /usr
 2    4pm-11pm        15        *          /
 3    16:00-18:00     15        *        /usr2
```

(*Line 3 in the list is the line you just added.*)

Step 6: Once there are reminder messages in the file, you can use **schedcheck**, selection 1 from the **bupsched** menu, to schedule checks for reminder messages. Reminder messages are sent to the console if **shutdown** occurs during the time interval specified in the file.

Schedule check is an added protection in case a **shutdown** does not take place during the specified time interval. It looks for messages that would have been sent had a **shutdown** occurred. The prompt sequence is similar to the one shown above.

Procedure 6

SYSTEM RECONFIGURATION PROCEDURES

The following procedures are covered in these sections:

Section 6.1 **Reconfigure the System** (Procedure 6.1)
To rebuild the operating system after tuning resulting from modifications to hardware or operating system software.

Section 6.2 **Unbootable Operating System Recovery**
To recover if you create an unbootable operating system after attempting a system reconfiguration.

6.1 RECONFIGURE THE SYSTEM (Procedure 6.1)

Purpose	To make a new **/unix.** To incorporate tunable changes resulting from hardware and software changes to the system.
When Performed	Only when the system must be tuned.
Starting Conditions	System state—(multi-user) Login—**root**
sysadm menu	MACHINE MANAGEMENT
Commands	**/etc/config**(1M) **sysadm reboot**(1)
Caution	1. Always copy the new **/unix** to **NEWunix.** 2. Do not change **nodename** during reconfiguration without coordinating it with interfacing systems.
Reference	Chapter 7, Performance Management

System reconfiguration is necessary whenever the physical configuration of the computer or the software configuration of the operating system itself changes. This happens when you upgrade your hardware (such as add more memory or disks), or when you edit tunable parameters to improve performance. The only way the system can understand such changes is by rebuilding it from its (partly modified) source files. This procedure allows you to modify and reconfigure (rebuild) the system residing in **/unix.**

As was shown in the table at the beginning of this procedure, there are three things to make sure of:

1. Always copy the new kernel to **/NEWunix.** If the new kernel does not boot, the old one will be saved automatically in **/OLDunix.**
2. When reconfiguring the operating system do not arbitrarily change the node name (NODE) of the *Series 32000* Computer. Once basic networking has been established, a change in node name must be coordinated with all interfacing systems. If not properly coordinated, a calling system will fail sequence checks because the returned name does not match the name stored in **/usr/lib/uucp/Systems.**

There are two major steps in reconfiguring the operating system:

1. edit the **/etc/master.d** files to modify the tunable parameters, and
2. rebuild the operating system.

6.1.1 Modifying the Tunable Parameters

The major steps in incorporating changes to the tunable parameters are as follows.

Step 1: Login as **root.**

Step 2: Change the present working directory to **/etc/master.d.**

```
# cd /etc/master.d
```

Step 3: Edit the applicable files in the **/etc/master.d** directory to modify (increasing or decreasing the value of) the tunable parameters. Refer to Table 7-1 in Chapter 7 for a complete listing of recommended values for the tunable parameters available with your system.

6.1.2 Rebuilding the Operating System

The major steps in rebuilding the operating system are as follows (note you must be logged in as **root**):

Step 1: Change the present working directory to **/usr/conf/kernel.**

```
# cd /usr/config/kernel
```

Step 2: Build a new kernel with the **makefile** in the current directory, **/usr/conf/kernel.** The **make** command runs **/etc/config** on all files in the **/etc/master.d** directory and relinks the UNIX kernel. Enter:

```
make unix
```

Step 3: Copy the new kernel to **/NEWunix:**

cp /usr/conf/kernel/unixdb332 /NEWunix

Step 4: Return to the firmware and reboot **/unix** (not **/NEWunix**). The reboot procedure will automatically:

boot **/unix** to an abbreviated single-user state

a. copy **/unix** to **/OLDunix**

b. copy **/NEWunix** to **/unix**

c. return to the firmware, and

d. reboot **/unix** (now the new one recently built)

To reboot the system:

```
# sysadm reboot
Running subcomamnd 'reboot' from menu 'machine mgmt,'
MACHINE MANAGEMENT
Once started a reboot CANNOT BE STOPPED.
Do you want to start an express reboot? [y, n, ?, q]
```

A new **/unix** will be generated automatically during the boot procedure.

6.2 UNBOOTABLE OPERATING SYSTEM RECOVERY (Procedure 6.2)

Purpose	To recover from an unbootable **/unix**. To get a viable version of the system running after an unsuccessful attempt at reconfiguring the system.
Starting Conditions	System state—variable. Login—**root**
sysadm menu	MACHINE MANAGEMENT
Commands	**sysadm firmware**(1)—to get to firmware
Bootable Programs	**/OLDunix**—boot program (from hard disk)
Reference	Chapter 7, Performance Managment

If you create a **/unix** that is unbootable or operates so poorly that recovery while operating in that version is impossible, do the following.

Step 1: If the system comes up to the point where you get the **Console login:** prompt, login as root and take the system to firmware mode.

 # sysadm firmware

Step 1a: If the system fails to come up, retry. At the point where the word DIAGNOSTICS appears, press the RESET switch. This takes the system directly to firmware mode.

Step 2: Enter the montior boot command:

#b

Step 3: In response to the boot prompt, "32boot:" or "boot:" enter the disk ID of the disk to be booted. The hard disk would be:

 # dc 0

Step 4: After the operating system has loaded and executed the stand-alone shell, you should see a double dollar prompt ($$). Enter the old bootable unix saved as **/OLDunix:**

$$ /OLDunix

NOTE: If you did not make this copy, reload the operating system using the core system diskettes via the partial restore procedure (see Procedure 3.9).

Step 5: After the system has rebooted and you have logged in again, move **/OLDunix** back to **/unix:**

 # mv /OLDunix /unix

This will protect you from having to do this procedure again in case of a sudden system crash.

Step 6: Finally, try to determine what went wrong. Consider first that you may have placed an incorrect value in the parameter(s) you were working with.

Step 7: Repeat Procedure 6.1 to correct errors in the previous reconfiguration. If you cannot determine what went wrong, document what happened as thoroughly as you can, and then contact your service representative.

Procedure 7

LP SPOOLER ADMINISTRATION PROCEDURES

The following procedures are covered in these sections:

Section 7.1 **Install the LP Spooler** (Procedure 7.1)
To install the LP Spooler Utilities and an LP printer in the *Series 32000* Computer: references to pertinent documentation.

Section 7.2 **Shut Down for Maintenance** (Procedure 7.2)
To either turn off the scheduler or a printer for administrative or maintenance purposes.

Section 7.3 **Restart the Scheduler** (Procedure 7.3)
To restart the LP scheduler and enable any printers.

Section 7.4 **Add Another LP Printer** (Procedure 7.4)
To add another LP printer to the current configuration.

7.1 INSTALL THE LP SPOOLER (Procedure 7.1)

Purpose	Provide checklist of essential information needed to install the LP Spooler.
References	Installation *Owner/Operator Manual* —Chapter 6, Utilities Packages User Environment Definition *User Environment Utilities Guide* (Contact AT&T for ordering information.) Appropriate printer manuals —*Dot Matrix Printer Manual* (Contact AT&T for ordering information.) —*Letter Quality Printer Manual* (Contact AT&T for ordering information.) Optional manual —*Expanded Input/Output Capability Manual* (Contact AT&T for ordering information.)

Chapter 6 of the *Owner/Operator Manual* contains the installation procedure for the LP Spooler Utilities Package. The other manuals listed provide information you may need to install specific printers.

7.2 SHUT DOWN FOR MAINTENANCE (Procedure 7.2)

Purpose	To turn off either the scheduler or a printer.
Starting Conditions	Multi-user or single-user state Login—an authorized login
Commands	**lpstat**(1) **2/usr/lib/lpshut**(1M)
Reference	"Administrative Commands", Section 8.2, in Chapter 8, LP Spooler Administration

Step 1: To execute **lpadmin**, the LP scheduler cannot be running. Enter the following command to see if the scheduler is running:

```
$ lpstat —r
scheduler is running
```

Step 2: If the scheduler is running, enter the following command to stop it:

$ /usr/lib/lpshut

This stops the scheduler completely. The printer stops, and requests for printing are not acted on.

7.3 RESTART THE SCHEDULER (Procedure 7.3)

Purpose	To make the LP system available again after administrative tasks or maintenance.
Starting Conditions	Multi-user or single-user state Login—an authorized login
Commands	**/usr/lib/lpsched**(1M) **enable**(1) **2/usr/lib/accept**(1M)
Reference	"Administrative Commands", Section 8.2, in Chapter 8, LP Spooler Administration

Step 1: After the appropriate administrative tasks have been completed, the scheduler can be restarted. Enter the command:

$ /usr/lib/lpsched

to restart the LP scheduler. To verify that the scheduler restarted, enter:

```
$ lpstat —r
scheduler is running
```

Step 2: When you are ready to start printing, be sure that the printer is ready to receive output. For several printers, this means that the top of form has been adjusted and that the printer is on-line. Enter the command:

$ enable *printer-name*

to enable printing to occur on the printer.

Step 3: Enter the following command to tell the scheduler to accept requests for the printer you are installing:

$ /usr/lib/accept *printer-name*

7.4 ADD ANOTHER LP PRINTER (Procedure 7.4)

NOTE: This procedure assumes that an Expanded Input/Output Capability feature card, with available ports, is installed in your *Series 32000* Computer.

Purpose	To meet the need for expanded output facilities.
Starting Conditions	Multi-user or single-user state You must **mount /usr** to run this procedure in single-user mode. Login—**root**
Commands	**lpstat**(1) **/usr/lib/lpshut**(1M) **lpadmin**(1M) **/usr/lib/lpshed**(1M) **/usr/lib/accept**(1M) **enable**(1)
Reference	"Administrative Commands", Section 8.2, in Chapter 8, LP Spooler Administration

When you install the LP Spooling Utilities, a printer can be easily added to the LP system by following the interactive installation script. If you need to add a printer at a later time, you have to add it manually. This section leads you through the steps required to manually add an LP printer.

Step 1: Ensure that only "lp" can write to the device file to avoid unwanted output from non-LP processes. Log in as root and enter the following commands:

```
# chown lp /dev/ttyxx
# chmod 600 /dev/ttyxx
```

xx is the port number to which the LP printer will be connected.

Step 2: Modify the port entry in the **/etc/inittab** file to add an LP printer to one of the Input/Output (I/O) expansion ports. The command **sysadm modtty** can be used to turn off the **getty** process on a tty line (see Procedure 8.3, Modify TTY Line Characteristics).

Step 3: Turn off the LP scheduler with the **/usr/lib/lpshut** command (see Procedure 7.2, Shutting Down for Maintenance).

Step 4: Introduce a printer to the LP system with the **lpadmin** command. The format of the **lpadmin** command is as follows:

lpadmin —**p***printer* —**v***device* [—**e***printer*|—**i***interface*|—**m***model*]

The following is an example command line for adding printer dqp10_3 to the LP system. The interface program used in the sample command line is the dqp10 model interface program and the I/O port is tty12.

$ /usr/lib/lpadmin —pdqp10_3 —v/dev/tty12 —mdqp10

Step 5: Restart the LP scheduler with the **/usr/lib/lpsched** command.

Step 6: Tell the scheduler to accept requests with the **/usr/lib/accept** command.

Step 7: Enter the command:

$ enable *printer-name*

to enable printing to occur on the printer.

Procedure 8

TTY MANAGEMENT PROCEDURES

Section 8.1 **Check TTY Line Settings** (Procedure 8.1)
To tell what line settings are defined.

Section 8.2 **Make TTY Line Settings** (Procedure 8.2)
To create new TTY line settings and hunt sequences.

Section 8.3 **Modify TTY Line Characteristics** (Procedure 8.3)
To change the characteristics of TTY lines. To turn lines on or off.

NOTE: The procedures described in this section apply to Release 2.1 and later releases.

8.1 CHECK TTY LINE SETTINGS (Procedure 8.1)

Purpose	To tell what line settings are defined.
Starting Conditions	System state—multi-user or single-user Login—an authorized login
sysadm menu	TTY MANAGEMENT
Commands	**sysadm lineset**(1)
Reference	"How the TTY System Works," Section 9.2.1, in Chapter 9, TTY Management

Step 1: Enter this command to go directly to the lineset display:

```
$ sysadm lineset
```

Step 2: This display appears on your terminal:

```
Running subcommand 'lineset' from menu 'ttymgmt',
TTY MANAGEMENT

Tty Line Settings and Sequences

console1  console2  console3  console4  console5  console
contty1   contty2   contty3   contty4   contty5   contty
contty1H  contty2H  contty3H  contty4H  contty5H  conttyH
pty       (does not sequence)
300       19200     9600      4800      2400      1200
300H      4800H     9600H     19200H    2400H     1200H
```

Each of the line settings is just a name, used to identify a set of tty line characteristics. During the **login** process, the line settings on one line "hunt" from left to right, moving from one to the next on receiving a **BREAK** signal. The rightmost setting on each line hunts to the first one again, forming a circular hunt sequence.

Note that the "pty" setting does not sequence. Sending a **BREAK** will not make it change in any way.

Step 3: To look at a line setting in detail:

```
Select one line setting to see it in detail [?, q]: 1200

Line Setting:          1200
     Initial Flags:    B1200 HUPCL
     Final Flags:      B1200 SANE IXANY TAB3 HUPCL
     Login Prompt:     login:
     Next Setting:     300

B1200     1200 Baud
HUPCL     Hang Up on Last Close
IXANY     Enable Any Character to Restart Output
SANE      Set All Modes To "Traditionally Reasonable" Values
TAB3      Expand Horizontal-tab To Spaces
```

Step 4: Notice that we didn't have to start with the leftmost entry of a row. Any entry can be specified.

```
Select another line setting or
<RETURN> to see the original list [?, q]: 300

Line Setting:          300
     Initial Flags:    B300 HUPCL
     Final Flags:      B300 SANE IXANY TAB3 HUPCL
     Login Prompt:     login:
     Next Setting:     19200

B300      300 Baud
HUPCL     Hang Up on Last Close
IXANY     Enable Any Character to Restart Output
SANE      Set All Modes To "Traditionally Reasonable" Values
TAB3      Expand Horizontal-tab To Spaces

Select another line setting or
<RETURN> to see the original list [?, q]: q
```

8.2 MAKE TTY LINE SETTINGS (Procedure 8.2)

Purpose	To create new line settings and hunt sequences.
Starting Conditions	System state—multi-user or single-user Login—an authorized login
sysadm menu	TTY MANAGEMENT
Commands	**sysadm mklineset**(1)
Reference	"How to Create New Line Settings and Hunt Sequences," Section 9.2.3, in Chapter 9, TTY Management

In this procedure, we are undertaking to connect a dual-speed modem to the computer to handle 300 baud and 1200 baud. There is a 1200 setting already in the table, but if the users somehow miss the speed, they will have to hit BREAK many times to hunt to 1200 again. Since we are interested in only the two speeds, let's create a new 1200-300 sequence:

Step 1: Enter this command to go directly to the **mklineset** display:

```
$ sysadm mklineset
```

Step 2: This sequence of prompts appears on your terminal:

```
Running subcommand 'mklineset' from menu 'ttymgmt',
TTY MANAGEMENT

Enter the name of the new tty line setting [?, q]: 1200300
Select a baud rate [?, q]: ?  (To ask for HELP)

Available baud rates:

     50   110   150   300   1200   2400    9600
     75   134   200   600   1800   4800   19200

Select a baud rate [?, q]: 1200

Enter the login prompt you want (default = "login: ") [?, q]: <CR>

   (Accepting the default)

Do you want to add another tty line setting to the sequence? [y, n, q] y

Enter the name of the new tty line setting [?, q]: 3001200
Select a baud rate [?, q]: 300
Enter the login prompt you want (default = "login: ") [?, q]: <CR>
Do you want to add another tty line setting to the sequence? [y, n, q] n

Here is the tty line setting sequence you created:

1200300    3001200

Line Setting:        1200300
   Initial Flags:    B1200 HUPCL
   Final Flags:      B1200 SANE IXANY HUPCL TAB3
   Login Prompt:     login:
   Next Setting:     3001200
Line Setting:        3001200
   Initial Flags:    B300 HUPCL
```

Final Flags: B300 SANE IXANY HUPCL TAB3
Login Prompt: login:
Next Setting: 1200300

B1200 1200 Baud
B 300 300 Baud
HUPCL Hang Up on Last Close
IXANY Enable Any Character to Restart Output
SANE Set All Modes To "Traditionally Reasonable" Values
TAB3 Expand Horizontal-tab To Spaces

Do you want to install this sequence? [y, n, q] **y**
Installed.

8.3 MODIFY TTY LINE CHARACTERISTICS (Procedure 8.3)

Purpose	To modify TTY line settings or turn line on or off.
Starting Conditions	System state—multi-user or single-user Login—an authorized login
sysadm menu	TTY MANAGEMENT
Commands	**sysadm modtty**(1)
Reference	"How to Modify TTY Line Characteristics", Section 9.2.4, in Chapter 9, TTY Management

The objective here is to tell the computer which port to use with the line settings defined above in Procedure 8.2.

Step 1: Enter this command to go directly to the **modtty** display:

```
$ sysadm modtty
```

Step 2: This sequence of prompts appears on your terminal:

```
Running subcommand 'modtty' from menu 'ttymgmt',
TTY MANAGEMENT

Changeable tty lines:
   contty   tty10   tty11   tty12   tty13   tty14   tty15   tty16   tty17
Select the tty you wish to modify,
or enter ALL to see a report of all ttys [?, q]: ALL

Changeable tty lines:

                        Hangup      Line
Tty       State         Delay       Setting       Description
_____     _______       _______     ___________   ____________________
contty    on            60          4800
tty10     off           off         9600          NOTE: These last 8
tty11     off           off         9600          entries will be
tty12     off           off         9600          printed only if
tty13     off           off         9600          the cd91 package
tty14     off           off         9600          has been
tty15     off           off         9600          installed
tty16     off           off         9600
tty17     off           off         9600

Continue (default: y)? [y, n, q] <CR>

Changeable tty lines:
   contty   tty10   tty11   tty12   tty13   tty14   tty15   tty16   tty17

Select the tty you wish to modify,
or enter ALL to see a report of all ttys [?, q]: tty14

tty14: current characteristics:
   State                  off
   Hangup Delay           off
   Line Setting           9600
   Description

Available states:
   off   on
Select a state (default: off) [?, q]: on

Enter a hangup delay, in seconds, or 'off' (default: off) [?, q]: 45
```

(*Because this is a dial-up line, we want to specify a* **timeout** *figure.*)

.
.
.

Available line settings:

console	console5	contty2	contty4	conttyH	1200	4800	19200
console1	contty	contty2H	contty4H	pty	1200H	4800H	19200H
console2	contty1	contty3	contty5	300	2400	9600	1200300
console3	contty1H	contty3H	contty5H	300H	2400H	9600H	3001200
console4							

Select a line setting (default: 9600) [?, q]: **1200300**

Current description:

Enter a new description (default: current description) [?, q]:
1200/300 baud dial in line

tty14: new characteristics:

State	on
Hangup Delay	45
Line Setting	1200300
Description	1200/300 baud dial in line

Do you want to install these new characteristics? [y, n, q] **y**
tty14 now has new characteristics.

Changeable tty lines:
contty tty11 tty12 tty13 tty14 tty15
Select the tty you wish to modify,
or enter ALL to see a report of all ttys [?, q]: **ALL**

Changeable tty lines:

Tty	State	Hangup Delay	Line Setting	Description
contty	on	60	4800	
tty10	off	off	9600	
tty11	off	off	9600	
tty12	off	off	9600	
tty13	off	off	9600	
tty14	on	45	1200300	1200/300 baud dial in line
tty15	off	off	9600	
tty16	off	off	9600	
tty17	off	off	9600	

Continue (default: y)? [y, n, q] **q**

Procedure 9

BASIC NETWORK PROCEDURES

The following procedures are covered in these sections:

Section 9.1 **Install Basic Networking Software** (Procedure 9.1)
To place basic networking software on the hard disk.

Section 9.2 **Set Up Basic Networking Files** (Procedure 9.2)
To configure basic networking files.

Section 9.3 **Basic Networking Maintenance** (Procedure 9.3)
To maintain basic networking files and operations.

Section 9.4 **Basic Networking Debugging** (Procedure 9.4)
To track down problems in basic networking.

Section 9.5 **Remove BNU Software** (Procedure 9.5)
To remove basic networking software from the hard disk.

Section 9.6 **Set Up BNU on NPACK NETWORK (Basic)** (Procedure 9.6)
To show how BNU files are created for an NPACK NETWORK.

Section 9.7 **Set Up BNU on NPACK NETWORK (Special)** (Procedure 9.7)
To show how BNU files are created for a NPACK NETWORK if **cu** and **uucico** services are to be handled differently.

9.1 INSTALL BASIC NETWORKING SOFTWARE (Procedure 9.1)

Purpose	To place basic networking software on the hard disk.
Starting Conditions	System state—2 (multi-user) or 1 (single-user) You must be at the computer to insert and remove diskettes. You must **mount /usr** to run this procedure in single-user mode. Login—**root**
sysadm menu	SOFTWARE MANAGEMENT
Commands	**sysadm installpkg**(1)
Media	The basic networking utilities diskette.
Reference	*GENIX V.3 Networking Guide*

This procedure describes Basic Networking installation. The Basic Networking Utilities for the *Series 32000* Computer are distributed on two floppy diskettes. Most of the utilities are object code files. However, some are shell scripts that you can modify to suit your operating environment.

9.1.1 Run Sysadm Installpkg

Step 1: To install the Basic Networking Utilities, use the direct access method of the System Administration menu as follows:

 # sysadm installpkg

This executes the sysadm subcommand **installpkg.**

Step 2: Insert the floppy diskette labeled **1** and press **<CR>** as instructed. Your *Series 32000* Computer will display its current node name and ask if you want to define or rename it.

CAUTION

Changing the node name of your *Series 32000* Computer requires careful coordination with all machines that communicate with yours using BNU or other communications packages that rely on node name.

Once you have responded to this question, your *Series 32000* Computer will display the full path names of the files as they are copied from the floppy diskette to the hard disk.

Step 3: When installation of the first floppy is complete, insert the floppy diskette labeled **2,** and type **<CR>** as instructed. You will again see the filenames as they are copied to hard disk.

Step 4: If you have BNU configuration files from previous BNU versions, you will be asked if you want to keep these files. If you respond **n,** the install script saves the files from **/usr/lib/uucp** listed below in **/usr/lib/uucp/OLD.**

Devconfig
Devices
Dialcodes
Dialers
Maxuuscheds
Maxuuxqts
Permissions
Poll
Sysfiles
Systems
remote.unknown

If you respond **y,** the system will keep the old versions of these files from **/usr/lib/uucp** and discard the new ones.

NOTE: The **/usr/lib/uucp** files delivered with this release contain valuable information, including new entries in the **Dialers** file and notes on adding Transport Interface compatible devices. Therefore, it is recommended you respond **n** to this question, then edit the old information into the new files or use the new **Sysfiles** file for defining multiple BNU system files.

Once this step is complete, the system gives you an overview of the **sysadm** subcommands and then indicates that the installation is complete.

Step 5: Remove the second floppy diskette and store both floppies with your other utilities diskettes.

BNU installation is complete. You must now set up Basic Networking Utilities files as described in Procedure 9.2.

9.2 SET UP BASIC NETWORKING FILES (Procedure 9.2)

Purpose	To configure basic networking files. To ensure proper communications links.
When Performed	Initial setup and when adding new devices or remote systems.
Starting Conditions	System state—2 (multi-user) or 1 (single-user). You must **mount /usr** to run this procedure in single-user mode. Login—an authorized login
sysadm menu	PACKAGE MANAGEMENT
Commands	**sysadm uucpmgmt**(1) **sysadm devicemgmt**(1) **sysadm portmgmt**(1) **sysadm systemmgmt**(1) **sysadm pollmgmt**(1)
Reference	*GENIX V.3 Networking Guide*

The procedure that follows provides instructions for setting up the Basic Networking facility and putting it into operation. This is done using **sysadm** subcommands and a text editor.

The following steps provide instructions on adding entries to three of the necessary support files: **Devices, Systems,** and **Permissions.** Instructions are also provided to modify existing entries in the **/etc/inittab** file for use with Basic Networking. Finally, the set up of several optional files is described.

Display the **uucpmgmt** System Administration submenu by entering:

```
$ sysadm uucpmgmt
Password:
                        BASIC NETWORKING UTILITIES MANAGEMENT

  1 devicemgmt    manage devices (list, add, delete)
  2 pollmgmt      manage poll entries (list, add, delete)
  3 portmgmt      manage I/O ports (list, modify)
  4 systemmgmt    manage remote systems entries (list, add, delete, call)

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, q to QUIT:
```

Each Basic Networking subcommand described below can be accessed from this display or directly from the shell.

9.2.1 Set Up Devices File - devicemgmt

The **Devices** file (**/usr/lib/uucp/Devices**) contains information about the devices used to call other machines. For details on this file, refer to Chapter 10.

Step 1: To add entries into the **Devices** file, type **sysadm devicemgmt**, then select **2** (add):

```
$ sysadm devicemgmt

Running subcommand 'devicemgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

This procedure is used to list, add, and delete entries
in the Basic Networking Utilities '/usr/lib/uucp/Devices' file.
This file contains information about devices
available for calling out using the commands: uucp, cu, and ct.

Type q at any time to quit the present operation.
If a ? appears as a choice, type ? for help.

If a default appears in the question type <RETURN> for the default.

Enter the operation you want to perform:

    1 list
    2 add
    3 delete

(default list)[q]: 2
```

The subcommand will prompt you for information on the devices used by Basic Networking.

port name	The name of the port to which the device will be connected.
device name	The name of the device that is being connected to the above port. Pick the one you are using from the list that is displayed. The default is "penril." If ACU is specified as the device type to be connected to the port, two entries are created, one for 300 baud and one for 1200 baud.

Step 2: After you have entered the requested information, it will be displayed to you before it is entered into the **Devices** file.

The **/etc/inittab** file may not contain a correct entry for the port just assigned. You can change the port now or later using the **portmgmt** subcommand in the next procedure.

9.2.2 Set Up /etc/inittab - portmgmt

The **inittab** file (**/etc/inittab**) contains information on the ports to which the devices are connected. For further information on this file, refer to Chapter 4, Processor Operations.

Step 1: To add BNU entries to the **inittab** file type **sysadm portmgmt**, then select **2** (modify).

```
$ sysadm portmgmt

Running subcommand 'portmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

This procedure is used to list and modify
the entries that control the direction of traffic
on the Basic Networking Utilities I/O ports used by uucp, cu, and ct commands.

Type 'q' at any time to quit the present operation.
If a '?' appears as a choice, type '?' for help.

If a default appears in the question, type <RETURN> for the default.

Enter the operation you want to perform:

    1 list
    2 modify

(default list)[q]: 2
```

The subcommand lists the ports available to be used by Basic Networking, then asks you to choose which one you want to modify. It then prompts you for the following information:

port name	Name of the port you want to add (must be a port shown in the list).
traffic direction	The direction of the traffic on the port. You must specify whether the traffic will be **incoming** only, **outgoing** only, or **bidirectional.**
baud rate	Enter the speed (baud rate) of the selected port.

After you have entered the requested information, it will be displayed to you before it is entered into the **/etc/inittab** file.

NOTE: Since adding a device (**sysadm devicemgmt**) automatically creates a port entry in **/etc/inittab**, you may only need to use **sysadm portmgmt** for modifications.

9.2.3 Set Up Systems File - systemmgmt

The **Systems** file (**/usr/lib/uucp/Systems**) contains the information needed by **uucp** to call and log on to a remote machine. Each entry represents one remote machine that can be called by your Basic Networking programs.

If the **Systems** entry is to be used to contact a machine that is hardwired to your *Series 32000* Computer, refer to Chapter 10 for special instructions on setting up the **Systems** file.

Step 1: To add other machines to your **Systems** file, type **sysadm systemmgmt**, then select **2**.

```
$ sysadm systemmgmt

Running subcommand 'systemmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

This procedure is used to list, add, and delete entries in the
Basic Networking Utilities '/usr/lib/uucp/Systems' file.
This file contains information about what remote systems
can be called by cu and uucp commands.
You can also try to call a remote system that appears
in the '/usr/lib/uucp/Systems' file.

Type 'q' at any time to quit the current operation.
If a '?' appears as a choice, type '?' for help.

If a default appears in the question, type <RETURN> for the default.

Enter the operation you want to perform:

    1 list
    2 add
    3 delete
    4 call

(default list)[q]: 2
```

After you select **2** (add), the subcommand will prompt you for the following information.

node name	Node name of the system you want to call.
device type	Type of device used to establish connection (for example, acu).
baud rate	The speed at which the device will place the call.
phone number	The telephone number of the remote machine. Special symbols can be embedded in the phone number, including abbreviations from the **Dialcodes** file (**/usr/lib/uucp/Dialcodes**).
login ID	Used by **uucp** to login on the remote machine.
password	The password associated with the above login.

Step 2: After you have entered the requested information it will be displayed to you before it is entered into the **Systems** file.

9.2.4 Set Up Poll File - pollmgmt

The **Poll** file (**/usr/lib/uucp/Poll**) contains a list of the machines that are to be called (polled) by your *Series 32000* Computer to see if they have anything to transmit to you. It also contains the times they are to be polled.

Step 1: To add entries to the **Poll** file, type **sysadm pollmgmt**, then select **2** (add).

```
$ sysadm pollmgmt
Running subcommand 'pollmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT

This procedure is used to list, add, and delete entries in the
Basic Networking Utilities '/usr/lib/uucp/Poll' file.
This file contains information about what systems and
the times (hours) the systems should be polled.

Type 'q' at any time to quit the current operation.
If a '?' appears as a choice, type '?' for help.

If a default appears in the question, type <RETURN> for the default.

Enter the operation you want to perform:

    1 list
    2 add
    3 delete

(default list)[q]:
```

The **pollmgmt** subcommand prompts you for the following information:

system name	Name of the system you want to poll.
polling hours	The hours you want to poll the system; must be an integer number between 0 and 23 (for example, 0 4 8 12 16 20 is every four hours).

Step 2: After you have entered the requested information it will be displayed to you before it is entered into the **Poll** file.

9.2.5 Set Up Permissions File

The default **/usr/lib/uucp/Permissions** file provides the maximum amount of security for your *Series 32000* Computer. The file, as delivered, contains the following entry:

LOGNAME=nuucp

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

See Chapter 10 for information on how to set up this file. If you want to change the contents of this file, you must edit it to modify the file and make the entries you desire.

9.2.6 Set Up Devconfig File

The **/usr/lib/uucp/Devconfig** file is only needed if you are using BNU over an NPACK NETWORK or some other Streams-based transport provider that conforms to the Transport Interface (TI). If you are using an NPACK NETWORK, the two entries shown in the **Devconfig** file are all you need in this file.

```
service=cu     device=NPACK  push=ntty:tirdwr:ld0
service=uucico device=NPACK  push=ntty:tirdwr:ld0
```

Remove the comment character (#) in front of each of these lines to activate them. You must also create an entry for NPACK in your **Devices** file. Descriptions in the **Devices** file tell how to define Transport Interface devices. (See Procedure 9.6 for a complete example of setting up BNU on NPACK.)

Devconfig entries define the STREAMS modules that are used for a particular TI device. (The **push=** variable shows the modules and the order they are pushed on to a stream.) Different modules and devices can be defined for **cu** and **uucp** services. If you want to change the contents of this file, you must use one of the editors (**ed** or **vi**) to modify the file and make the entries you desire.

9.2.7 Set Up Sysfiles File

/usr/lib/uucp/Sysfiles 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 **cu** login services can be made to addresses other than **uucp** services.
- You may want different **Dialers** files to use different chat scripts 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 convenient to split it into several smaller files.

The format of the **Sysfiles** file is described in Chapter 10. The following is an example of the file.

```
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
```

If you want to change the contents of this file, you must use one of the editors (**ed** or **vi**) to modify the file and make the entries you desire.

9.2.8 Add Uucp logins

You must add one or more administrative logins to your system so incoming **uucp** (**uucico**) requests from remote machines can be handled properly. Each remote machine should have an entry in its **Systems** file for your machine that contains the login ID and password that you add to your **/etc/passwd** file.

An example of a common entry in the **/etc/passwd** file is shown below.

nuucp:????:6:1:UUCP.Admin:/usr/spool/uucppublic:/usr/lib/uucp/uucico

This entry shows that a login request by **nuucp** is answered by **/usr/lib/uucp/uucico.** The home directory is **/usr/spool/uucppublic.** The ???? will be replaced by an encrypted password that would be added using **passwd nuucp.**

9.2.9 Other Networking Files

There are three other files that affect the use of Basic Networking facilities. In most cases, the default values are fine and no changes are needed. If you want to change them, however, use any standard GENIX V.3 system editor (**ed** or **vi**).

Maxuuxqts This file defines the maximum number of **uuxqt** programs that can run at once.

Maxuuscheds This file defines the maximum number of **uusched** programs that can run at once.

remote.unknown This file is a shell script that executes when an unknown machine 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, your system will accept any conversation requests.)

9.3 BASIC NETWORKING MAINTENANCE (Procedure 9.3)

Purpose	To keep files related to basic networking from consuming too much disk space.
When Performed	Automatically with **cron**(1M) or as needed.
Starting Conditions	System state—2 (multi-user) or 1 (single-user)

Basic Networking Utilities 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 your basic networking running smoothly. Normally, they are run automatically with **cron**(1M), though they can also be run manually. The few areas needing clean up that are not handled by these shell scripts should be maintained manually.

9.3.1 Automated Networking Maintenance (cron)

The Basic Networking Utilities are delivered with entries for **uudemon** shell scripts in the **/usr/spool/cron/crontabs/root** file. These entries will automatically handle some BNU administrative tasks for you. Each of these shell scripts is in **/usr/lib/uucp.**

When the *Series 32000* Computer is in run state 2 (multi-user), **cron** scans the **/usr/spool/cron/crontabs/root** file every minute for entries scheduled to execute at that time. As the UUCP administrator, you should become familiar with **cron** and the four **uudemon** shell scripts.

9.3.2 Uudemon.poll

The **uudemon.poll** shell script, as delivered, does the following:

- Reads the **Poll** file (**/usr/lib/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 **/usr/spool/uucp/***nodename* directory, where *nodename* is replaced by the name of the machine.

The shell script is scheduled to run twice an hour just before **uudemon.hour** so that the work files will be there when **uudemon.hour** is called. The default root crontab entry for **uudemon.poll** is as follows:

1,30 * * * * "/usr/lib/uucp/uudemon.poll > /dev/null"

9.3.3 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 *Series 32000* Computer and were not processed at the time they were transferred.

The default root crontab entry for **uudemon.hour** is as follows:

41,11 * * * * /usr/lib/uucp/uudemon.hour > /dev/null

As delivered, this is run twice an hour. You may want it to run more often if you expect high failure rates.

9.3.4 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 (**/usr/spool/locks**).
- Sends resulting status information to the **uucp** administrative login via mail.

There is no default entry **/usr/spool/cron/crontabs/root** for **uudemon.admin.** The following is recommended:

48 8,12,16 * * * /bin/su uucp —c "/usr/lib/uucp/uudemon.admin" > /dev/null

9.3.5 Uudemon.cleanup

The delivered **uudemon.cleanu** shell script does the following:

- Takes log files for individual machines from the **/usr/spool/uucp/.Log** directory, merges them, and places them in the **/usr/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**).

No default root crontab entry for **uudemon.cleanup** is delivered. This is a recommended entry.

45 23 * * * ulimit 5000; /bin/su uucp —c "/usr/lib/uucp/uudemon.cleanup" > /dev/null 2>&1

9.3.6 Manual Maintenance

Some files may grow indirectly from **uucp** and other basic networking activities. Here are two files you should check and delete if they have become too large.

/usr/adm/sulog This file keeps a history of all super user commands. Since the uudemon entries in the **/usr/cron/root** file use the **su** command, the **sulog** will grow over time. You should delete this file if it becomes too large.

/usr/lib/cron/log This file is a log of cron activities. While it grows with use, it is automatically truncated when the system goes to the multi-user state.

9.4 BASIC NETWORKING DEBUGGING (Procedure 9.4)

Purpose	To use available monitoring tools to solve basic networking problems.
Starting Conditions	System state—2 (multi-user) or 1 (single-user)
Commands	**uustat**(1) **cu**(1) **Uutry**(1) **uuname**(1M) **uulog**(1) **uucheck**(1)

These procedures describe how to go about solving common problems that may be encountered with Basic Networking Utilities.

9.4.1 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 —l***line*. 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**.)

9.4.2 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

9.4.3 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:

1. Start the transfer daemon (**uucico**) with debugging. You will get more debugging information if you are **root.**
2. Direct the debugging output to /tmp/*machine*,
3. Print the debugging output to your terminal (**tail —f**). Hit **BREAK** 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.

9.4.4 Check Error Messages

There are two types of error messages for Basic Networking Utilities: ASSERT and STATUS. See Appendix C for a listing of these messages.

ASSERT Error Messages

When a process is aborted, ASSERT error messages are recorded in **/usr/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 **/usr/spool/uucp/.Status** directory. The directory contains a separate file for each remote machine with which your *Series 32000* Computer attempts to communicate. These files contain status information on the attempted commmunication and whether it was successful.

9.4.5 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.

9.5 REMOVE BNU SOFTWARE (Procedure 9.5)

Purpose	To remove the software from the hard disk
Starting Conditions	System state—2 (multi-user) or 1 (single-user) You must be at the computer to insert and remove diskettes. Login—**root**
sysadm menu	SOFTWARE MANAGEMENT
Commands	**sysadm removepkg**(1)
Media	The basic networking utilities diskette.

This procedure describes how to remove Basic Networking Utilities software from your *Series 32000* Computer. Since removing Basic Networking Utilities leaves you without a means of communicating with other computers, this should be a last resort for freeing up disk space.

NOTE: Removing BNU Utilities will turn off the **inittab** entry for every device in **/usr/lib/uucp/Devices** and for all devices in **inittab** that are running **uugetty.**

9.5.1 Prerequisites

Before you remove the Basic Networking Utilities, be sure to do the following:

Step 1: When the software is removed, many support files will be lost. Therefore, be sure to save the information in the following files on paper or floppy disk.

/usr/lib/uucp/Devices
/usr/lib/uucp/Dialcodes
/usr/lib/uucp/Dialers
/usr/lib/uucp/Permissions
/usr/lib/uucp/Poll
/usr/lib/uucp/Systems
/usr/lib/uucp/Sysfiles
/usr/lib/uucp/Devconfig
/usr/spool/uucppublic/*

Step 2: Before you remove Basic Networking Utilities, you should be logged in as **root** at the CONSOLE terminal.

9.5.2 Single-User Mode

Step 1: Have all terminals except the CONSOLE terminal log off.

Step 2: Place the machine in single-user mode by typing:

```
# shutdown —y —g0 —i6
```

(Note: A series of shutdown messages appear, indicating that the system is coming down.)

9.5.3 Run sysadm removepkg

Step 1: The following command will let you use System Administration menu subcommands while in the single-user mode.

```
# mount /usr
```

Step 2: To remove the Basic Networking Utilities, use the direct access method of the System Administration menu as follows:

```
# sysadm removepkg
```

Step 3: Insert the first BNU floppy diskette and press <**CR**> as instructed. The system will display the full path names of the files that are being removed from the hard disk.

9.5.4 Return to Multi-User Mode

Step 1: Unmount the **/usr** file system, leaving only the root file system mounted.

```
# umount /usr
```

Step 2: To return to the normal operating condition, enter:

```
# init 2
```

Your *Series 32000* Computer is available for use with the exception of the Basic Networking Utilities.

9.6 SET UP BNU NPACK NETWORK (Basic) [Procedure 9.6]

Purpose	To show how to set up BNU files for an NPACK NETWORK using **sysadm.**
Starting Conditions	System state—2 (multi-user) or 1 (single-user) Login—**root**
Commands	**sysadm uucpmgmt**(1)

This procedure contains an example of setting up BNU files for an NPACK NETWORK. Though the example is particular to setting up BNU on an NPACK NETWORK, the concepts could be applied to setting up BNU to run on any transport provider that is compatible with the Transport Interface.

The first part of this procedure shows how to set up BNU files so that **uucp** and **cu** requests both go through login to connect to machines on the network. The second part shows what you must change so **uucico** services can connect to hosts without going through login. Since the second part could present security problems, you should only make this type of service available when your NPACK NETWORK consists of trusted machines.

9.6.1 Prerequisites

Before running this procedure to set up BNU to run on an NPACK NETWORK, you must install these packages in the order shown:

- Networking Support Utilities
- NPACK NETWORK
- Basic Networking Utilities

9.6.2 Create NPACK Systems Entry

Repeat the following steps for each machine (node name) on the network you want to communicate with.

Step 1: Type **sysadm uucpmgmt** to display the Basic Networking Utilities Management menu.

Step 2: Select **systemmgmt** to display the list of operations you can perform on the **Systems** file.

Step 3: Type the number required to choose the **add** function.

Step 4: Type the node name for the remote machine you want to communicate with over the NPACK NETWORK. (The convention is to use node name. However, some networks some other means of identifying another machine.)

Step 5: Type number required to select TLI (Transport Layer Interface) as the device type.

Step 6: Type **NPACK** to choose that name as the network name. (The word **NPACK** need not be used. You can choose any name to represent the network. The key is that

this name must match the network name that goes in the caller type field in the **Devices** file. For consistency, NPACK will be used when the **Devices** file entry is made in this procedure.)

Step 7: Type the network address used for connection to the remote machine. (The recommended convention is to use node name as the NPACK NETWORK address of the remote machine for login services.)

Step 8: Type **nuucp** as the login ID that is used when calling the remote machine. (This is a conventional login to use, though others can be used.)

Step 9: Type a password if one is required by the remote machine.

Step 10: Type **y** to place the entry in the **Systems** file.

Step 11: Type **y** if you want to add another system, then repeat steps 4 through 11. Type **n** if you don't want to add any more systems.

Step 12: Type **q<CR>** three times to return to the shell.

9.6.3 Create NPACK Devices Entry

Step 1: Type **sysadm uucpmgmt** to display the Basic Networking Utilities Management menu.

Step 2: Select **devicemgmt** to display the list of operations you can perform on the **Devices** file.

Step 3: Type the number required to choose the **add** function.

Step 4: Type **npack** as the name of the port. (In this case you must use the literal **npack**. It indicates the device name, relative to the **/dev** directory.)

Step 5: Select TLI network to indicate that the port is connected to this type of network.

Step 6: Type **NPACK** as the network name. (This is the Caller Type field. By using different names here you can have multiple TLI networks on your system. This name must match the network name added to the **Systems** file entry.)

Step 7: Type **y** to add the entry to the **Devices** file.

Step 8: Type **q<CR>** three times to return to the shell.

9.6.4 Create NPACK Devconfig Entries

The **/usr/lib/uucp/Devconfig** file entries must be created to define the **STREAMS** modules to **push** for **cu** and **uucico** services. There is no **sysadm** menu for this file, so you should edit the file to make sure the following two entries are included:

```
service=cu      device=NPACK  push=ntty:tirdwr:ld0
service=uucico  device=NPACK  push=ntty:tirdwr:ld0
```

The two lines are already in the **Devconfig** file. To activate them, remove the comment character (#) in front of each line.

9.6.5 Set Up NPACK NETWORK Listener

NPACK and other TLI networks running on the GENIX V.3 system each has a separate network listener process associated with it. The listener process "listens" to the network for service requests, accepts requests when they arrive, and spawns servers in response to those service requests.

Nlsadmin administers the network listener process(es) on a machine. **Nlsadmin** can establish a listener process for a given network, configure the specific attributes of that listener, and start and kill the listener process for that network. (See **nlsadmin**(1M) for more information.)

Chances are, you already started a listener that was configured to provide standard login services when you installed NPACK NETWORK. The following steps show you how to check that NPACK NETWORK is set up properly to support BNU and what to enter in case it isn't.

Step 1: Type:

nlsadmin —v npack

You should see:

```
  1   ENABLED   NOMODULES   /usr/npack/lib/ttysrv   comment
102   ENABLED   NOMODULES   /usr/npack/lib/ttysrv   comment
```

If you don't see the above, then type:

```
# nlsadmin —a 1 —c "/usr/npack/lib/ttysrv" —y "tty login service" —m npack
# nlsadmin —a 102 —c "/usr/npack/lib/ttysrv" —y "tty login service" npack
```

Step 2: Type:

nlsadmin —x

You should see:

npack ACTIVE

If you see:

npack INACTIVE

then type:

nlsadmin —s npack

to start it up. If you don't see either **npack ACTIVE** or **npack INACTIVE,** then the NPACK NETWORK was not properly installed.

9.6.6 NPACK Dialers Entry

You do not need to create an entry in the **Dialers** file for standard BNU services over NPACK as shown in the previous procedure.

BNU is now ready to communicate over an NPACK network.

9.7 SET UP BNU NPACK NETWORK (Special) [Procedure 9.7]

Purpose	To show how to set up BNU files for an NPACK NETWORK to handle **uucico** and **cu** services differently.
Starting Conditions	System state—2 (multi-user) or 1 (single-user) Login—**root**
Commands	Any text editor [**ed**(1), **vi**(1)]

In the previous procedure, both **uucico** and **cu** services go through a login procedure to connect to other computers on an NPACK NETWORK. To improve performance of **uucico** services, you can have **uucico** requests connect directly to remote machines on the NPACK NETWORK without going through login.

You should only set up **uucico** as described above if you trust all machines on your NPACK NETWORK. The reason is that there is a potential security breach since you would not be requesting a password when a machine tries to transfer files to your machine.

Before you run this procedure you should run the previous procedure so **cu** services are handled correctly.

Step 1: You must define separate **Systems, Devices,** and **Dialers** files to handle **uucico** services. This is done in the **Sysfiles** file. The following is an example.

```
service=cu      systems=Systems \
                devices=Devices \
                dialers=Dialers
service=uucico  systems=Systems.cico:Systems \
                devices=Devices.cico:Devices \
                dialers=Dialers.cico:Dialers
```

In the example, **cu** services would use the standard BNU files. **uucico** services, however, would use the standard BNU files only after checking **Systems.cico, Devices.cico,** and **Dialers.cico.** You can use any filenames you want. The ones shown above are only given as an example.

Step 2: The systems you want to transfer files to on your NPACK NETWORK without going through login must be listed in the new **Systems** file you defined in the **Sysfiles** file.

If you used the file names defined above in the **Sysfiles** file, you would edit **/usr/lib/uucp/Systems.cico** and add entries similar to the following entry for each system.

abc Any ListenCico - abc.serve

In this example, **abc** is the system name, **Any** says use the highest baud rate possible, **ListenCico** points to a device type in the **Devices.cico** file, the dash is a field place holder, and **abc.serve** is the NPACK NETWORK address of **abc.**

Step 3: Continuing with the example above, from the entry in the **Systems.cico** file, you must add an entry to the **/usr/lib/uucp/Devices.cico** file called **ListenCico.** This entry should look like the following:

ListenCico,eg npack - - TLI \D listencico

In this example, **ListenCico** is the device name, **npack** identifies the network, the two dashes are field place holders, **TLI** is the network type, **\D** says read in the network address (**abc**), and **listencico** is the dialer type.

NOTE: You do not need an entry in the **Devconfig** file for **ListenCico** since you will not be pushing any Streams modules.

Step 4: From the entry in the **Devices.cico** file, you must add an entry to the **/usr/lib/uucp/Dialers.cico** file called **listencico.** The following is an example of what you should enter.

listencico"""" NLPS:000:001:101\N\c

Step 5: You must register the **uucico** service with the network listener. The command shown below must all be typed on one line.

 # nlsadmin —a 101 —c "/usr/lib/uucp/uucico —r 0 —i TLI —u \ nuucp" —y "uucico server with NO login checking" npack

This entry says to answer requests for service code 101 with **uucico** directly as it is used for a **TLI** (Transport Interface) network. Notice that the default **nuucp** login ID is used; you can use a different one if you choose.

Procedure 10

REMOTE FILE SHARING PROCEDURES

The following procedures are covered in these sections.

Section 10.1 **Domain Setup (primary)** (Procedure 10.1)
To set up the Remote File Sharing primary name server for the domain.

Section 10.2 **Host Setup** (Procedure 10.2)
To set up a host computer and connect it to the Remote File Sharing domain.

Section 10.3 **Advertise Resources** (Procedure 10.3)
To make your directories available to other hosts.

Section 10.4 **Mount Remote Resources** (Procedure 10.4)
To make remote directories available to your host.

10.1 DOMAIN SETUP (primary) [Procedure 10.1]

Purpose	To set up the primary domain name server for a Remote File Sharing Domain.
Starting Conditions	System state—2 (multi-user) or 1 (single-user) You must **mount /usr** to run this procedure in single-user mode. Login—**root**
Commands	**uname**(1M) **nlsadmin**(1M) **dname**(1M) **rfadmin**(1M) **rfstart**(1M) **adv**(1M)
Reference	Chapter 11, Remote File Sharing

Before beginning this procedure, read Chapter 11, "Remote File Sharing." It will help you understand what Remote File Sharing is and how you can use it. This procedure describes how you should set up your Remote File Sharing domain. Domain setup is split into Basic Domain Setup and Optional Domain Setup. If you simply want to add your host to an existing domain, skip to the Host Setup procedure.

10.1.1 Basic Domain Setup

The following procedure contains the minimum you need to do to get your domain up and running.

Prerequisites

Before you begin setting up your domain, you must do the following.

- You must pick one or more computers on the network to act as domain name servers. Exactly one primary is required. All domain administration is done from the primary. You can choose zero or more secondary domain name servers. These are defined simply to keep the name server running temporarily, should the primary fail.
- GENIX V.3 System software, Remote File Sharing software, Networking Support Utilities, and STREAM-based PACK protocol and utility must be installed and running. (See the *Remote File Sharing Utilities Notes* and the network manuals that accompany the product for installation instructions.)

  NPACK NETWORK can be used as the Remote File Sharing network (transport provider) for the first release. Other networks could be used, however, if they conform to the Transport Interface specification.
- Login as **root**. All domain administration must be done with **root** permission from the primary domain name server.

Configuration

You must run the following steps on the computer you choose to be the primary domain name server.

Step 1: Check to see if your computer's node name is set (**uname —n**). If it's not, set it by typing:

 # sysadm nodename

You will be asked to type in your host's node name. A Remote File Sharing node name can consist of up to 14 characters of letters (upper- and lowercase), digits, hyphens (-), and underscores (_). Some networks, such as the NPACK NETWORK, require that every node name in the network be different. Remote File Sharing, however, only requires that every node name in a domain be different.

CAUTION

Changing the node name of your *Series 32000* Computer requires careful coordination with all machines that communicate with yours using Remote File Sharing or other communications packages that rely on node name.

Step 2: Set up the network listener. If you have installed the NPACK NETWORK and Remote File Sharing in the order described in the *Remote File Sharing Utilities Notes*, you can skip this step. The listener will already be installed and set up to run automatically, and Remote File Sharing will be listed as an available service.

If you are using another Transport Interface-compatible network, or suspect that your NPACK NETWORK listener is improperly set up this step shows how to manually set up the listener. In the following example NPACK NETWORK is used. To set up the listener for other networks compatible with the AT&T Transport Interface, you would replace **npack** with the name of the network (*net_spec*) you are installing. (For more details, see the **nlsadmin**(1M) manual page).

If you run any of these commands and they have already been run, you will receive a message telling you so. This won't harm your listener configuration. Type:

nlsadmin —i npack

to initialize the files needed for the listener process for the network specified, in this case **npack**.

Next type (on one line):

nlsadmin —a 105 —c /usr/net/servers/rfs/rfsetup —y "rfsetup" npack

to add the Remote File Sharing service (**rfsetup**) to the list of services available to the **npack** listener.

Use the following command line to report the status of the npack listener process installed on this machine (ENABLED or DISABLED):

nlsadmin —x npack

Next type:

nlsadmin —l "*listeneraddr*" **—t** "*terminaladdr*" **npack**

to register the network addresses of your machine the listener will listen for on the network. (Other networks will use other types of network addresses. See the description of **rfmaster**(1M) for the syntax of network address.)

To start the listener, type:

nlsadmin —s npack

Normally, it will be started automatically when your machine enters multi-user mode (**init 2**).

Step 3: Set the domain name by typing:

 # dname —D *domain*

where *domain* is replaced by the domain your host will be a member of. The domain name must:

- contain no more than 14 characters
- consist of any combination of letters (upper- or lowercase), digits, hyphens, and underscores

- be different from every other domain name on your network

Step 4: The default transport provider is NPACK. This is automatially done for you if you have installed networking utilities properly and your system is in multi-user mode (level 2). Type:

dname —n

to see what the current transport provider's name is. If it is not set correctly, type:

dname —N npack

Step 5: You create an **rfmaster** file in the **/usr/nserve** directory, using any standard file editor. The contents of this file will define:

- the primary name server for your domain
- secondary name servers for your domain
- network addresses for each of these machines

Here is an example of an **rfmaster** file for a domain called **graph**, whose primary and secondary name servers' node names are **fee** and **fie**. Adding each machine's domain name (**graph**) to its node name, separated by a period, forms its full Remote File Sharing host name. Each line of the example translates as follows.

- For domain **graph** the primary is **graph.fee.**
- For domain **graph** a secondary is **graph.fie.**
- For host **graph.fee** the network address is **\X060000000800140050660000.**
- For host **graph.fie** the network address is **\X060000000800141053170000.**

(The addresses shown are an example of NPACK NETWORK addresses. These addresses should be in the form **\X0m0000000XXXXXXXXXXXX0000**, where *X* specifies a hexadecimal address, *m* is the minor device that the listerner will use, and *XXXXXXXXXXXX* is a hexadecimal value that usually corresponds to the machine's physical Ethernet address. Default minor device numbers for listener and terminal are 6 and 7 respectively.

```
graph       p   graph.fee
graph       s   graph.fie
graph.fee   a   \X060000000800140050660000
graph.fie   a   \X060000000800141053170000
```

Each line in the example is an entry. The second field is the *Type* field, which indicates whether the entry defines a primary name server (**p**), secondary name server (**s**), or the network address (**a**) for one of the two.

Here is the information needed for the first field, *Name*, and the third field, *Rdata*, for each type of entry.

p Primary entry. *Name* is the domain name. *Rdata* is the full host name of the domain's primary name server (*domain.nodename*).

s Secondary entry. *Name* is the domain name. *Rdata* is the full host name of the domain's secondary name server (*domain.nodename*).

a Address entry. *Name* is the full host name (*domain.nodename*) of a name server host. *Rdata* is the network address of the host. The manuals that come with your network should describe how to find a computer's network address.

Here are some special considerations when creating the file.

- Fields in each entry must be separated by one blank or one tab. (Do not use multiple white space characters in between fields or the entry will be invalid.)
- The address must be in plain ASCII text or hexadecimal notation. For hexadecimal, the field must begin with \x and contain an even number of digits. If the address contains white space, the field must be surrounded by double quotes (").
- An entry can extend beyond one line if you enter a back slash, then a carriage return to continue to the second line.
- This file should be write protected from all but **root**, but all read permissions should be enabled (644 permissions).
- If you start a line with the # character, the entire line will be treated as a comment.

Step 6: To add each host to the domain, use the following command:

```
# rfadmin -a domain.nodename
Enter password for nodename:
Re-enter password for nodename:
```

where *nodename* is replaced by the node name of the host you want to add to your *domain*. (The two names must be connected by a period.)

You will be prompted for an initial password, which will be stored in the **/usr/nserve/auth.info/***domain***/passwd** file for the *domain*. When the host starts Remote File Sharing, the host's administrator must enter this password. You can simply **<CR>** for a null password. Repeat this command for each host you want to add to the domain.

NOTE: All primary and secondary hosts must be added to a domain before you can proceed to the next step.

Step 7: Type the following command to start Remote File Sharing on your machine:

 # rfstart
rfstart: Please enter machine password:

You will be prompted for a password; it must match the password you entered for your host using **rfadmin —a.** Your host will save this password, so you do not have to enter it again. Once your password is entered, the next time you start Remote File Sharing you will be able to do it via **init 3.**

The **init 3** command runs a series of scripts to automatically enter Remote File Sharing mode. These include running the **rfstart** command, offering your resources to other machines (advertising), and mounting remote resources on your machine. (See Section 11.4.4 in Chapter 11 for more information on **init 3.**)

BASIC DOMAIN SETUP IS COMPLETE.

You can now either:

- Go to the Host Setup section and setup each host.
- Continue through the Optional Domain Setup.

10.1.2 Optional Domain Setup

The steps in this section describe optional procedures to:

- advertise the domain's host password and user and group information
- collect user and group information from hosts
- communicate with other domains on your network

Each step described below is independently optional.

Step 1: To make domain information available to other hosts, you can *advertise* your domain's /usr/nserve/auth.info/*domain* directory. The other hosts can then have access to:

- the domain's host password file, to use for host verification
- each host's user and group files, to map remote users and groups by name (optional)
  (Step 2 describes how to collect this information.)

Here is an example of advertising domain information for a domain called **graph.**

adv —r —d "graph info" GRAPH /usr/nserve/auth.info/graph

This example advertises a resource:

- read only (—**r**)
- with a description of **graph info** (—**d "graph info"**)
- a resource name of **GRAPH**
- consisting of the **/usr/nserve/auth.info/graph** directory from your computer

Because no clients are listed on the command line, any host with access to domain **graph** will be able to mount the resource. Notice that the resource name is in all capitals. This is a recommended convention for adding resource names, but it is not required.

See Chapter 10 for information on where to put this **adv** command so the resource is automatically advertised when the system enters **init 3** state.

Step 2: Collect user and group information. The primary domain name server provides a central place for storing **passwd** and **group** files from each host. These files will go into the **/usr/nserve/auth.info/***domain/host* directories for each *host* in each *domain*. (See Section 11.3 in Chapter 11 to help decide if you want to collect user and group lists.)

The Host Setup procedure describes how each host will create these files. You can collect these files in one of two ways:

- have each host mail them to the primary name server and place them in the proper directories
- leave the permissions open to administrators of remote hosts on each remote host when you advertise your **/usr/nserve/auth.info/***domain* directory, so each host can put this information in itself. The resource must be advertised read/write to be mounted read/write.

Step 3: For hosts in your domain to communicate with other domains on your network, you must find out:

- the primary name server for each domain
- the secondary name server(s) for each domain
- the network address for each of the above name servers

You must then add this information to your domain's **rfmaster** file on the primary, just as you added information for your own domain. See the description of the **rfmaster**(4) file for more information. After you have run the **rfstart** command, this information will be picked up by all hosts in your domain the next time each does an **rfstart**.

10.2 HOST SETUP (Procedure 10.2)

Purpose	To set up a host on a Remote File Sharing Domain.
Starting Conditions	System state—2 (multi-user) or 1 (single-user) You must **mount /usr** to run this procedure in single-user mode. Login—**root**
Commands	**uname**(1M) **nlsadmin**(1M) **dname**(1M) **rfstart**(1M) **mount**(1M)
Reference	Chapter 10, Remote File Sharing

This section contains configuration and startup procedures required for every host computer on a Remote File Sharing network. If the host is the domain name server host, you can skip over the steps you have already done. Host setup is split into Basic Host Setup and Optional Host Setup.

10.2.1 Basic Host Setup

The following procedure contains the minimum you must do to connect your host to a Remote File Sharing system.

Prerequisites

Before you begin setting up your host, you must meet the following requirements.

- Domain must be created. Your host must be defined as a member of a domain on the domain's primary name server. The primary must also be running when you first start Remote File Sharing (**rfstart**(1M) command). See Procedure 10.1, Domain Setup.
- Install software. GENIX V.3 System software, Remote File Sharing software, Networking Support Utilities, and network software must be installed and running. (See the *Remote File Sharing Utilities Notes* and the network manuals that accompany the product for installation instructions.)

  NPACK NETWORK can be used as the Remote File Sharing network (transport provider) for the first release. Other networks could be used, however, if they conform to the Transport Interface specification.
- Login as **root**. All host administration must be done with **root** permission on your host.

Configuration

This procedure should be run on every host in a domain that is not a primary domain name server.

Step 1: Check to see if your computer's node name is set (**uname —n**). If it's not, set it by typing:

 # sysadm nodename

You will be asked to type in your host's node name. A Remote File Sharing node name can consist of up to 14 characters of letters (upper- and lowercase), digits, hyphens (—), and underscores (_). Some networks, such as NPACK NETWORK require that every node name in the network be different. Remote File Sharing, however, only requires that every node name in a domain be different.

CAUTION

Changing the node name of your *Series 32000* Computer requires careful coordination with all machines that communicate with yours using Remote File Sharing or other communications packages that rely on node name.

Step 2: Set up the network listener. If you have installed the NPACK NETWORK and Remote File Sharing in the order described in the *Remote File Sharing Utilities Notes*, you can skip this step. The listener will already be installed and set up to run automatically, and Remote File Sharing will be listed as an available service. If you are using another Transport Interface-compatible network, or suspect that your NPACK NETWORK listener is improperly set up, this step shows how to manually set up the *listener*.

In the following example NPACK NETWORK is used. To set up the listener for other networks compatible with the Transport Interface, you should replace **npack** with the name of the network (*net_spec*) you are installing. (For more details, see the **nlsadmin**(1M) manual page).

If you run any of these commands and they have already been run, you will receive a message telling you so. This won't harm your listener configuration.

Type:

nlsadmin —i npack

to initialize the files needed for the listener process for the network specified, in this case **npack.**

Next type (on one line):

nlsadmin —a 105 —c /usr/net/servers/rfs/rfsetup —y "rfsetup" npack

This will add the Remote File Sharing service (**rfsetup**) to the list of services available to the **npack** listener.

nlsadmin —x npack

will report the status of the npack listener process installed on this machine.

Next type:

nlsadmin —l "*listeneraddr*" **—t** "*terminaladdr*" **npack**

to register the network addresses of your machine the listener will listen for on the network. (Other networks will use other types of network addresses. See the description of **rfmaster**(1M) for the syntax of network address.)

To start the listener, type:

nlsadmin —s npack

Normally, it will be started automatically when your machine enters multi-user mode (**init 2**).

Step 3: Set the domain name by typing:

 # dname —D *domain*

where *domain* is replaced by the domain your host will be a member of. The domain name must:

- contain no more than 14 characters
- consist of any combination of letters (upper- or lowercase), digits, hyphens, and underscores
- be different from every other domain name on your network

Step 4: You must tell Remote File Sharing the network (transport provider) it should use. (In our example, this is **npack.**)

 # dname —N npack

Step 5: To start Remote File Sharing on your machine, type the following command.

rfstart —p *primary_ns_address*
rfstart: Please enter machine password:

The *primary_ns_address* is replaced by the network address of the primary name server for your domain.

You will be prompted for a password; it must match the password entered at the primary domain name server when your host was added to the domain. Your host will save this password, so you do not have to enter it again.

Likewise, your machine will save the network address of the primary name server. Therefore, the next time you start up Remote File Sharing, you will be able to do it via **init 3.**

The **init 3** command runs a series of scripts to automatically enter Remote File Sharing mode. These include running the **rfstart** command, offering your resources to other machines (advertising), and mounting remote resources on your machine. (See Section 11.4.4 in Chapter 11 for more information on **init 3.**)

BASIC HOST SETUP IS NOW COMPLETE.

If you don't want any special security measures or user mapping you can skip the rest of this section. This is the security your computer currently has:

- You will be able to mount any resources available to your domain that are either unrestricted or made specially available to your host. The permissions your users will have to the resource will depend on how the other host sets its permissions.
- Any resources you advertise can be left unrestricted or can be restricted to certain hosts or domains. Users from the other hosts will be mapped into a special guest login. The special guest login is guaranteed to be different from all user IDs on your system. This will give them minimal permission to files and directories created by your users.

The rest of this section describes how to further control host and user access to your host's resources.

10.2.2 Optional Host Setup

The following steps are only needed if you want to verify hosts that try to connect to your host, or if you want to map remote users into your host by name. Each step is independently optional.

Step 1: The primary name server for your domain keeps records of host passwords and user and group information. These are stored in one set of directories for your domain and made available to you over the Remote File Sharing network under the resource name *DOMAIN*, where *DOMAIN* is the name of your domain in all capital letters. (The primary must have advertised this resource before you can mount it.)

You should mount this resource in another location and copy the directory structure into **/usr/nserve/auth.info/***domain* directory, where *domain* is your literal domain name. You can unmount the resource once you have copied the information. If a host changes its password or user and group information, however, you must either copy in the change files yourself or set up a program to automatically copy over changed files from the primary domain name server.

Below is an example of how to mount and copy the information. (Replace *domain* with the name of your domain and *DOMAIN* with the name of your domain in all caps. The **mkdir** will not be necessary if the directory already exists.)

```
# mkdir /mnt
# mount —r —d DOMAIN /mnt
# cd /mnt
# find . —depth —print | cpio —pd /usr/nserve/auth.info/domain
# cd /
# umount —d DOMAIN
```

NOTE: If the resource was advertised read-only, it must be mounted read-only.

Once this is copied, you will have all password and user mapping information available in your domain. You can also mount the *DOMAIN* resource from other domains containing hosts with which you want to communicate.

Step 2: When you set up the files that are read when you enter **init 3**, you can use **rfstart** with the **—v** option to indicate that all hosts must be verified when they try to connect to your host. The *domain* resource you copied into your **/usr/nserve/auth.info/***domain* directory will contain a **passwd** file. This file will have the name and encrypted password for each host in the domain. You can mount this information from other domains, so you can verify hosts from multiple domains.

If you use **rfstart —v**, any host in the *domain*/**passwd** file(s) will be verified; all other hosts will not be allowed to connect. If you use **rfstart** without the **—v** option, any host in the *domain*/**passwd** file(s) will be verified; all other hosts will be allowed to connect.

If you want to verify only a limited subset of these hosts, you must use a manually edited version of this file, removing any hosts you don't want to verify. (You cannot edit this file if you are a primary or secondary name server, however, since you must maintain a complete list of hosts for your domain. You must verify all hosts in the domains you serve.) When a host tries to connect, it will be checked against the name and password in the **/usr/nserve/auth.info/***domain*/**passwd** file.

See the description of **rfstart** in Chapter 11 to for details on how to set up host verification to work automatically upon entering **init 3.**

Step 3: You must determine the permissions remote users will have to your resources. This is done by mapping the remote users into the permissions of local users. You can map remote users both on a global basis or individually. (For details on setting remote user permissions, see Section 11.3 in Chapter 11.)

To map remote users you will use the **idload** command. This command will read several files to map remote users and groups:

/usr/nserve/auth.info/uid.rules
contains the rules for mapping remote users into user IDs on your host. You must create this file manually.

/usr/nserve/auth.info/gid.rules
contains the rules for mapping remote groups into group IDs on your host. You must create this file manually.

/etc/passwd your host's standard password file. It will be read by **idload** to get the information it needs on your local users.

/etc/group your host's standard group file. It will be read by **idload** to get the information it needs on your local groups.

remote The **idload** command will read all **passwd** and **group** files available from every *host* directory in every *domain* directory in **/usr/nserve/auth.info,** if you are mapping by remote name.

Step 4: Other hosts can map your users into their resources transparently or by numeric ID, without your providing lists of your users. However, in order to map your users and groups into remote hosts by name, you must provide a user and a group file. To create user and group lists, do the following:

1. Copy **/etc/passwd** and **/etc/group** files. Create copies of these files. (You can remove the encrypted passwords, if you like, but leave the colon (:) intact so the field is preserved.) The two new files should be called **passwd** and **group,** respectively.

2. Deliver files to domain name server. The files must be delivered to the name server host to become part of your domain's *domain* resource.

You can send these files to the name server host (**mail, uucp,** etc.) or put them into the proper place yourself by mounting the *domain* resource and copying them in. You can only put them in yourself if the resource was advertised and mounted with read/write permissions.

In either case, the two files must go into the directory **/usr/nserve/auth.info/***domain/host* where *domain* is the name of your domain and *host* is the name of your host. When these files have been added to the domain name server, other hosts will be able to copy these files and map your users and groups into the desired IDs on their hosts.

10.3 ADVERTISE RESOURCES (Procedure 10.3)

Purpose	Share your resources with other hosts.
Starting Conditions	System state—2 (multi-user) or 3 (RFS state) Login—**root**
Commands	**adv**(1M)
Reference	Chapter 11, Remote File Sharing

When your Remote File Sharing system is up and running, you can selectively share parts of your computer's file system using the **adv** command. The **adv** command will do the following:

- assign a resource name to a directory you want to share
- indicate whether the resource is read/write or read only (optional)
- attach a short description to the resource (optional)
- restrict access to the resource to certain hosts or domains (optional)
- register the resource with the domain name server

Step 1: You can share any directory on your GENIX V.3 system file system tree. When a remote host mounts the directory you advertise, all files and directories below that point will be available to the remote host. The files can also include executable files, named pipes, and devices. You can still use standard GENIX V.3 system means for protecting particular files and directories (**chmod**(1), for example).

Step 2: You must assign a name to the resource. This is the name remote hosts will use to access the resource. The resource name is limited to 14 printable ASCII characters and must be unique in your domain. Slash (/), period (.), and white space may not be used. If you make resource names all uppercase letters, it will be easier to distinguish them from path names.

Step 3: The form of the **adv** command to advertise a resource is:

adv [—**r**] [—**d** "*description*"] *resource pathname* [*clients...*]

The options are as follows:

—**r**	The —**r** option is for read-only. It is used to advertise the resource with read-only access. If it is not used, the remote hosts will have read/write access.
—**d** "*description*"	The —**d** option indicates that the next argument (*description*) is a description of the resource. The description can be from 0 to 32 characters and must be enclosed in quotes.
resource	This is the resource name you assign. The name is limited to 14 printable ASCII characters. Slash (/) and period (.) may not be used.
pathname	This is the full pathname to any local directory you want to share. **Adv** doesn't limit the number of characters in *pathname*.

The directory must reside on your local system, and it cannot be advertised as another resource.

clients... This is an optional list of one or more hosts or domains that you want to restrict this resource to. If clients are not included, the resource will be accessible to any Remote File Sharing host that can connect with your host.

A client name can be in the form *nodename* for hosts within your domain, *domain.* to advertise to an entire domain, or *domain.nodename*, to advertise to a host outside your domain. Note that you must place a period (.) at the end of a node name and in the middle of a full host name for each to be interpreted correctly. The client name can also be an alias to a group of hosts and/or domains you define in **/etc/host.alias.** (See Chapter 11 for the format of **host.alias.**)

Below are two examples of an **adv** command line:

adv —r —d "Department news" DNEWS /usr/news graph.

adv —d "My devices" MDEV /dev fee fie doc.host1

The first example advertises your **/usr/news** directory with read-only permissions under the resource name **DNEWS** to all hosts in the **graph** domain. The second advertises your **/dev** directory as **MDEV** to hosts **fee** and **fie** in your domain and **host1** in domain **doc.**

Step 4: You can set up all your **adv** commands to start automatically when the system enters the Remote File Sharing state (**init 3**). Do this by placing each full **adv** command line in the **/etc/rstab.** As soon as **init 3** successfully starts Remote File Sharing, all **adv** commands in **/etc/rstab** will be run. (See Chapter 11 for more details.)

The following is an example of an **/etc/rstab** file to automatically advertise the two resources shown above:

```
# cat /etc/rstab
adv —r —d "Department news" DNEWS /usr/news graph.
adv —d "My devices" MDEV /dev fee fie doc.host1
#
```

10.4 MOUNT REMOTE RESOURCES (Procedure 10.4)

Purpose	Use remote resources on your machine.
Starting Conditions	System state—2 (multi-user) or 3 (RFS state) Login—**root**
Commands	**nsquery**(1M) **mkdir**(1) **mount**(1M)
Reference	Chapter 11, Remote File Sharing

You can attach another host's advertised resource to your system using the standard **mount** command with new options. This procedure describes how to mount remote resources.

Step 1: You can check to see what resources are available from your domain using the **nsquery** command, as follows.

```
# nsquery
RESOURCE    ACCESS     SERVER     DESCRIPTION

GRAPH       read/write graph.fee  graph info
CALENDAR    read-only  graph.fie  Meetings
USERHELP    read-only  graph.foe  System help
#
```

For each available resource, **nsquery** will list the resource name (RESOURCE), whether it can be mounted read/write or read-only (ACCESS), the host that owns the resource (SERVER), and a short description. Using different options, you can also print resource information for a particular domain, host, or resource name.

Step 2: You must choose where you want to mount the resource. If the directory does not yet exist, create it as follows:

 # mkdir *dirname*

Step 3: To mount a remote resource, use the **mount** command as follows:

mount [**—r**] **—d** *special directory*

The options are as follows:

—r The **—r** option indicates that the resource should be mounted "read-only." If it is not used, the resource is mounted with read/write permissions. (A remote resource must be mounted "read-only" if it was advertised that way.)

—d This option is used to indicate that you are mounting a remote resource.

special For *special* you must substitute the name of the remote resource. Resources outside your domain must be entered in the form *domain.resource.*

directory Here you must specify the directory on which you will mount the resource.

Step 4: You can set up your **mount** commands to start automatically when the system enters Remote File Sharing state (**init 3**). This is done by placing entries in the **/etc/fstab** file. Each remote resource entry in **/etc/fstab** must be in the following format:

special directory —**d**[**r**]

Where *special* is replaced by the resource name, *directory* is where the resource will be mounted, and **—d** specifies that this is a remote mount. (You can use **—dr** instead of **—d** if you want to mount the remote resource read-only.)

After Remote File Sharing is started, **init 3** will spin off processes to try to do all remote mounts listed in **/etc/fstab** until they succeed.

Appendix A

DEVICE NAMES AND DESIGNATORS

A.1 INTRODUCTION

A.1.1 System V Release 2.0 and Later

Standard device names are used to identify the diskette and hard disk. The term "integral" is used to define devices driven by a controller on the system board. The first diskette drive and the first two hard disk drives are integral devices by this definition.

All disk device files use **/dev/dsk** for the block device, and **/dev/rdsk** for the raw (character) device:

- For the integral diskette drive, the controller/drive designator is **c0d2** for both the raw and block device.
- For the integral hard disk drives, the controller/drive designators are **c0d0** and **c0d1** for both the raw and block device.

The partition or section designator is appended to the drive control/drive designator.

The devices have a maximum of 8 partitions (0 through 7). The partitioning for the diskette are fixed by the processes used to format the media. The hard disk partitioning can be configured to best support the system application.

Certain of the device partitions are used for specific functions. These functions are required when a device is a bootable device. For example:

- Partition 0 is used for root (/).
- Partition 1 is used for swap when the device is configured as the root device for the GENIX V.3 operating system.
- Partition 2 is used for the **/usr** file system.
- Partition 7 specifies the entire device.

Device partitions should fall on cylinder boundaries to obtain the best possible file system performance. For a root device, boot and swap partitions (partitions 7 and 1) are special in this regard. The number of blocks assigned to the boot and swap partitions are collectively chosen to cause the next partition values to fall on cylinder boundaries. (The next partitions are normally used as file systems.) This approach eliminates wasted space that would result from strict assignment of partition values based on a modulo cylinder size for each partition of the root device.

Table A-1 describes typical *Series 32000* Computer controller/drive/section equipment configurations for the GENIX V.3 release.

TABLE A-1. EXAMPLES OF TYPICAL DEVICE PARTITIONS

DEVICE	CONTROLLER	DRIVE	SECTION	DESCRIPTION
c0d2s0	0	2	0	Specifies the section 0 (s0) of the integral floppy disk drive (d2) connected to controller 0 (c0).
c0d0s7	0	0	7	Specifies the section 7 (s7) of the first integral hard disk drive (d0) connected to controller 0 (c0).
c0d1s7	0	1	7	Specifies the section 7 (s7) of the second integral hard disk drive (d1) connected to controller 0 (c0).
c0d3s0	0	3	0	Specifies the section 0 (s0) of the second floppy disk (d3) connected to controller 0 (c0).

A.2 HARD DISK DEFAULT PARTITIONS

Table A-2 defines the use, size, and number of i-nodes for the 40-, and 140-megabyte hard disks. Note the following:

- The rotational gap (Gap) and blocks per cylinder (Cyl) are defined for each device.
- The number of i-nodes is based on the default value used by the **/etc/mkfs** command (the number of i-nodes is calculated based on the number of logical blocks (1024-byte blocks) divided by 4 and then rounded down to a modulo 16 value). Therefore, the values shown in the following tables are the result of a Full Restore procedure in which default partitioning is specified.
- The controller/device/section identifiers are applicable to both the raw and block devices.

NOTE: The rotational gap figures in the following tables are different from the default used by **mkfs**(1M). The figures in the tables are closer to optimum for the specific equipment listed; the **mkfs** command is designed for use across a broader range of equipment.

A.3 DISKETTE PARTITIONS

Table A-3 defines the diskette partitions in terms of use, starting sector, and total number of blocks for the various controller (c), drive (d), and section (s) identifiers for the diskette. Note the following:

- The raw and block device partitions for the entire diskette (partition 0) are linked to **/dev/rSA/diskette1** and **/dev/SA/diskette1**, respectively. The use of these names when specifying the entire diskette is preferred over the use of the controller, drive, and section identifiers to avoid accidentally writing to a different device or partition.
- These identifiers are applicable to both the raw and block devices.

TABLE A-2. SINGLE HARD DISK DEFAULT PARTITIONING

CONFIGURATION	PARTITION	USE	SECTOR START	SIZE	I-NODES
QUANTUM Q540 40 Megabyte Hard Disk (Gap = 9) (Blks/Cyl = 136)	c0d0s0	root	136	12648	1568
	c0d0s1	swap	12784	10200	---
	c0d0s2	usr	22984	45560	5680
	c0d0s7	entire disk	136	68408	---
MAXTOR X7-1140 140 Megabyte Hard Disk (Gap = 9) (Blks/Cyl = 240)	c0d0s0	root	240	15120	1888
	c0d0s1	swap	15360	10320	---
	c0d0s2	usr	25680	192240	24016
	cod057	entire disk	240	217680	---

TABLE A-3. FLOPPY DISK DRIVE

DISK PARTITION	USE	SECTOR START	TOTAL SECTORS*
c0d0s0	entire disk	0	1280
*Sectors are equivalent to 512-byte blocks			

Appendix B

DIRECTORIES AND FILES

B.1 INTRODUCTION

Appendix B describes directories and files of interest to a system administrator.

B.2 DIRECTORIES

The directories of the **root** file system (/) are as follows:

bck	Directory used to mount a backup file system for restoring files.
bin	Directory containing public commands.
dev	Directory containing special files that define all of the devices on the system.
etc	Directory containing administrative programs and tables.
install	Directory used by System Administration to mount utilities packages for installation and removal (**/install** file system).
lib	Directory containing public libraries.
lost+found	Directory used by **fsck**(1M) to save disconnected files.
mnt	Directory used to temporarily mount file systems during restoration of the operating system from floppy disks.
save	Directory used by Simple Administration for saving data on floppies.
stand	Directory containing stand-alone utilities.
tmp	Directory used for temporary files.
usr	Directory used to mount the **/usr** file system.

B.3 FILES

The following files and directories are important in the administration of the *Series 32000* Computer:

- **/etc/checklist**
- **/etc/fstab**
- **/etc/gettydefs**
- **/etc/group**
- **/etc/init.d Directory**

- /etc/inittab
- /etc/master.d Directory
- /etc/motd
- /etc/passwd
- /etc/profile
- /etc/rc0
- /etc/rc0.d Directory
- /etc/rc2
- /etc/rc2.d Directory
- /etc/rc.d Directory
- /etc/rc3
- /etc/rc3.d Directory
- /etc/rstab
- /etc/save.d Directory
- /etc/shutdown
- /etc/shutdown.d Directory
- /etc/TIMEZONE
- /etc/utmp
- /etc/wtmp
- /usr/adm/sulog
- /usr/lib/cron/log
- /usr/lib/help/HELPLOG
- /usr/lib/spell/spellhist
- /usr/news Directory
- /usr/options Directory.
- /usr/spool/cron/crontabs Directory

Each of these files is briefly described in this appendix.

B.3.1 /etc/checklist

The **/etc/checklist** file is used to define a default list of file system devices to be checked for consistency by **/etc/fsck** and **/etc/ncheck.** The character (raw) device partition for the file system should be identified. The devices listed normally correspond to those mounted when the system is in the multi-user mode (run level 2). The **root** file system (**/dev/rdsk/c1d0s0**) should not be listed in this file. Remember that with the exception of **root,** a file system must be unmounted to be checked. Therefore, the **checklist** file is a convenience for use when in the single-user mode of operation with only the **root** file system mounted. When the system is delivered, this file is empty. For a single-disk system the list is understandably brief. A typical *Series 32000* Computer **/etc/checklist** file is shown in Figure B-1. (See the **checklist**(4) manual page in the *Programmer's Reference Manual* for additional information.)

```
/dev/dsk/c1d0s8/usr2
/dev/dsk/c1d1s2/usr
```

Figure B-1. Typical /etc/checklist File

B.3.2 /etc/fstab

The **/etc/fstab** file is used as an argument to the **/etc/mountall** command. The **fstab** file specifies the file system(s) to be mounted by **/etc/mountall** and remote file system(s) to be mounted by **/etc/rmountall.** A typical **/etc/fstab** file for a dual-disk *Series 32000* Computer is shown in Figure B-2. The format of the file is the block device name followed by the mount point name. (See the **mountall**(1M) manual page in the *Administrator's Reference Manual* for additional information.)

```
/dev/dsk/c1d0s2 /usr
/dev/dsk/c1d1s2 /usr2
RES1 /mnt -d
```

Figure B-2. Typical /etc/fstab File

B.3.3 /etc/gettydefs

The **/etc/gettydefs** file contains information that is used by **/etc/getty** to set the speed and terminal settings for a line. The **getty** command accesses the **gettydefs** file with a label. The general format of the **gettydefs** file is as follows:

label# initial-flags # final-flags #login-prompt #next-label

Each line entry in the **gettydefs** file is followed by a blank line. (Refer to the **gettydefs**(4) manual page in the *Programmer's Reference Manual* for complete information.) Figure B-3 shows a typical *Series 32000* Computer **/etc/gettydefs** file.

B.3.4 /etc/group

The **/etc/group** file describes each group to the system. An entry is added for each new group. Each entry in the file is one line and consists of four fields, which are separated by a colon (:):

group name:password:group id:login names

Explanations for these fields are as follows:

group name — The first field defines the group name. The group name is from three to six characters long. The first character is alphabetic. The rest of the characters are alphanumeric. No uppercase characters appear.

password — The second field contains the encrypted group password. The encrypted group password contains 13 bytes (characters). The actual password is limited to a maximum of 8 bytes. The encrypted password can be followed by a comma and up to 4 more bytes of password aging information. The use of group passwords is discouraged.

group id — The third field contains the group identification number, which must be between 0 and 60,000. Group identification numbers 0 through 99 are reserved; 0 indicates the super-user (root). Commas are not entered in this field.

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600
9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800
4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400
2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200
1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300
300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
console# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3 #Console Login: #console
1
console1# B1200 HUPCL OPOST ONLCR # B1200 SANE IXANY TAB3 #Console Login: #consol
e2
console2# B300 HUPCL OPOST ONLCR # B300 SANE IXANY TAB3 #Console Login: #console3
console3# B2400 HUPCL OPOST ONLCR # B2400 SANE IXANY TAB3 #Console Login: #consol
e4
console4# B4800 HUPCL OPOST ONLCR # B4800 SANE IXANY TAB3 #Console Login: #consol
e5
console5# B19200 HUPCL OPOST ONLCR # B19200 SANE IXANY TAB3 #Console Login: #cons
ole
contty# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3 #login: #contty1
contty1# B1200 HUPCL OPOST ONLCR # B1200 SANE IXANY TAB3 #login: #contty2
contty2# B300 HUPCL OPOST ONLCR # B300 SANE IXANY TAB3 #login: #contty3
contty3# B2400 HUPCL OPOST ONLCR # B2400 SANE IXANY TAB3 #login: #contty4
contty4# B4800 HUPCL OPOST ONLCR # B4800 SANE IXANY TAB3 #login: #contty5
contty5# B19200 HUPCL OPOST ONLCR # B19200 SANE IXANY TAB3 #login: #contty
pty# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3 #PC login: #pty
4800H# B4800 # B4800 SANE IXANY TAB3 HUPCL #login: #9600H
9600H# B9600 # B9600 SANE IXANY TAB3 HUPCL #login: #19200H
19200H# B19200 # B19200 SANE IXANY TAB3 HUPCL #login: #2400H
2400H# B2400 # B2400 SANE IXANY TAB3 HUPCL #login: #1200H
1200H# B1200 # B1200 SANE IXANY TAB3 HUPCL #login: #300H
300H# B300 # B300 SANE IXANY TAB3 HUPCL #login: #4800H
conttyH# B9600 OPOST ONLCR # B9600 HUPCL SANE IXANY TAB3 #login: #contty1H
contty1H# B1200 OPOST ONLCR # B1200 HUPCL SANE IXANY TAB3 #login: #contty2H
contty2H# B300 OPOST ONLCR # B300 HUPCL SANE IXANY TAB3 #login: #contty3H
contty3H# B2400 OPOST ONLCR # B2400 HUPCL SANE IXANY TAB3 #login: #contty4H
contty4H# B4800 OPOST ONLCR # B4800 HUPCL SANE IXANY TAB3 #login: #contty5H
contty5H# B19200 OPOST ONLCR # B19200 HUPCL SANE IXANY TAB3 #login: #conttyH
```

Figure B-3. Typical Gettydefs File

login names The fourth field contains a list of all login names in the group. Names in the list are separated by commas. The names listed may use the **/etc/newgrp** command to become a member of the group.

Figure B-4 shows a typical *Series 32000* Computer **/etc/group** file.

```
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
mail::6:root
rje::8:rje,shqer
daemon::12:root,daemon
```

Figure B-4. Typical /etc/group File

B.3.5 /etc/init.d Directory

The **/etc/init.d** directory contains executable files used in upward and downward transitions to all system run levels. These files are linked to files beginning with **S** (start) or **K** (stop) in **/etc/rc***n***.d**, where *n* is the appropriate run level. Files are not executed from this directory. They are only executed from **/etc/rc***n***.d** directories.

B.3.6 /etc/inittab

The **/etc/inittab** file contains instructions for the **/etc/init** command. The instructions define the processes that are to be created or terminated for each initialization state. Initialization states are called run levels or run-states. By convention, run level 1 (or S or s) is single-user mode; run levels 2 and 3 are multi-user modes. Chapter 4 summarizes the various run levels and describes their uses. (See the **inittab**(4) manual page in the *Programmer's Reference Manual* for additional information.) Figure B-5 shows a typical *Series 32000* Computer **/etc/inittab** file. The typical entry is a series of fields separated by a colon (:):

identification:run-state:action:process

Explanations for these fields are as follows:

```
zu::sysinit:/etc/bzapunix </dev/console >/dev/console 2>&1
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
ck::sysinit:/etc/setclk </dev/console >/dev/console 2>&1
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
pt:23:bootwait:/etc/ports </dev/console >/dev/console 2>&1
is:2:initdefault:
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2>&1 </dev/console
fl:056:wait:/etc/led -f >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
RB:6:wait:echo "The system is being restarted." >/dev/console 2>&1
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
ct:234:off:/etc/getty -t 60 contty 4800
he:234:respawn:sh -c 'sleep 20 ; exec /etc/hdelogger >/dev/console 2>&1'
11:234:off:/etc/getty tty11 9600
12:234:off:/etc/getty tty12 9600
13:234:off:/etc/getty tty13 9600
14:234:off:/etc/getty tty14 9600
15:234:off:/etc/getty tty15 9600
31:234:respawn:/usr/lib/uucp/uugetty -r -t 60 tty31 4800H
32:234:respawn:/usr/lib/uucp/uugetty -r -t 60 tty32 4800H
33:234:off:/etc/getty tty33 9600
34:234:off:/etc/getty tty34 9600
35:234:off:/etc/getty tty35 9600
```

Figure B-5. Typical /etc/inittab File

identification The identification field is a one- or two-character identifier for the line entry. The identifier is unique for a line.

run-state The run-state defines the run level in which the entry is to be processed.

action The action field defines how **/etc/init** treats the process field. (Refer to the **inittab**(4) manual page in the *Programmer's Reference Manual* for complete information.)

process The process field defines the shell command that is to be executed.

B.3.7 /etc/master.d Directory

The **/etc/master.d** directory contains files that define the configuration of hardware devices, software drivers, system parameters and aliases. The files are used by **/etc/config** to obtain device information for the generation of device driver and configurable module files. The **/etc/sysdef**(1M) program uses the **master.d** files to get the names of supported devices. The first step in reconfiguring the system to run with different tunable parameters is to edit the appropriate files in the **/etc/master.d** directory. (Refer to the **/etc/master**(4) manual page in the *Programmer's Reference Manual* for additional information.)

B.3.8 /etc/motd

The **/etc/motd** file contains the message-of-the-day. The message-of-the-day is output by instructions in the **/etc/profile** file after a successful login. This message should be kept short and to the point. The **/usr/news** file(s) should be used for lengthy, more explicit messages.

B.3.9 /etc/passwd

The **/etc/passwd** file identifies each user to the system. An entry is added for each new user. Each entry in the file is one line and consists of seven fields. The fields are separated by a colon (:):

login name:passwd:user:group:account:login directory:program

Explanations for these fields are as follows:

login name The first field defines the login name. The login name is from three to six characters long. The first character is alphabetic. The rest of the characters are alphanumeric. No uppercase characters appear.

passwd The second field contains the encrypted login password. The encrypted login password contains 13 bytes (characters). The actual password is limited to a maximum of 8 bytes. The encrypted password can be followed by a comma and up to 4 more bytes of password aging information.

user id The third field contains the user identification number, which must be between 0 and 60,000. Group identification numbers 0 through 99 are reserved; 0 indicates the super-user (root). Commas are not entered in this field.

group id The fourth field contains the group identification number, which must be between 0 and 60,000. Group identification numbers 0 through 99 are reserved; 0 indicates the super-user (root). Commas are not entered in this field.

account The fifth field is used by accounting programs. This field typically contains the user name, department number, and bin number.

login directory The sixth field defines the full path name of the login directory.

program The seventh field defines the program to be executed after login. If it is null, the shell (**/bin/sh**) is invoked.

Figure B-6 shows a typical **/etc/passwd** file. (See the **passwd**(4) manual page in the *Programmer's Reference Manual* for additional information.)

B.3.10 /etc/profile

The default profile for all users is in the **/etc/profile** file. The standard (default) environment for all users is established by the instructions in the **/etc/profile** file. The system administrator can modify this file to set options for the **root** login. For example, the following can be added to the **/etc/profile** for the **root** login to cause the erase character to back up and to set the TERM variable.

```
if [ ${LOGNAME} = root ]
      then
            stty echoe
            echo "Enter TERM: \c"
            read TERM
            export TERM
```

Figure B-7 shows the *Series 32000* Computer default profile.

B.3.11 /etc/rc0

The **/etc/rc0** file contains a shell script that is executed by **/etc/shutdown** for transitions to single-user state, and by **/etc/init** on transitions to run levels 0, 5, and 6. Files in the **/etc/shutdown.d** and **/etc/rc0.d** directories are executed when **/etc/rc0** is run. The file **K00ANNOUNCE** in **/etc/rc0.d** prints the message "System services are now being stopped." Any task that you want executed when the system is taken to run levels 0, s, 5, or 6 can be done by adding a file to the **/etc/shutdown.d** directory. Figure B-8 shows a typical *Series 32000* Computer **/etc/rc0** file.

```
root:CQCpoiY.hyx5M:0:1:0000-Admin(0000):/:
daemon:DT8E7TMqEGAE.:1:1:0000-Admin(0000):/:
bin:NTwEs.ajCTuQ6:2:2:0000-Admin(0000):/bin:
sys:ZQJgJg55l50HI:3:3:0000-Admin(0000):/usr/src:
adm:iQ0OMArb9NXZM:4:4:0000-Admin(0000):/usr/adm:
uucp:rQ5fM/1P8Z6yU:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp:4Rka.6jZpmZeI:10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/uuc
ico
rje:FRStaYWcMn0do:18:18:0000-rje(0000):/usr/rje:
trouble:PRS31ul156VCI:70:1:trouble(0000):/usr/lib/trouble:
lp:YR6Oh4EJQfTg2:71:2:0000-lp(0000):/usr/spool/lp:
setup:xEwCpdK/pnbHk:0:0:general system administration:/usr/admin:/bin/rsh
powerdown:BF.j6TnLXfDjc:0:0:general system administration:/usr/admin:/bin/rsh
sysadm:NFS9vmN./S0m2:0:0:general system administration:/usr/admin:/bin/rsh
checkfsys:KGAk9JTaQ6Cjw:0:0:check diskette file system:/usr/admin:/bin/rsh
makefsys:ZGbRfC.pymZXs:0:0:make diskette file system:/usr/admin:/bin/rsh
mountfsys:lG1NIjh7PBEJk:0:0:mount diskette file system:/usr/admin:/bin/rsh
umountfsys:xGBzGUp7U.i3E:0:0:unmount diskette file system:/usr/admin:/bin/rsh
listen:np:37:4:Network Admin:/usr/net/nls:
ral::100:1:Robert Lippman:/usr/ral:
mcn::101:1:Chris Negus:/usr/mcn:
kol::102:1:Kathy O'Leary:/usr/kol:
jcb::103:1:Jim Blinn:/usr/jcb:
```

Figure B-6. Typical /etc/passwd File

```
# The profile that all logins get before using their own .profile.
trap "" 2 3
export LOGNAME
. /etc/TIMEZONE
# Login and -su shells get /etc/profile services.
# -rsh is given its environment in its .profile.
case "$0" in
-su )
        export PATH
        ;;
-sh )
        export PATH
      # Allow the user to break the Message-Of-The-Day only.
        trap "trap " 2" 2
        cat -s /etc/motd
        trap "" 2
      if mail -e
        then
                echo "you have mail"
        fi
      if [ ${LOGNAME} != root ]
        then
                news -n
        fi
        ;;
esac
umask 022
trap 2 3
```

Figure B-7. Standard /etc/profile File

```
# "Run Commands" for init state 0
# Leaves the system in a state where it is safe to turn off the power
# or go to firmware.
stty sane tab3 2>/dev/null
echo 'The system is coming down.  Please wait.'
if [ -d /etc/shutdown.d ]
then
        for f in /etc/shutdown.d/*
        { if [ -s $f ]
                then
                        /bin/sh ${f}
                fi
        }
fi
#       End of historical section
if [ -d /etc/rc0.d ]
then
        for f in /etc/rc0.d/K*
        {
                if [ -s ${f} ]
                then
                        /bin/sh ${f} stop
                fi
        }
.
.
.
```

Figure B-8. Typical /etc/rc0 File
(Sheet 1 of 2)

```
.
.
.
# system cleanup functions ONLY (things that end fast!)
for f in /etc/rc0.d/S*
        {
                if [ -s ${f} ]
                then
                        /bin/sh ${f} start
                fi
        }
fi
trap "" 15
kill -15 -1
sleep 10
/etc/killall  9
sleep 10
sync;sync;sync
/etc/umountall
stty sane 2>/dev/null
sync;  sync
echo '
The system is down.'
sync
```

Figure B-8. Typical /etc/rc0 File
(Sheet 2 of 2)

B.3.12 /etc/rc0.d Directory

The **/etc/rc0.d** directory contains files executed by **/etc/rc0** for transitions to system run levels 0, 5, and 6. Files in this directory are linked from the **/etc/init.d** directory, and begin with either a **K** or an **S**. **K** indicates processes that are stopped, and **S** indicates processes that are started when entering run levels 0, 5, or 6.

B.3.13 /etc/rc2

The **/etc/rc2** file contains a shell script that is executed by **/etc/init** on transitions to run level 2 (multi-user state). Executable files in the **/etc/rc.d** and any executable files beginning with **S** or **K** in **/etc/rc2.d** directories are executed when **/etc/rc2** is run. All files in **rc2.d** are linked from files in the **/etc/init.d** directory. The following are descriptions of some of those files. These files are prefixed with an **S** or a **K** and a number in the **/etc/rc2.d** directory.

MOUNTFILESYS Sets up and mounts file systems. Builds the mount table and mounts the **root** (/) and user (**/usr**) file systems. Makes the **/usr/tmp** directory, cleaning up (deleting) any previous files in that directory.

autoconfig Makes a **/unix** if self-configuration occurred during the boot sequence. The new in-memory operating system is copied to **/unix.**

cron Starts the **cron** daemon by executing **/etc/cron.**

sysetup Removes the **/etc/ps_data** to force the **/bin/ps** command to read the **/unix** file. Outputs the system configuration if the **/etc/prtconf** command exists. Outputs the system trademark information.

uucp When basic networking is added to the system, the **uucp** file is added to this directory. The **uucp** file deletes **uucp** locks (LCK*), status files (STST*), and temporary files (TM*) under the **/usr/spool/uucp** directory structure.

lp When line printer spooling is added to the system, the **lp** file is added to the **rc.d.** The **lp** file removes the spooler lock file and starts the scheduler.

Other files may also be added to **/etc/rc2.d** and **/etc/rc.d** directories as a function of adding hardware or software to the system. Figure B-9 shows a typical *Series 32000* Computer **/etc/rc2** file.

B.3.14 /etc/rc2.d Directory

The **/etc/rc2.d** directory contains files executed by **/etc/rc2** for transitions to system run level 3. Files in this directory are linked from the **/etc/init.d** directory, and begin with either a **K** or an **S**. **K** indicates processes that should be stopped, and **S** indicates processes that should be started when entering run levels 2 or 3.

```
# "Run Commands" executed when the system is changing to init state 2,
# traditionally called "multi-user".
. /etc/TIMEZONE
# Pickup start-up packages for mounts, daemons, services, etc.
set 'who -r'
if [ $9 = "S" ]
then
  echo 'The system is coming up.  Please wait.'
  BOOT=yes
  if [ -f /etc/rc.d/PRESERVE ]  # historical segment for vi and ex
        then
                mv /etc/rc.d/PRESERVE /etc/init.d
                ln /etc/init.d/PRESERVE /etc/rc2.d/S02PRESERVE
        fi
elif [ $7 = "2" ]
then
        echo 'Changing to state 2.'
        if [ -d /etc/rc2.d ]
        then
                for f in /etc/rc2.d/K*
                {
                        if [ -s ${f} ]
                        then
                                /bin/sh ${f} stop
                        fi
                }
        fi
fi
  .
  .
  .
```

Figure B-9. Typical /etc/rc2 File
(Sheet 1 of 2)

```
.
.
.
if [ -d /etc/rc2.d ]
then
        for f in /etc/rc2.d/S*
        {
                if [ -s ${f} ]
                then
                        /bin/sh ${f} start
                fi
        }
fi
if [ "${BOOT}" = "yes" ]
then
        stty sane tab3 2>/dev/null
fi
if [ "${BOOT}" = "yes" -a -d /etc/rc.d ]
then
        for f in `ls /etc/rc.d`
        {
                if [ ! -s /etc/init.d/${f} ]
                then
                        /bin/sh /etc/rc.d/${f}
                fi
        }
fi
if [ "${BOOT}" = "yes" -a $7 = "2" ]
then
        echo 'The system is ready.'
elif [ $7 = "2" ]
then
        echo 'Change to state 2 has been completed.'
fi
```

Figure B-9. Typical /etc/rc2 File
(Sheet 2 of 2)

B.3.15 /etc/rc.d Directory

The **/etc/rc.d** directory contains executable files that do the various functions needed to initialize the system to run level 2. The files are executed when **/etc/rc2** is run. (Files contained in this directory prior to the GENIX V.3 release were moved to **/etc/rc2.d.** This directory is only maintained for compatibility reasons.)

B.3.16 /etc/rc3

The **/etc/rc3** file is executed by **/etc/init.** It executes the shell scripts in **/etc/rc3.d** on transitions to system run level 3 (the Remote File Sharing state).

B.3.17 /etc/rc3.d Directory

The **/etc/rc3.d** directory contains files executed by **/etc/rc3** for transitions to system run level 3 (multi-user mode). Files in this directory are linked from the **/etc/init.d** directory, and begin with either a **K** or an **S. K** indicates processes that should be stopped, and **S** indicates processes that should be started when entering run level 3.

B.3.18 /etc/rstab

The **/etc/rstab** file is used to specify the Remote File Sharing resources from your machine that are automatically offered to remote machines upon entering system run level 3 (Remote File Sharing state). Entries in this file should be entire **adv**(1M) command lines. (See Chapter 11 for details.)

B.3.19 /etc/save.d Directory

The **/etc/save.d** directory contains files that are used by the System Administration Menu commands associated with backing up data on floppy disks. The following files are included:

except A list of the directories and files that should not be copied as part of a backup (**savefiles**) is maintained in this file.

timestamp/... The date and time of the last backup (volume or incremental) is maintained for each file system in the **/etc/timestamp** directory.

B.3.20 /etc/shutdown

The **/etc/shutdown** file contains a shell script to shut down the system gracefully in preparation for system backup or scheduled downtime. After stopping all nonessential processes, the **shutdown** script executes files in the **/etc/shutdown.d** directory by calling **/etc/rc0** for transitions to run level s or S. For transitions to other run levels, the **shutdown** script calls **/etc/init.** Figure B-10 shows a typical *Series 32000* Computer **/etc/shutdown** file.

```
# Sequence performed to change the init stat of a machine.
# This procedure checks to see if you are permitted and allows an
# interactive shutdown. The actual change of state, killing of
# processes and such are performed by the new init state, say 0,
# and its /etc/rc0.
# Usage: shutdown [ -y ] [ -g <grace-period> ] [ -i <init-state> ]
#!      chmod +x ${file}
if [ 'pwd' != / ]
then
echo "$0: You must be in the / directory to run /etc/shutdown."
exit 1
fi
#       Check the user id.
if [ -x /usr/bin/id ]
then
        eval 'id | sed 's/[^a-z0-9=].*//''
        if [ "${uid:=0}" -ne 0 ]
        then
              echo "$0: Only root can run /etc/shutdown."
                exit 2
        fi
fi
grace=60
askconfirmation=yes
initstate=s
while [ $# -gt 0 ]
do
        case $1 in
        -g[0-9]* )
                grace='expr "$1" : '-g\(.*\)''
                ;;
        -i[Ss0156] )
.
.
.
```

Figure B-10. Typical /etc/shutdown File
(Sheet 1 of 4)

```
.
.
.
        initstate=`expr "$1" : '-i\(.*\)'`
        ;;
  -i[234] )
        initstate=`expr "$1" : '-i\(.*\)'`
        echo "$0: Initstate $i is not for system shutdown"
        exit 1
        ;;
  -y )
        askconfirmation=
        ;;
  -* )
        echo "Illegal flag argument '$1'"
        exit 1
        ;;
  * )
        echo "Usage: $0 [ -y ] [ -g <grace> ] [ -i <initstate> ]"
        exit 1
        esac
        shift
done
if [ -n "${askconfirmation}" -a -x /etc/ckbupscd ]
        then
#       Check to see if backups are scheduled at this time
        BUPS=`/etc/ckbupscd`
        if [ "$BUPS" != "" ]
        then
        echo "$BUPS"
        echo "Do you wish to abort this shutdown and return to
command level to do these backups? [y, n] \c"
                read YORN
.
.
.
```

Figure B-10. Typical /etc/shutdown File
(Sheet 2 of 4)

```
.
.
.
                if [ "$YORN" = "y" -o "$YORN" = "Y" ]
                        then
                        exit 1
                fi
        fi
fi
if [ -z "${TZ}"  -a  -r /etc/TIMEZONE ]
then
  . /etc/TIMEZONE
fi
echo '
Shutdown started.    \c'
date
echo
sync
cd /
trap "exit 1"  1 2 15
a="`who  |  wc -l`"
if [ ${a} -gt 1  -a  ${grace} -gt 0 ]
then
su adm -c /etc/wall<<-!
        The system will be shut down in ${grace} seconds.
        Please log off now.
!
        sleep ${grace}
fi
/etc/wall  <<-!
        THE SYSTEM IS BEING SHUT DOWN NOW !!!
        Log off now or risk your files being damaged.
!
sleep ${grace}
.
.
.
```

Figure B-10. Typical /etc/shutdown File
(Sheet 3 of 4)

```
.
.
.
if [ ${askconfirmation} ]
then
        echo "Do you want to continue? (y or n):  \c"
        read b
else
        b=y
fi
if [ "$b" != "y" ]
then
        /etc/wall << -\!
        False Alarm: The system will not be brought down.
        !
        echo 'Shut down aborted.'
        exit 1
fi
case "${initstate}" in
s | S )
   . /etc/rc0
esac
/etc/init ${initstate}
```

Figure B-10. Typical /etc/shutdown File
(Sheet 4 of 4)

B.3.21 /etc/shutdown.d Directory

The executable files in **/etc/shutdown.d** perform the various functions required during the transition to the single-user state (run levels 1, s, or S). (Files contained in this directory prior to GENIX V.3 release were moved to **/etc/rc0.d**. This directory is only maintained for compatibility reasons.)

B.3.22 /etc/TIMEZONE

The **/etc/TIMEZONE** file sets the time zone shell variable TZ. The TZ variable is initially established for the system via the System Administration **setup** function. The TZ variable in the **TIMEZONE** file is changed by the System Administration **timezone** command (**sysadm timezone**). The TZ variable can be redefined on a user (login) basis by setting the variable in the associated **.profile**. The **TIMEZONE** file is executed by **/etc/rc2**. Figure B-11 shows a typical **/etc/TIMEZONE** file.

```
# Set timezone environment to default for the machine
TZ=EST5EDT
export TZ
```

Figure B-11. Typical /etc/TIMEZONE File

The format of the **TZ** is as follows:

TZ=*TTT#SSS*

Explanations for the fields of the **TZ** variable are as follows:

TTT	The three-character abbreviation for the local time zone.
#	The number of hours that the local time zone differs from Greenwich Mean Time (GMT). This field can be entered as a positive or negative number.
SSS	The three-character abbreviation for the local daylight savings time zone. This field is entered only if daylight savings time is observed.

B.3.23 /etc/utmp

The **/etc/utmp** file contains information on the run-state of the system. This information is accessed with a **who -a** command.

B.3.24 /etc/wtmp

The **/etc/wtmp** file contains a history of system logins. The owner and group of this file must be **adm**, and the access permissions must be 664. Each time **login** is run this file is updated. As the system is accessed, this file increases in size. Periodically, this file should be cleared or truncated. The command line **>/etc/wtmp** when executed by **root** creates the file with nothing in it. The following command line limits the size of the **/etc/wtmp** file to the last 3600 characters in the file:

tail -3600c /etc/wtmp > /tmp/wtmp; mv /tmp/wtmp /etc/wtmp

Note that **/etc/cron, /etc/rc0,** or **/etc/rc2** can be used to clean up the **wtmp** file. To use one of these functions, add the appropriate command line to the **/usr/spool/cron/crontab/root, /etc/shutdown.d/ ...,** or **/etc/rc.d/, rc2.d, rc3.d ...** file.

B.3.25 /usr/adm/sulog

The **/usr/adm/sulog** file contains a history of substitute user (**su**) command usage. As a security measure, this file should not be readable by **others.** The **/usr/adm/sulog** file should be periodically truncated to keep the size of the file within a reasonable limit. Note that **/etc/cron, /etc/rc0,** or **/etc/rc2** can be used to clean up the **sulog** file. To use one of these functions, add the appropriate command line to the **/usr/spool/cron/crontab/root, /etc/shutdown.d/ ...,** or **/etc/rc.d/, rc2.d, rc3.d ...** file. The following command lines limit the size of the log file to the last 100 lines in the file:

tail -100 /usr/adm/sulog > /tmp/sulog
mv /tmp/sulog /usr/adm/sulog

Figure B-12 shows the contents of a typical **/usr/adm/sulog** file.

B.3.26 /usr/lib/cron/log

A history of all actions taken by **/etc/cron** is recorded in the **/usr/lib/cron/log** file. The **/usr/lib/cron/log** file should be periodically truncated to keep the size of the file within a reasonable limit. Note that **/etc/cron, /etc/rc0,** or **/etc/rc2** can be used to clean up the **/usr/lib/cron/log** file. To use one of these functions to limit the size of a log file, add the appropriate command line to the **/usr/spool/cron/crontab/root, /etc/shutdown.d/ ...,** or **/etc/rc.d/ rc2.d, rc3.d ...** file, as applicable. The following command line limits the size of the log file to the last 100 lines in the file:

tail -100 /usr/lib/cron/log > /tmp/log; mv /tmp/log /usr/lib/cron/log

Figure B-13 shows the information typically found in the **/usr/lib/cron/log** file.

```
SU 08/18 12:35 + console root-sysadm
SU 08/18 16:11 + console root-sysadm
SU 08/18 16:16 + console root-sysadm
SU 08/18 23:45 + tty?? root-uucp
SU 08/19 11:53 + console root-sysadm
SU 08/19 15:25 + console root-sysadm
SU 08/19 23:45 + tty?? root-uucp
SU 08/20 10:16 + console root-adm
SU 08/20 10:33 + tty24 rar-root
SU 08/20 10:42 + console root-sysadm
SU 08/20 10:59 + console root-root
SU 08/20 11:01 + console root-sysadm
SU 08/20 12:36 + tty11 bin-bin
SU 08/20 12:37 + tty11 tws-bin
SU 08/20 14:42 - tty24 awa-sys
SU 08/20 14:47 - tty24 awa-sys
SU 08/20 14:48 + tty24 awa-root
SU 08/20 15:44 + console root-sysadm
```

Figure B-12. Typical /usr/adm/sulog File

```
! *** cron started ***  pid = 237 Sun Aug 19 14:06:45 1984
> CMD: /usr/lib/uucp/uudemon.hour > /dev/null
> root 251 c Sun Aug 19 14:11:00 1984
< root 251 c Sun Aug 19 14:11:01 1984
> CMD: /usr/lib/uucp/uudemon.poll > /dev/null
> root 370 c Sun Aug 19 14:30:00 1984
< root 370 c Sun Aug 19 14:30:03 1984
> CMD: /usr/lib/uucp/uudemon.hour > /dev/null
> root 417 c Sun Aug 19 14:41:01 1984
< root 417 c Sun Aug 19 14:41:02 1984
> CMD: /usr/lib/uucp/uudemon.poll > /dev/null
> root 452 c Sun Aug 19 15:01:00 1984
< root 452 c Sun Aug 19 15:01:04 1984
> CMD: /usr/lib/uucp/uudemon.hour > /dev/null
> root 460 c Sun Aug 19 15:11:00 1984
< root 460 c Sun Aug 19 15:11:00 1984
> CMD: /usr/lib/uucp/uudemon.poll > /dev/null
> root 541 c Sun Aug 19 15:30:00 1984
< root 541 c Sun Aug 19 15:30:07 1984
```

Figure B-13. Typical /usr/lib/cron/log File

B.3.27 /usr/lib/help/HELPLOG

Providing that **help** monitoring has been enabled, a history of all actions taken by the **/usr/bin/help** command is kept in the **/usr/lib/help/HELPLOG** file. The **HELPLOG** file is copied to **/usr/lib/help/oHELPLOG** and a new **/usr/lib/help/HELPLOG** file is created by the **/usr/lib/help/helpclean** command. Executing the **helpclean** command twice in succession zeros out the **HELPLOG** and the **oHELPLOG** files. Note that **/etc/cron, /etc/rc0,** or **/etc/rc2** can be used to clean up the **HELPLOG** file. To use one of these functions, add the appropriate command line to the **/usr/spool/cron/crontab/root, /etc/shutdown.d/ ...**, or **/etc/rc.d/ rc2.d, rc3.d ...** file, as applicable. The following command lines limit the size of the log file to the last 100 lines in the file:

```
tail -100 /usr/lib/help/HELPLOG > /tmp/help
mv /tmp/help /usr/lib/help/HELPLOG
```

Figure B-14 shows the information typically found in the **/usr/lib/help/HELPLOG** file.

```
login=bin     uname=bugc    date=Mon Aug 20 12:51:03 EDT 1984
name=locate   response='l'  status=OK
name=locate   response='d'  status=ERROR
name=getkey   response='k'  status=OK
name=keysrch  response='list' status=OK
name=quit     response='q'  status=OK
login=bin     uname=bugc    date=Mon Aug 20 12:51:41 EDT 1984
```

Figure B-14. Typical /usr/lib/help/HELPLOG File

B.3.28 /usr/lib/spell/spellhist

If the Spell Utilities is installed, a history of all words that **spell**(1) fails to match is kept in the **/usr/lib/spell/spellhist** file. Periodically, this file should be reviewed for words that should be added to the dictionary. After the **spellhist** file is reviewed, it can be cleared.

B.3.29 /usr/news

The **/usr/news** directory contains news files. The file names are descriptive of the contents of the files; they are analogous to headlines. When a user reads the news, using the **news** command, an empty file named **.news_time** is created in his or her login directory. The date (time) of this file is used by the **news** command to determine if a user has read the latest news file(s).

B.3.30 /usr/options Directory

The **/usr/options** directory contains files that identify the utilities that are installed on the system. Figure B-15 shows a typical **/usr/options** directory. Because no system has all possible utilities installed on it at once, the list of files in the **/usr/options** directory will not include all utilities.

The example shown in Figure B-15 was taken from a system with a 32-megabyte hard disk. All of the utilities identified in Figure B-15 cannot be loaded on a 10-megabyte hard disk at the same time. The full names of the utilities are contained in the **/usr/options** files. The contents of the files identified in Figure B-15 are shown in Figure B-16.

```
-r- xr-xr-x   1 bin    bin        22 Sep  7 15:17 calc.name
-r- xr-xr-x   1 bin    bin        23 Sep 10 17:03 cc.name
-r- -r--r--   1 bin    bin        34 Aug 10 15:30 crypt.name
-r- xr-xr-x   1 bin    bin        40 Aug 10 15:32 dfm.name
-r- -r--r--   1 bin    bin        19 Aug 21 12:44 dmd.name
-r- -r--r--   1 bin    bin        38 Aug 21 14:23 dmdapd.name
-r- -r--r--   1 bin    bin        55 Aug 21 14:47 dmdtxt.name
-r- xr-xr-x   1 bin    bin        30 Apr 26 11:28 dwb.name
-r- -r--r--   1 bin    bin        18 Aug 10 15:30 ed.name
-r- xr-xr-x   1 bin    bin        39 Aug  3 11:10 esg.name
-r- xr-xr-x   1 bin    bin        19 Aug 10 15:48 graph.name
-r- xr-xr-x   1 bin    bin        15 Aug 10 15:51 help.name
-r- xr-xr-x   1 bin    bin        38 Aug 10 15:53 ipc.name
-r- -r--r--   1 bin    bin        22 Aug 11 01:34 kersrc.name
-r- xr-xr-x   1 bin    bin        32 Aug 10 15:55 lp.name
-r- xr-xr-x   1 bin    bin        35 Aug 10 15:57 perf.name
-r- xr-xr-x   1 bin    bin        30 Aug 10 15:58 sccs.name
-r- xr-xr-x   1 bin    bin        30 Aug 10 16:04 sgs.name
-r- xr-xr-x   1 bin    bin        28 Sep  7 15:17 shell.name
-r- xr-xr-x   1 bin    bin        16 Aug 10 16:07 spell.name
-r- xr-xr-x   1 bin    bin        32 Aug 10 16:08 sysadm.name
-r- xr-xr-x   1 bin    bin        27 Aug 10 16:10 term.name
-r- -r--r--   1 bin    bin        31 Aug 10 16:11 terminf.name
-r- xr-xr-x   1 bin    bin        27 Sep  7 15:17 usrenv.name
-rwxr-xr-x   1 root   sys        27 Aug 10 15:18 uucp.name
```

Figure B-15. Typical /usr/options Directory

Calculation Utilities
C Programming Language
Security Administration Utilities
Directory and File Management Utilities
DMD Core Utilities
DMD Application Development Utilities
DMD Text Processing and Graphics Application Utilities
DOCUMENTER'S WORKBENCH System
Editing Utilities
Extended Software Generation Utilities
Graphics Utilities
HELP Utilities
Inter-Process Communication Utilities
Kernel Source Package
Line Printer Spooling Utilities
Performance Measurements Utilities
Source Code Control Utilities
Software Generation Utilities
Shell Programming Utilities
SPELL Utilities
System Administration Utilities
Terminal Filters Utilities
Terminal Information Utilities
User Environment Utilities
Basic Networking Utilities

Figure B-16. Typical /usr/options File Contents

B.3.31 /usr/spool/cron/crontabs

The **/usr/spool/cron/crontabs** directory contains crontab files for **adm, root,** and **sys** logins. Providing their lognames are in the **/usr/lib/cron/cron.allow** file, users can establish their own **crontabs** file using the **crontab** command. If the **cron.allow** file does not exist, the **/usr/lib/cron/cron.deny** file is checked to determine if the user is denied the use of the **crontab** command.

As **root,** you can either use the **crontab**(1) command or edit the appropriate file under **/usr/spool/cron/crontabs** to make the desired entries. Revisions to the file take effect at the next reboot. The line entry format of a **/usr/spool/cron/crontabs/***logname* file is as follows:

minute hour day month day-of-week command

The various fields of a **crontabs/***logname* line entry are the following:

minute The minutes field is a one- or two-digit number in the range 0 through 59.

hour The hour field is a one- or two-digit number in the range 0 through 24.

day The day field is the numerical day of the month in the range 1 through 31.

month The month field is the numerical month of the year in the range 1 through 12.

day-of-week The day-of-week field is the numerical day of the week where Sunday is 0, Monday is 1, . . . and Saturday is 6.

command The command field is the program or command that is executed at the time specified by the first five fields.

The following syntax applies to the first five fields:

- Two numbers separated by a minus indicates an inclusive range of numbers between the two specified numbers.
- A list of numbers separated by commas specifies all of the numbers listed.
- An asterisk specifies all legal values.

In the command field (sixth field), a percent sign (%) is translated to a new-line character. Only the first line of a command field (character string up to the percent sign) is executed by the shell. Any other lines are made available to the command as standard input.

Figure B-17 shows a typical **/usr/spool/cron/crontabs/***logname* file. The data shown are the **root** file. The file entries support the **calendar** reminder service and basic networking. Remember, you can use the **cron** function to decrease the number of data terminal driven system administration tasks: include recurring and habitual tasks in your crontab file.

(Refer to the **crontab**(1) manual page in the *User's Reference Manual* for additional information.)

```
0 1 * * * /usr/bin/calendar -
41,11 * * * * /usr/lib/uucp/uudemon.hour > /dev/null
45 23 * * * ulimit 5000; /bin/su uucp -c "/usr/lib/uucp/uudemon.cleanup"
  > /dev/null 2>&1
1,30 * * * * /usr/lib/uucp/uudemon.poll > /dev/null
```

Figure B-17. Typical /usr/spool/cron/crontabs/root File

Appendix C

GENIX V.3 SYSTEM ERROR MESSAGES

C.1 INTRODUCTION

GENIX V.3 system error messages are divided into three severity classes: **NOTICE, WARNING,** and **PANIC.** When an error message is displayed, its severity class is displayed as the first part of the message. The following GENIX V.3 system error message tables are divided into severity classes. A description of each severity class is given with each table.

C.2 GENIX V.3 SYSTEM NOTICE MESSAGES

NOTICE error messages provide information on the system status. These messages can sometimes help you to anticipate problems before troubles occur. These error messages are defined alphabetically in the following table.

ERROR MESSAGE (Prefaced by NOTICE)	DESCRIPTION/ACTION
bad block on DSDU floppy drive #, slice #	An out of range block number was specified. Check the file system. (Procedure 5.3)
bad block on DSDU Winchester drive #, slice #	An out of range block number was specified. Take the system to single-user mode (Procedure 3.3), and check the file system (Procedure 5.3).
bad count on DSDU floppy drive #, slice #	A bad count in the super block. Check the file system (Procedure 5.3)
bad count on DSDU Winchester drive #, slice #	A bad count in the super block. Take the system to single-user mode (Procedure 3.3), and check the file system (Procedure 5.3).
Bad free count on DSDU floppy drive #, slice #	The free list count is inconsistent. Check the file system (Procedure 5.3).
Bad free count on DSDU Winchester drive #, slice #	The free list count is inconsistent. Take the system to single-user mode (Procedure 3.3), and check the file system (Procedure 5.3).
bn = # er = #, #	A device error occurred during a read/write operation. Log that the message occurred. No action is required unless the problem persists.
Can't allocate message buffer	System is out of message buffers. Either try again later, or send fewer inter-process communication messages.
cannot allocate MOD page	System is out of available memory. It will probably crash. Either reconfigure kernel to be smaller with fewer features or add more memory.

ERROR MESSAGE (Prefaced by NOTICE)	DESCRIPTION/ACTION
Configured value of NOFILES (#) is greater than max (#) NOFILES set to #.	The value of NOFILES in **/etc/master.d/kernel** exceeds the allowed maximum. No immediate action required. Maximum value has been used. To avoid repetitions on future config boots, change **/etc/master.d/kernel** and run **config.**
Configured value of NOFILES (#) is less than min (#) NOFILES set to #.	The value of NOFILES in **/etc/master.d/kernel** is less than the allowed minimum. No immediate action required. Minimum value has been used. To avoid repetitions on future config boots, change **/etc/master.d/kernel** and run **config.**
Device Error bn = # er = #, #	Device error occurred during a read/write. Log that the error message occurred. Call your service representative if the problem persists.
File table overflow	The system file access table has overflowed. Reconfigure the system with larger NFILE turnable parameter if this is a frequent problem. Call your service representative if the problem continues.
iaddress > 2 24	Block number in an i-node was found to be greater than permissible when updating the file control block. Take the system to single-user mode (Procedure 3.3), and check the file system (Procedure 5.3). Call your service representative if you cannot resolve the problem.
str - Insufficient memory to allocate # pages - system call failed	*Str* may be *growreg, exec, fork* or other kernel function name. Try again later.
no space on DSDU floppy drive #, slice #	The disk has reached its capacity.
no space on DSDU Winchester drive #, slice #	Clean up file system indicated by the partition number. Repartition if necessary.
Out of inodes on DSDU floppy drive #, slice #	Backup the file system (Procedure 5.4). Run Procedure 5.1 increasing the number of inodes. (This destroys what is on the disk)
Out of inodes on DSDU Winchester drive #, slice #	Backup the file system (Procedure 5.4). Run Procedure 5.2 increasing the number of inodes. (This destroys what is on the disk)
page read error on DSDU floppy drive #, slice #	Probable bad diskette. Try a different diskette. If the problem persists, call your service representative.
page read error on DSDU Winchester drive #, slice #	An I/O error has occurred while trying to read. Probable fault in a page from a file. Take the system to single-user mode (Procedure 3.3) and run **hdefix.**
shmctl - couldn't lock # pages into memory	Memory overcommitted. Try again later.
stray interrupt at #	No action required.
str - swpuse count overflow	More than 256 processes are sharing the same page of swap. A copy has been made. No action required.

Error Message (Prefaced by NOTICE)	Description/Action
swapdel - too few free pages	An attempt to delete a swap file has failed because it would result in too little remaining space. No action required.
swap space running out: Needed # blocks	The system had to do more work than normal to swap out a process. If this occurs frequently, try to run fewer simultaneous processes. The computer may hang in these circumstances.
tune.t_maxfc reduced to #	The tunable parameter **maxfc** was found to be greater that the system-imposed limit. It has been automatically reduced to that limit. When convenient, correct kernel master file, reconfigure and reboot.
tune.t_maxsc reduced to #	The tunable parameter **maxsc** was found to be greater that the system-imposed limit. It has been automatically reduced to that limit. When convenient, correct kernel master file, reconfigure and reboot.
useracc - couldn't lock page	Insufficient main memory available to lock a user data page in memory to service a read or write system call to a raw device. Reduce system load, reduce size of raw I/O buffer in user program, or add more memory.

C.3 GENIX V.3 SYSTEM WARNING MESSAGES

WARNING error messages indicate that the GENIX V.3 system may stop functioning if corrective action is not taken. These error messages are defined alphabetically in the following table.

ERROR MESSAGE (Prefaced by WARNING)	DESCRIPTION/ACTION
HDE queue full, following report not logged	The hard disk error logger is full. No more errors can be logged. Save the un-logged report and add it manually to the disk error log. See Section 5.6.
hdeeqd: major(ddev) = #) > = (cdevcnt = #)	Log that the error message occurred. The **hdelogger** found a bad block and logged it. See Section 5.6.
iget - inode table overflow	Out of free slots in the inode table. Too many files open at once. Reduce number of simultaneous processes, or increase number of inode table entries.
	If recurring problem, reconfigure the system with a bigger i-node table (NINODE).
mfree map overflow #. Lost # items at #	If recurring problem, reconfigure the system with a larger core map size (CMAPSIZ).
No kernel virtual space. size= #, mode= #, base=#	If the problem persists, take dump and reboot.
No swap space for exec args	Run fewer simultaneous processes or repartition the disk with increased swap space.
power fail	System has lost power.
Region table overflow	Each text, data, stack, and shmem process segment requires one entry in the region table. The system call that tried to allocate another region failed. Reduce the number of processes, or increase the number of region table overflows (NREGION).
swap space runing out. Needed # pages.	A process was left in memory because there was no room to swap it out. It will be swapped out if room becomes available. More room can be made by killing some processes. Run fewer simultaneous processes to avoid this problem. The computer may hang in these circumstances.
too few HDE equipped disk slots	Log that the error message occurred. Reboot the system. Contact your service representative if the problem recurs.
unreadable CRC hard disk error: maj/min = #/# block = #	Log that the error message occurred. Reboot the system. The **hdelogger** logged a disk access error. Refer to Chapter 5, Disk Management, for additional information.
... on bad dev #(8)	Log that the error message occurred. A file system problem is indicated. Reboot the system. Contact your service representative if the problem recurs.

C.4 GENIX V.3 SYSTEM PANIC MESSAGES

PANIC error messages indicate a problem severe enough that the GENIX V.3 operating system must stop. The cause is usually a hardware problem or a minor problem in the kernel software. Some file systems may be corrupted, but the GENIX V.3 system checks for this when it is restarted. As with most sophisticated computer systems, PANICs will occasionally occur; they should not cause much concern. If a particular PANIC error message occurs repeatedly (or predictably), you should contact your service representative. These error messages are defined alphabetically in the following table.

NOTE: After the panic completes, take a crash dump if desired (see Procedure 3.7 for information about how to do that). Reboot the GENIX V.3 system (see Procedure 3.6). If panics occur frequently, contact your service representative.

ERROR MESSAGE (Prefaced by PANIC)	DESCRIPTION/ACTION
blkdev	The system description file may be incorrect.
bumprcnt — region count list overflow	Ran out of region count entries.
devtab	The list header for the chain of buffers attached to the block-type device cannot be found.
getpages — pbremove	The kernel was attempting to remove a page from the page cache but could not find it in the cache. Probable software bug.
iget — mounted on inode not in mount table.	Inode has mount flag set but not in mount table. Probable software bug.
i/o error in swap	An access error occurred on the swap device. Check the hard disk error log.
illegal SIT counter selected	An illegal Sanity Interval Timer(SIT) counter was selected.
iput — bad mount count	The count of the number of inodes in use on a particular file system is incorrect. Probable software bug.
iupdat — fifo iaddress > 2 24	Block number in inode is too big.
iupdat — iaddress > 2 24	Block number in inode is too big.
KERNEL BUS TIMEOUT	A bus request by the system was not fulfilled within the allotted time.
KERNEL DATA ALIGNMENT ERROR	The system software attempted to execute an instruction using an odd number as a pointer to a 16-bit quantity or a number that is not a multiple of 4 as a pointer to a 32-bit quantity.
KERNEL MMU FAULT ...	The Memory Management Unit (MMU) acknowledged a fault during the execution of an instruction while in kernel mode.
KERNEL MMU FAULT #	A bus request by the system was not fulfilled within the allotted time.
KERNEL MODE ... FAULT	The processor unexpectedly registered the error given by
KERNEL MODE FAULT, FT=#, ISC=#	The processor unexpectedly registered an error, identified by Fault Type (FT) and internal State Code (ISC).
KERNEL MODE *str* FAULT	The processor unexpectedly registered the error given by *str*. These errors are detected by the module and are listed in fault type class 3.
kernel process stack exception	A stack reference caused a memory fault.
Krnlflt returned to k_trap	Take a sysdump and reboot.
kseg — sptalloc failed	No physical memory available for server when needed.
main — copyout of icode failed	The kernel was not able to copy the assembly code which is used to start up the system.

Error Message (Prefaced by PANIC)	Description/Action
main — swapadd failed	The kernel was not able to attach to the first swap area. Check to see if there is a swap area available on the boot disk.
newproc -- fork failed	The kernel was not able to create one of the kernel processes while booting. Check tunables.
newproc — no procs	The kernel ran out of process table slots while creating kernel processes at boot time. Check value of NPROCS.
no fs	No file system or super-block was found.
no imt	No indirect mount point was found in the mount table.
no procs	A process table entry was not found during a fork when an entry is available.
not a valid root	The root file system magic number is incorrect or the root device is improperly specified.
pinsert — pinsert dup	The kernel was attempting to add a page to the page cache but the page was found to be in the cache already. Probable kernel bug.
setrq — proc on q	The kernel was attempting to add a process to the run queue but the process was found to be on the queue already.
srmount -- cannot mount root	The kernel was not able to mount the root file system when booting.
srmount -- not a valid root	When the root file system was being mounted during the kernel boot, it did not contain the correct "magic number". Boot off another disk and run **fsck** on root.
startup - no memory	The amount of available memory, after the kernel is loaded, is less than tune.t_minarmem. Correct kernel master file, reconfigure and reboot.
svirtophys -- movtrw failed	The "movtrw" instruction failed to convert a virtual to a physical address. Check to see if the address was within range.
swapin lost text	The shared text table entry pointer is zero or an attempt was made to link to text owner process.
swapseg - i/o error in swap	An I/O error occurred during a transfer to or from the swap area.
sys32000 — DELMEM premove failed	An attempt to remove memory from a system's available memory failed. Kernel bug. Do not remove so much memory.
Timeout table overflow	The queue for timeout requests has overflowed while attempting to add another entry.
uballoc — ptmemall failed for u block	System failed to allocate page table for user area. Shouldn't happen, since space for user area page table is reserved in proc structure.

Error Message (Prefaced by PANIC)	Description/Action
unexpected user stack fault, ISC=#	User stack fault occurred which was neither a stack bound or page fault. Probably due to an interrupt vector ID fetch fault. Check interrupt vector table (beginning at virtual location 140) and hardware configuration.
unknown level in cmn_err (level=#, msg = ...)	The common error software was invoked to process an error, but given an invalid severity level. This problem is secondary; the original problem is given by the msg.
vfault -- bad dbd_type	The page being faulted is not a recognized type (? demand fill, demand zero, in file or on swap).
xalloc — bad magic	An invalid magic number was found in an a.out header during an exec system call. This should have been detected earlier by the kernel so this is a probable kernel bug.
xalloc lost text	A pointer to the process text table points to a bad address.

GLOSSARY

address — a number, label, or name that indicates the location of information in the computer's *memory*.

advertise — a means of making *resources* available from a local *host* to other *hosts* in a *Remote File Sharing* environment.

a.out — the default name of a freshly compiled *object file*, pronounced 'A-dot-out'; historically a.out signified assembler output.

archive — 1. a collection of data gathered from several *files* into one file. 2. especially, such a collection gathered by **ar**(1) for use as a library.

automatic calling unit — a hardware *device* used to dial stored telephone numbers; allows the system to contact another system over phone lines without manual intervention.

bad block — a section of a storage medium which cannot store data reliably.

block — the basic unit of *buffering* in the *kernel*, 1024 bytes; see *indirect*, *logical*, and *physical blocks*.

block device — a *device* upon which a *file system* [1] can be *mounted*, typically a permanent storage device such as a tape or disk drive, so called because data transfers to the device occur by *blocks*; cf. *character device*.

boot — to start the operating system, so called because the *kernel* must bootstrap itself from secondary storage into an empty machine. No *login* [3] or *process* persists across a boot. **Boot block** the first block of a *file system* [1], which is reserved for a booting program.

boot program — loads the *operating system* into *core*.

buffer — 1. a staging area for input-output where arbitrary-length transactions are collected into convenient units for system operations; the *file system* [3] uses buffers, as does *stdio*. 2. to use buffers.

buffer pool — a region of store available to the *file system* [3] for holding *blocks*; all but *raw* [2] input-output for *block devices* goes through the buffer pool so read and write operations may be independent of device blocks.

cartridge tape — a storage medium that consists of a magnetic tape wound on spools housed in a plastic container.

character device — a *device* upon which a *file system* [1] cannot be *mounted* such as a terminal or the *null device*.

child process — see *fork*.

client — a *host* that has *mounted* an *advertised resource* from another *host* in a *Remote File Sharing* environment.

command — 1. an instruction to the *shell*, usually to run a *program* [1] as a *child process*. 2. by extension, any *executable file*, especially a *utility program*.

command file — same as *shell script*.

configuration — the arrangement of the software or hardware of a system, peripheral, or network as defined by the nature, number, and chief characteristics of its functional units.

controller — a *device* that directs the transmission of data over the data links of a *network*.

core file — a *core image* of a terminated *process* saved for debugging; a core file is created under the name 'core' in the *current directory* of the process.

core image — a copy of all the *segments* of a running or terminated program; the copy may exist in main store, in the *swap area*, or in a *core file*.

crash — If a hardware or software *error* condition develops that the system can't handle, it takes itself out of service, or crashes. Such conditions occur when the system can't allocate resources, manage *processes*, respond to requests for system functions, or when the electrical power is unstable.

cron — a command which creates a daemon that invokes commands at specified dates and times.

cylinder — the set of all *tracks* on a *disk* which are the same distance from the axis about which the disk rotates.

daemon — a *background* process, often perpetual, that performs a system-wide public function, *e.g.*, **calendar** (1) and **cron** (8); the affected spelling is an ancient legacy.

destination — the remote system that will ultimately receive a *file* transferred over a *network*.

device — 1. a *file* [2] that is not a *plain file* or a *directory*, such as a tape drive, or the *null device*; a *special file*. 2. a physical input-output unit.

diagnostic — a message printed at your terminal that identifies and isolates *program* errors.

directory — a *file* that comprises a catalog of *filenames* [2]; the organizing principle of the *file system* [2], a directory consists of *entries* which specify further *files* (sense 2, including directories), and constitutes a node of the *directory tree*.

directory entry, entry 1. an association of a name with an *inode number* appearing as an element of a *directory*. 2. the name part of such an association.

directory hierarchy the tree of all *directories*, in which each is reachable from the *root* via a chain of *subdirectories*.

directory tree same as *directory hierarchy*.

disk a platter coated with magnetic material on which data can be stored.

diskette a magnetic storage medium which is smaller and more flexible than a hard *disk*.

domain a logical grouping of *hosts* in a *Remote File Sharing* environment. Each *host* in a *domain* relies on the same *domain name server*(s) for certain *resource* sharing and security services. Each *domain* has one *primary* and zero or more *secondary domain name servers*.

domain name server a computer that creates and maintains the following information for *hosts* in a *Remote File Sharing* domain: *advertised resources*, *host* names and passwords, names and addresses for name servers of other domains (optional), *host* user and group information used for *ID mapping* (optional).

drive the hardware device that holds magnetic disks, diskettes, and tapes while they are in use.

dump a copy of the *core image* of the operating system.

environment 1. a set of strings, distinct from the *arguments*, made available to a *process* when it *executes* [2] a *file*; the environment is usually inherited across *exec*(2) operations. 2. a specific environment [2] maintained by the *shell*. 3. a nebulously identified way of doing things, as in 'interactive environment': a deprecated usage, not always expunged from these manuals.

error occurs when a hardware or software condition prevents the successful *execution* of a system or a user *process*.

error message a message sent from the system to the *system console* when an *error* occurs.

exec a system call which allows the user to request the execution of another program.

executable file 1. an *object file* that is ready to be copied into the *address* space of a *process* to run as the code of that process. 2. a file that has execute *permission*, either an *executable file* [1] or a *shell script*.

execute 1. informally, to run a *program*. 2. to replace the *text segment* and *data segments* of a *process* with a given *program* [1].

FIFO a named permanent *pipe* which allows two unrelated *processes* to exchange information using a pipe connection.

file — 1. in general, a potential source of input or destination for output. 2. most specifically, an *inode* and/or associated contents, *i.e.*, a *plain file*, a *special file*, or a *directory*. 3. a *directory entry*; several directory entries may name the same file [2]. 4. most loosely, a *plain file*.

file descriptor — a conventional integer quantity that designates an *open file*.

filename — 1. a *pathname*. 2. the last component name in a pathname.

file system — 1. a collection of *files* that can be *mounted* on a block *special file*; each file of a file system appears exactly once in the *i-list* of the file system and is accessible via some *path* from the *root* directory of the file system. 2. the collection of all *files* on a computer. 3. the part of the kernel that deals with file systems [1].

filter — a *program* [1] that reads from the *standard input* and writes on the *standard output*, so called because it can be used as a data-transformer in a *pipeline*.

firmware — the memory which permanently holds a few, special programs.

floppy key — a copy of the default *firmware* password for a *Series 32000* computer on a *diskette*. It may be used to reset the password to its original value.

flush — to empty a *buffer*, for example to throw away unwanted input-output upon *interrupt* or to release output from the clutches of *stdio*.

fork — to split one *process* into two, the **parent process** and **child process**, with separate, but initially identical, *text*, *data*, and *stack segments*.

formatting — the process of imposing an addressing scheme on a *disk*. This includes the mapping of both sides of the disk into *tracks* and *sectors*.

free list — in a *file system* [1], the list of *blocks* that are not occupied by data.

getty — one of a series of *processes* which connect the user to the GENIX V.3 system. *Getty* is invoked by *init*, and in turn invokes *login*.

group — 1. a set of *permissions* alternative to *owner* permissions for access to a *file*. 2. a set of *userids* that may assume the privileges of a group [1]. 3. the *groupid* of a file.

groupid — an integer value, usually associated with one or more *login names*; as the *userid* of a process becomes the *owner* of files created by the process, so the groupid of a process becomes the *group* [3] of such files.

hole — a gap in a *plain file* caused by *seeking* while writing; **read**(2) takes data in holes to be zero; a *block* in a hole occupies no space in its *file system*.

host — a computer that is configured to share *resources* in a *Remote File Sharing* environment.

ID mapping — a means of setting the permissions that each remote user and group will have for a *host*'s *advertised resources* in a *Remote File Sharing* environment.

i-list — the index to a *file system* [1] listing all the *inodes* of the file system; cf. *inode number*.

indirect blocks — data blocks that are not directly referenced by a *inode* (because the file is larger than 10 1024-byte blocks); the inode has 3 *addresses* that indirectly reference (by a cascade of pointers) some 2,114,114 data blocks (an extremely large potential *file* size). the inode has 1 address that points to 128 more data blocks; a second address that points to 128 blocks that each point to 128 data blocks; and finally a third address that points to 128 blocks each of which point to another 128 blocks, each of which point to 128 data blocks!

init — a general *process* spawner which is invoked as the last step in the *boot* procedure; it regularly checks a table that defines what processes should run at what *run level*.

inode — an element of a *file system* [1]; an inode specifies all properties of a particular *file* [2] and locates the file's contents, if any.

inode number, i-number — the position of an *inode* in the *i-list* of a *file system* [1].

instruction — see *address*.

integrity — in a *file system*, the quality of being without errors due to *bad blocks*.

interface programs — *shell scripts* furnished with the LP *spooling* software which interface between the user and the printer.

interrupt — 1. a *signal* that normally terminates a *process*, caused by a break or an interrupt character. 2. a signal generated by a hardware condition or a peripheral *device*. 3. loosely, any *signal*.

IPC — an acronym for interprocess communication.

kernel — the GENIX V.3 system proper; resident code that implements the *system calls*.

kernel address space — a portion of memory used for data and code addressable only by the *kernel*.

line discipline — a module to handle protocol or data conversion for a *stream* [2]. A line discipline, unlike a *filter*, is part of the *kernel*.

link — 1. to add an entry for an existing *file* to a directory; converse of *unlink*. 2. by extension, a *directory entry*. 3. loosely, any but one putatively primary directory entry for a given *inode*; either linked [1] or a *symbolic link*.

link count — the number of *directory entries* that pertain to an *inode*; a *file* ceases to exist when its link count becomes zero and it is not *open.*

load device — designates the physical *device* from which a program will be loaded into main *memory.*

log files — contain records of transactions that occur on the system; software that *spools*, for example, generates various log files.

logical block — a unit of data as it is handled by the software; the GENIX V.3 system handles data in 1024-byte logical blocks.

login — 1. the *program* that controls logging in. 2. the act of *logging in.* 3. by extension, the computing session that follows a login [2].

memory — 1. same as *memory image.* 2. physical memory represents the available space in main memory; *programs* are either *swapped* or *paged* into physical memory for *execution.* 3. virtual memory management techniques permit *programs* to treat *disk* storage as an extension of main memory.

memory image — same as *core image.*

mode, file mode — the *permissions* of a *file*; colloquially referred to by a 3-digit octal number, *e.g.*, 'a 755 file'; see **chmod**(1).

mount — to extend the *directory hierarchy* by associating the *root* of a *file system* [1] with a *directory entry* in an already mounted file system; converse is unmount, spelled 'umount'.

namelist — same as *symbol table.*

network — the hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals.

networking — for computer systems, means sending data from one system to another over some communications medium (coaxial cable, phone lines, etc.). Common networking services include *file* transfer, remote *login*, remote *execution.*

node name — an up-to-six character name for the system; used as the official name of the machine in a *network.* The node name resides in the NODE parameter.

null device — a *device* [1] that always yields *end of file* on reading and discards all data on writing.

object file — a *file* of machine language code and data; object files are produced from source programs by compilers and from other object files and libraries by the link editor; an object file that is ready to run is an *executable file* [1].

operating system — the *program* for managing the resources of the computer. It takes care of such things as input/output procedures, process scheduling, the file system, removing this burden from user programs.

open file — 1. the destination for input or output obtained by *opening* a *file* or creating a *pipe*; a *file descriptor*; open files are shared across *forks* and persist across *executes* [2]. 2. loosely, a file that has been opened, however an open file [1] need not exist in a *file system* [1], and a file [2] may be the destination of several *open files* simultaneously.

other — 1. a set of *permissions* regulating access to a *file* by processes with *userid* different from the *owner* and *groupid* different from the *group* of the file. 2. the customary name of the default *group* [2] assigned upon *login*.

owner — the *userid* of the *process* that created a *file*; the owner has distinctive *permissions* for a file.

page — a fixed length block of data that has a virtual *address*, and that can be transferred between main and secondary storage.

paging — the process by which *programs* are truncated into *pages* and transferred between main and secondary storage by the virtual handler (or paging *daemon*).

parent process — see *fork*.

partitions — units of storage space on disk.

path, pathname — a chain of names designating a *file*; a **relative pathname** leads from the current directory, for example, a path to *directory* A, thence to directory B, thence to *file* C is denoted A/B/C; a **full pathname** begins at the *root*, indicated by an initial '/', as in /A/B/C.

permission — a right to access a *file* in a particular way; read, write, execute (or look up in, if a directory); permissions are granted separately to *owner*, *group*, and *others*. **permission bit** a permission, so called because each permission is encoded into one bit in an *inode*.

physical block — a unit of data as it is actually stored and manipulated; the GENIX V.3 system handles data in 1024-byte physical blocks.

physical — see *memory*.

pipe — a direct stream connection between *processes*, whereby data written on an *open file* in one process becomes available for reading in another.

pipeline — a sequence of *programs* [1] connected by *pipes*.

polling — the interrogation of *devices* by the *operating system* to avoid contention, determine operation status, or ascertain readiness to send or receive data.

ports — the point of physical connection between a peripheral *device* (such as a terminal or a printer) and the device *controller* (ports board), which is part of the computer hardware.

primary name server the computer on which administration for a *Remote File Sharing domain* is performed.

process a connected sequence of computation; a process is characterized by a *core image* with instruction location counter, *current directory*, a set of *open files, control terminal, userid,* and *groupid.*

process id an integer that identifies a *process.*

process number same as *process id.*

profile 1. an optional *shell script*, '.profile', conventionally used by the *shell* upon *logging in* to establish the *environment* [3] and other working conditions customary to a particular user. 2. to collect a histogram of values of the instruction location counter of a *process.*

program 1. an *executable file.* 2. a *process.* 3. all the usual meanings.

queue a line or list formed by items in a system waiting for service.

raw device a *block device,* read and write operations to which are not *buffered,* and are synchronized to natural records of the physical *device.*

reboot same as *boot.*

region 1. a group of machine *addresses* that refer to a base address.
2. a collection of executable code or data contiguous within a process' virtual address space. A process must have a "text," "data," and "stack" region.

release a distribution of fixes or new functions for an existing software product.

Remote File Sharing a software utilities package that enables computers to share *resources* across a network.

resource a directory that is *advertised* in a *Remote File Sharing* environment. When a *resource* is *mounted* on a *client,* the contents of the directory (files, devices, and named pipes) and any of its subdirectories are potentially available to users on the *client.*

re-tension the process of re-winding the tape in a *cartridge tape device* to make sure it is at the correct tautness for accurate recording of data.

root 1. a distinguished directory that constitutes the origin of the *directory hierarchy* in a *file system* [1]. 2. specifically, the origin for the *file system* [2], with the conventional *pathname* '/'. 3. the origin of the directory hierarchy in a *file system* [1].

rotational gap the gap between the actual *disk* locations of blocks of data belonging to the same *file;* the rotational gap compensates for the continuous, high-speed rotation of the disk so that when the controller is ready to reference the next physical block the read-

write head is positioned correctly at the beginning of that block.

run level — a software *configuration* of the system which allows a particular group of *processes* to exist.

schedule — to assign resources— main store and CPU time—to *processes.*

scheduler — a permanent *process*, with *process number* 1, and associated *kernel* facilities that does scheduling.

search path — in the *shell*, a list of *pathnames* of *directories* that determines the meaning of a *command*; the command name is prefixed with members of the search path in turn until a pathname of an *executable file* [2] results; the search path is given by the shell variable PATH.

secondary name server — a *host* that is configured to take over *domain name server* responsibilities temporarily in case the *primary name server* goes down.

section, sector — A 512-byte portion of a *track* which can be accessed by the magnetic disk heads in the course of a predetermined *rotational* displacement of the storage device.

segment — a contiguous range of the address space of a *process* with consistent store access capabilities; the four segments are (i) the **text segment**, occupied by executable code, (ii) the **data segment**, occupied by *static* data that is specifically initialized, (iii) the **bss segment**, occupied by static data that is initialed by default to zero values, and (iv) the **stack segment**, occupied by *automatic* data, see *stack*; sometimes (ii), (iii), and (iv) are collectively called data segments.

semaphore — an IPC facility which allows two or more processes to be synchronized.

server — a *host* that is actively sharing one of its *advertised resources* with another *host* in a *Remote File Sharing* environment.

set userid — a special *permission* for an *executable file* [1] that causes a *process* executing it to have the access rights of the *owner* of the file; the owner's *userid* becomes the **effective userid** of the process, distinguished from the **real userid** under which the process began.

set userid bit — the associated *permission bit.*

shared memory — an IPC facility which allows two or more processes to share the same data space.

shell — 1. the program *sh*(1), which causes other programs to be executed on *command*; the shell is usually started on a user's behalf when the user *logs in.* 2. by analogy, any program started upon logging in.

shell script an executable *file* of *commands* taken as input to the *shell.*

signal an exceptional occurrence that causes a *process* to terminate or divert from the normal flow of control; see *interrupt, trap.*

single-user a state of the operating system in which only one user is supported.

source file 1. the uncompiled version of a *program.* 2. generally, the unprocessed version of a *file.*

special file an *inode* that designates a *device,* further categorized as either (i) a **block special file** describing a *block device,* or (ii) a **character special file** describing a *character device.*

spool to collect and serialize output from multiple *processes* competing for a single output service.

spool area a *directory* in which a spooler collects work.

spooler a *daemon* that spools.

stack a *segment* of the *address* space into which *automatic* data and subroutine linkage information is allocated in last-in-first-out fashion; the stack occupies the largest data addresses and grows downward towards *static* data.

standard error one of three files described below under *standard output.*

standard input the second of three files described below under *standard output.*

standard output *open files,* customarily available when a *process* begins, with *file descriptors* 0, 1, 2 and *stdio* names 'stdin', 'stdout', 'stderr'; where possible, utilities by default read from the standard input, write on the standard output, and place error comments on the standard error file. Initially, all three of these files default to your terminal.

startup same as *boot.*

sticky bit a *permission* flag that identifies a file as a *sticky file.*

sticky file a special *permission* for a *shared text* file that causes a copy of the *text region* to be retained in the *swap area* and/or memory to improve system response.

super block the second *block* in a *file system* [1], which describes the allocation of space in the file system; cf. *boot block.*

userid 0, which can access any *file* regardless of *permissions* and can perform certain privileged *system calls, e.g.,* setting the clock.

swap to move the pages of an executing program between main and secondary storage to make room for other *processes.*

swap area the part of secondary store to which pages are *swapped*; the swap area is disjointed from the *file system.*

symbolic link an *inode* that contains the *pathname* of another. References to the symbolic link become references to the named inode.

symbol table information in an *object file* about the names of data and functions in that file; the symbol table and *address* relocation information are used by the link editor to compile *object files* and by debuggers.

System Administration when capitalized, refers to the package of screens and interactive prompts, invoked through the **sysadm**(1) command, that help you accomplish most system administration tasks.

system calls 1. the set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated. 2. the system primitives invoked by user *processes* for system-dependent functions, such as I/O, process creation, etc.

system console the directly connected terminal used for communication between the operator and the computer.

system name an up-to-six character name for the system; resides in the SYS parameter.

table an array of data each item of which may be uniquely identified by means of one or more arguments.

text file, a *file*, the bytes of which are understood to be in ASCII code.

track an addressable ring of *sections* on a *disk* or *diskette*; each disk or diskette has a predefined number of concentric tracks, which allows the disk head to properly access *sections* of data.

trap a method of detecting and interpreting certain hardware and software conditions via software; a trap is set to catch a *signal* (or *interrupt*), and determine what course of action to take.

tunable parameters variables used to set the sizes and thresholds of the various control structures of the *operating system.*

tuning 1. modifying the *tunable parameters* so as to improve system performance. 2. the reconfiguration of the *operating system* to incorporate the modifications into *executable* version of the system.

userid an integer value, usually associated with a *login name*; the userid of a *process* becomes the *owner* of files *created* by the process and descendent (*forked*) processes.

utility, utility program a standard, generally useful, permanently available *program.*

version a separate *program* product, based on an existing one, but containing significant new code or new functions.

virtual memory see *memory.*

INDEX

A

Accessing a file 6-11
  create 6-13
  open 6-12
  pathname conversion 6-14
  read 6-13
  used by more than one process 6-14
  write 6-13
Add another LP printer P7-3
Add groups, user services P2-1
Add new domains, RFS 11-34
Add new hosts, RFS 11-33
Add users, user services P2-1
Administration, RFS 11-4
Administrative commands, LP 8-3
Administrative directories 4-2
Administrative files 4-2
Administrative programs, networking 10-3
  uucheck 10-3
  uucleanup 10-3
  uulog 10-3
  Uutry 10-3
Adminstrative files, networking 10-23
adv command, RFS 11-1, 11-8
Advertise resources, RFS 11-8, P10-14
Algorithm, fsck 6-34
Aliases resources, RFS 11-9
Anticipating user requests 3-9
Automated networking maintenance P9 12
Automatic remote mounts, RFS 11-13

B

Backup
  file system 6-23, P5-13
  file system, complete 6-23, P5-14
  file system, incremental 6-24, P5-16
  file system, restore 6-26, P5-20
  file system, schedule reminders 6-25, P5-21
  file system, selective 6-25, P5-18
Bad blocks 5-8
  data loss, dealing with 5-12
  detected 5-9
  detecting new 5-11
  disk identification 5-11
  fix 5-8
  handling feature 5-7
  handling screnario 5-10
  logging 5-11
  mapped, cannot be 5-9
  normal operation 5-9
  numbers, fsck 6-33
  questions about 5-9
  reporting 5-11
  single user mode 5-12
  unreliable 5-8
Bad I-Node number 6-35
Basic domain setup, configuration, RFS P10-2
Basic domain setup, prerequisites, RFS P10-2
Basic domain setup, RFS P10-2
Basic host setup, configuration, RFS P10-9
Basic host setup, prerequisites, RFS P10-8
Basic host setup, RFS P10-8
Basic networking debugging P9-14
Basic networking maintenance P9-11
batch command 7-10
  shift workload to off-peak hours 7-10
baud rate, TTY 9-1
Block 0 6-3
Block 1 6-3
Block devices 5-4
Blocks and cylinder recommendations 6-16

C

Change commands 4-20
  firmware monitor information 4-20
Change host password, RFS 11-31
Changing passwood entries 3-2
Character devices 5-4
Check basic information, networking P9-15
Check blocks and sizes 6-37
Check connectivity, phase 3 6-41
Check error messages, networking P9-15
Check for faulty ACU/modem, networking P9-14
Check free list, phase 5 6-45
Check path names, phase 2 6-39
Check reference counts, phase 4 6-42
Check systems file, networking P9-14
Checking for file system consistency 6-19
chmod command 2-8
Choosing buffer size 7-3
  improving performance 7-3
Cleanup phase 6-47
Cleanup phase messages 6-47
Commands
  cpio 6-23, 6-26
  du 6-22
  find 6-21, 6-22
  fsck 6-28
  mv 6-21
  networking 10-2
  sample use 6-29
  sync 6-27
  sysadm fileage 6-22
  tail 6-21
Configuration register commands
  firmware monitor information 4-20
Connect security, RFS 11-5
Connect security, RFS, group 11-7
Connect security, RFS, mount 11-5
Connect security, RFS, user 11-7
Console terminal configuration, security P1-1
cpio command 6-23, 6-26
CPU time, RFS 11-42

Create, accessing a file 6-13
Creating a file system 6-15
cron, networking P9-12
ct command, networking 10-2
cu command, networking 10-2
Cylinder and blocks recommendations 6-16

D

daemons, networking 10-3
  uucico 10-3
  uusched 10-4
  uuxqt 10-3
Data commands
  firmware monitor information 4-21
Data loss, bad blocks 5-12
Database, supporting, networking 10-4
date command, security P1-2
dcutest diagnostic 4-25
  Series 32000 defect mapping 4-32
Debug transmissions, networking P9-14
Default Shell 3-7
Defining a new special file 5-4
Defining best system usage patterns 7-6
  improving performance 7-6
Delete groups P2-7
Delete users P2-7
Deleting passwood entries 3-2
Devconfig file, networking 10-22, P9-9
Device names and designators A-1
  diskette partitions A-3
  hard disk default partitions A-3
  System V Release 2.0 and later A-1
Device types 5-1
  diskette drives 5-2
  hard disk 5-1
devicemgmt, networking P9-4
Devices file entries, making, networking 10-26
Devices file, networking 10-5
Diagnostic information 4-25
  dcutest 4-25
  dcutest for Series 32000 defect mapping 4-32
  stand-alone 4-25
Dialcodes file, networking 10-15
Dialers file, networking 10-9
Direct links, networking 10-1, 10-25, 10-26
Directories B-1
  administrative 4-2
  device files 5-3
  /etc/init.d B-6
  /etc/master.d B-8
  /etc/rc0.d B-14
  /etc/rc2.d B-14
  /etc/rc3.d B-17
  /etc/rc.d B-17
  /etc/save.d B-17
  /etc/shutdown.d B-22
  monitoring ones that grow 6-21
  organization, improving performance 7-5
  regular files 5-3
  root 4-2
  run level 4-13
  /usr/news B-26
  /usr/options B-26, B-27
  /usr/spool/cron/crontabs B-29
Directory data blocks 6-34
Directory unallocated 6-34
Disconnected directories 6-35
Disk bottleneck, check for 7-9
  general procedures, samples 7-9
Disk format 6-7
Disk identification 5-11
  bad block 5-11
Disk management 5-1
Disk management procedures P4-1
  duplicate diskettes P4-3
  format diskettes P4-1
  hard disk errors P4-5
Disk partitions
  30-megabyte drive 6-9
Disk space
  monitoring percent used 6-20
Disk usage, monitoring 6-20
Diskette drives 5-2
Diskettes
  create file systems on P5-2
  duplicate P4-3
  format P4-1
  partitions A-3
Disks
  duplicating 5-6
  formatting 5-5
Domain administration, RFS 11-26
Domain advertise table, RFS 11-11
Domain maintenance, RFS 11-32
  add new domains 11-34
  add new hosts 11-33
  change name server 11-33
  init 3 11-34
  init 3, adding scripts 11-37
  init 3, entering 11-35
  init 3, processing 11-35
  init 3, processing, changing 11-36
  monitoring 11-40
  monitoring, CPU time 11-42
  monitoring, remote disk space 11-45
  monitoring, remote system calls 11-40
  monitoring, resource usage 11-44
  monitoring, server processes 11-42
  monitoring, servers, too few 11-42
  monitoring, servers, too many 11-44
  name server, reassign 11-34
  recovery 11-37
  recovery, primary goes down 11-38
  recovery, rfuadmin 11-39
  recovery, rfudaemon 11-39
  recovery, secondary goes down 11-38
  recovery, server goes down 11-39
  remove a host 11-33
  update group files 11-34
  update user files 11-34
Domain mount table, RFS 11-15
Domain name server, RFS
  change current 11-33
  reassign 11-34
Domain setup, RFS P10-1
Domains, RFS 11-3
du commands 6-22
Dumb line printer interface program 8-9
Duplicate blocks 6-33
Duplicating disks 5-6

E

Early initialization 4-11
Environment
  array for a typical user 3-6
  user 3-4
  variables 3-4
Error message abbreviations, fsck 6-36
Error messages C-1
  NOTICE C-1
  PANIC C-5
  phase 1 6-38
  phase 2 6-40
  phase 3 6-42
  phase 4 6-43
  phase 5 6-46
  phase 6 6-47
  type for phase 1 6-37
  types for phase 2 6-39
  types for phase 3 6-41
  types for phase 4 6-42
  types for phase 5 6-45
  WARNING C-4
/etc/inittabl file, changing, networking 10-27

F

File system administration 6-1, 6-2, 6-15
File system administration procedures P5-1
  backup P5-13
  backup, complete P5-14
  backup, incremental P5-16
  backup, restore P5-20
  backup, schedule reminders P5-21
  backup, selective P5-18
  file system, checking P5-9
  file system, create on diskettes P5-2
  file system, create on hard disk P5-5
  file system, maintain P5-9
  monitoring disk usage P5-11
  restore P5-13
File system administration
  accessing a file 6-11
  address chain 6-6
  backup 6-23
  Block 0 6-3
  Block 1, super-block 6 3
  consistency, checking for 6-19, 6-27
  creating 6-15
  free blocks 6-7
  I-Node table 6-10
  I-Nodes 6-4
  maintaining 6-18
  making it available 6-15
  mounting 6-17
  open file table 6-11
  organization 6-1
  restore 6-23
  shell scripts 6-19
  size and I-Node list size 6-31
  storage blocks 6-7
  storage device, relationship 6-7
  system file table 6-10
  tables 6-12
  tables in memory 6-10
  unmounting 6-17
  view of a 6-3
  what can go wrong 6-26
  works, how 6-10
File system device
  relating to a file system name 6-16
File system name
  relating to a file system device 6-16
File system organization
  improving performance 7-5
File system
  backup P5-13
  backup, complete P5-14
  backup, incremental P5-16
  backup, restore P5-20
  backup, schedule reminders P5-21
  backup, selective P5-18
  checking P5-9
  create on diskette P5-2
  create on hard disk P5-5
  maintaining P5-9
  organization, improving performance 7-5
  organization of free list 7-5
  restore P5-13
  restoring good organization 7-5
Files B-1
  administrative 4-2
  defining a new special 5-4
  device 5-3
  /etc/checklist B-3
  /etc/fstab B-3, B-4
  /etc/gettydefs B-4
  /etc/group B-4, B-6
  /etc/inittab B-6, B-7
  /etc/motd B-8
  /etc/passwd B-8, B-10
  /etc/profile B-9, B-11
  /etc/rc0 B-9, B-12
  /etc/rc2 B-14, B-15
  /etc/rc3 B-17
  /etc/rstab B-17
  /etc/shutdown B-17, B-18
  /etc/TIMEZONE B-22
  /etc/utmp B-23
  /etc/wtmp B-23
  gettydefs B-5
  holes in 6-15
  indentifying inactive 6-21
  monitoring ones that grow 6-21
  regular 5-3
  removing inactive 6-21
  used by more than one process 6-14
  /usr/adm/sulog B-23, B-24
  /usr/lib/cron/log B-23, B-25
  /usr/lib/help/HELPLOG B-25, B-26
  /usr/lib/spell/spellhist B-26
  /usr/options B-28
  /usr/spool/cron/crontabs/root B-30
find command 6-21, 6-22
Finding problems, performance management 7-2
Firmware monitor information 4-17
  change commands 4-20
  configuration register commands 4-20
  data commands 4-21
  invocation 4-17
  memory test 1 address parity 4-24

memory test 2 increasing addresses 4-24
memory test 3 decreasing addresses 4-24
memory test 4 double-words 4-24
memory test 5 even words 4-24
memory test 6 even bytes 4-24
memory test 7 load 4-24
MMU register commands 4-20
monitor error messages 4-25
monitor memory organization 4-18
print commands 4-19
program execution commands 4-22
test commands 4-22
test failures 4-23
test patterns 4-22
test sequence 4-22
Fixing problems, performance management 7-2
FLCKREC parameter 7-36
Floppy disk drive A-4
Forced unmount, RFS 11-16
Format, fsck 6-32
Formatting 5-5
disks 5-5
Free blocks 6-7
Free I-Node count 6-31
Free-block count 6-31
Free-block list 6-31
fsck command 6-28
fsck utility 6-28
fsck
algorithm 6-34
bad block numbers 6-33
bad I-Node number 6-35
directory data blocks 6-34
directory unallocated 6-34
disconnected directories 6-35
duplicate blocks 6-33
error message abbreviations 6-36
file system components checked by 6-30
file system size and I-Node list size 6-31
format 6-32
free I-node count 6-31
free-block count 6-31
free-block list 6-31
incorrect . and .. entries 6-35
indirect blocks 6-34
I-Node size 6-33
I-Nodes 6-32
link count 6-32
regular data blocks 6-35
running 6-35
super-block 6-30
type 6-32

G

General errors 6-36
General operating policy 4-1
General procedures, samples
check for disk bottleneck 7-9
check for excess swapping 7-9
check for potential table overflows 7-9
investigating performance problems 7-7
performance management 7-7
shift workload to off-peak hours 7-10
system reconfiguration 7-10
troubleshooting procedures, outline 7-8
getty entries, TTY 9-5
gettydefs entries, TTY 9-3
Group files, RFS 11-21
group ID 3-3
Group security, RFS 11-7

H

Hard disk default partitioning A-4
single A-4
Hard disk default partitions A-3
Hard disk devices 5-1
Hard disk errors, check for P4-5
Hard disk
change partitions 5-5
create file systems P5-5
partitioning 5-5
Hardware failure 6-26
Holes in files 6-15
Host administration, RFS 11-25
Host maintenance, RFS 11-27
start up host 11-28
Host mount table, RFS 11-14
Host password, RFS 11-21
Host setup, RFS P10-8
Human error 6-27
Hunt sequence, TTY 9-1

I

Identifying devices
block 5-4
character 5-4
new special file, defining 5-4
operating system 5-2
Identifying inactive files 6-21
Identifying large space users 6-22
idload command, RFS 11-20
Important systems files 4-3
Improving disk utilitization
improving performance 7-3
Improving performance
choosing buffer size 7-3
defining best system usage patterns 7-6
directory organization 7-5
file system organization 7-5
improving disk utilization 7-3
modifying the tunable configuration parameters 7-3
organization of file system free list 7-5
ps command 7-6
restoring good file system organization 7-5
setting text-bit (sticky-bit) 7-4
user $PATH variables 7-7
Incorrect . and .. entries 6-35
Indirect blocks 6-34
init 3
entering 11-35
processing 11-35
init
controls the system state 4-6
Initialization, early 4-11
Initialization phase 6-35
inittab entry, TTY 9-4

I-Node size 6-33
I-Node table 6-10
I-Nodes 6-4, 6-32
Install basic networking software P9-1
Install the LP spooler P7-1
Interface programs, LP
  model 8-8
  writing 8-8
Interface programs
  dumb line printer 8-9
Internal programs, networking
  uugetty 10-4
Interrecord gap recommendations 6-16
Investigating performance problems 7-7
  general procedures, samples 7-7
Invocation, firmware monitor information 4-17

K

Kernel parameters 7-33

L

labelit command 6-16
LAN 10-1
Line characteristics, modify, TTY P8-6
Line printer administration (LP) 8-1
  administrative commands 8-3
  files and directories 8-11
  printer interface programs 8-8
  spooling system works, how 8-1
  user commands 8-2
Line settings, check, TTY P8-1
Line settings, make, TTY P8-4
Line settings, TTY 9-1
  create new 9-3
  modify characteristics 9-4
  what are defined 9-2
link count 6-32
List groups P2-9
List users P2-9
Local advertise table, RFS 11-10
Local area network (LAN) 10-1
Lock files, LP 8-13
Locking unused logins 2-5
Log driver parameters 7-39
Log files, cleaning out, LP 8-14
Logging bad blocks 5-11
Logins 2-2
  adding users 3-1
  administration 3-1
  locking unused 2-5
LP administrative commands 8-2
  command examples 8-4, 8-7, 8-8
  descriptions 8-2
  examples 8-2
  /usr/lib/accept 8-7
  /usr/lib/lpadmin 8-2
  /usr/lib/lpmove 8-6
  /usr/lib/lpsched 8-5
  /usr/lib/lpshut 8-6
  /usr/lib/reject 8-7
LP files and directiories 8-11
  cleaning out log files 8-14
  lock files 8-13
  /usr/spool/lp/class 8-12
  /usr/spool/lp/default 8-11
  /usr/spool/lp/FIFO 8-11
  /usr/spool/lp/interface 8-12
  /usr/spool/lp/log 8-11
  /usr/spool/lp/member 8-12
  /usr/spool/lp/model 8-13
  /usr/spool/lp/oldlog 8-11
  /usr/spool/lp/outputq 8-11
  /usr/spool/lp/pstatus 8-12
  /usr/spool/lp/qstatus 8-12
  /usr/spool/lp/request 8-13
  /usr/spool/lp/seqfile 8-12
LP spooler administration procedures P7-1
  add another printer P7-3
  install the spooler P7-1
  restart the scheduler P7-2
  shutdown for maintenance P7-2
LP spooling system works, how 8-1

M

mail 3-9
mailx 3-9
Maintaining a file system 6-18
Maintenance, RFS 11-25
  domain 11-32
  domain, add new domains 11-34
  domain, add new hosts 11-33
  domain adminstration 11-26
  domain, change name server 11-33
  domain, name server, reassign 11-34
  domain, remove a host 11-33
  domain, update group files 11-34
  domain, update user files 11-34
  host 11-27
  host adminstration 11-25
  host, change password 11-31
  host, passwords 11-29
  host, rfstart options 11-30
  host, shut down 11-31
  host, start up host 11-28
  host, update group information 11-32
  host, update user information 11-32
  host, verification 11-29
  init 3 11-34
  init 3, adding scripts 11-37
  init 3, entering 11-35
  init 3, processing 11-35
  init 3, processing, changing 11-36
  monitoring 11-26, 11-40
  monitoring, CPU time 11-42
  monitoring, remote disk space 11-45
  monitoring, remote system calls 11-40
  monitoring, resource usage 11-44
  monitoring, server processes 11-42
  monitoring, servers, too few 11-42
  monitoring, servers, too many 11-44
  recovery 11-37
  recovery, primary goes down 11-38
  recovery, rfuadmin 11-39
  recovery, rfudaemon 11-39
  recovery, secondary goes down 11-38
  recovery, server goes down 11-39

tuning 11-26
Making a file system available 6-15
Manual networking maintenance P9-13
Map all, RFS 11-23
Map names, RFS 11-24
Mapping components, RFS 11-18
Mapping remote IDs, RFS 11-22
Mapping remote names, RFS 11-23
Mapping remote users, RFS 11-17
components 11-18
group files 11-21
host password 11-21
how mapping works 11-17
idload command 11-20
mapping strategy, choosing 11-21
mapping strategy, map all 11-23
mapping strategy, map name 11-24
mapping strategy, mapping remote IDs 11-22
mapping strategy, mapping remote names 11-23
mapping strategy, no mapping 11-21
rules files 11-18
Mapping strategy, RFS, choosing 11-21
Mapping strategy, RFS, map all 11-23
Mapping strategy, RFS, map name 11-24
Mapping strategy, RFS, mapping remote IDs 11-22
Mapping strategy, RFS, mapping remote names 11-23
Mapping strategy, RFS, no mapping 11-21
MAXPMEM parameter 7-35
MAXSLICE parameter 7-36
MAXUP parameter 7-35
Meaning of yes/no responses 6-37
phase 1 6-37
phase 2 6-40
phase 3 6-41
phase 4 6-43
phase 5 6-45
Memory, tables in 6-10
Memory test 7 load 4-24
firmware monitor information 4-24
mesg command 3-9
Message of the day 3-7
Message parameters 7-40
Message tunable parameters 7-40
mkfs command, using 6-15
MMU register commands 4-20
firmware monitor information 4-20
Mode, TTY 9-1
Model interface programs, LP 8-8
Modify user information, user services P2-4
Modifying the tunable configuration parameters
improving performance 7-3
Monitor error messages
firmware monitor information 4-25
Monitor memory organization 4-18
Monitoring directories that grow 6-21
Monitoring disk usage 6-20, P5-11
Monitoring files that grow 6-21
Monitoring percent of disk space used 6-20
Monitoring, RFS 11-26, 11-40
CPU time 11-42
remote disk space 11-45
remote system calls 11-40
resource usage 11-44
server processes 11-42
servers, too few 11-42
servers, too many 11-44
mount command 6-10, 6-17
Mount remote resources, RFS 11-12, P10-16
Mount security, RFS 11-5
Mounting file systems 6-17
Mounting guidelines, RFS 11-13
Mounting rules, RFS 11-13
MSGMAP parameter 7-40
MSGMAX parameter 7-40
MSGMNB parameter 7-40
MSGMNI parameter 7-40
MSGSEG parameter 7-40
MSGSSZ parameter 7-40
MSGTQL parameter 7-40
Multi-user mode, return to, networking P9-17
Multi-user state, entering 4-11
mv command 6-21

N

NAUTOUP parameter 7-35
NBUF parameter 7-33
NCALL parameter 7-33
NCLIST parameter 7-34
Networking procedures P9-1
add uucp logins P9-10
automated maintenance P9-12
basic debugging P9-14
basic information, check P9-15
basic maintenance P9-11
debug transmissions P9-14
error messages, check P9-15
falut ACU/modem, check P9-14
install basic software P9-1
manual maintenace P9-13
NPACK devconfig entries, create P9-20
NPACK devices entry, create P9-19
NPACK dialers entry P9-21
NPACK NETWORK listener, set up P9-20
NPACK systems entry, create P9-18
other files P9-11
remove BNU software P9-16
return to multi-user mode P9-17
run sysadm installpkg P9-2
run sysadm removepkg P9-17
set BNU NPACK network P9-18
set BNU NPACK network, prerequisites P9-18
set up basic files P9-3
Set up BNU NPACK network P9-21
set up devconfig file P9-9
set up devices file P9-4
set up /etc/inittab P9-6
set up permissions file P9-9
set up poll file P9-8
set up sysfiles file P9-10
set up systems file P9-7
single-user mode P9-17
systems file, check P9-14
uudemon.admin P9-13
uudemon.cleanup P9-13
uudemon.hour P9-12
uudemon.poll P9-12
Networking
administrative files 10-23
admistrative programs 10-3
admistrative programs, uucheck 10-3

admistrative programs, uucleanup 10-3
admistrative programs, uulog 10-3
admistrative programs, Uutry 10-3
basic 10-1
commands 10-2
daemons 10-3
daemons, uucico 10-3
daemons, uusched 10-4
daemons, uuxqt 10-3
database, supporting 10-4
devconfig file 10-22
devices file 10-5
devices file entries, making 10-26
dialcodes file 10-15
dialers file 10-9
direct links 10-1, 10-25, 10-26
/etc/inittab file, changing 10-27
hardware 10-1
internal programs, uugetty 10-4
local area network (LAN) 10-1
other files 10-23
permissions file 10-15
permissions file, considerations 10-16
permissions file, entries are structured 10-15
permissions file, options 10-16
poll file 10-21
protocols 10-9
sysfiles file 10-22
systems file 10-11
systems file entries, making 10-28
telephone lines 10-1
user programs 10-2
user programs, ct 10-2
user programs, cu 10-2
user programs, uucp 10-2
user programs, uupick 10-2
user programs, uustat 10-2
user programs, uuto 10-2
user programs, uux 10-2
news 3-8
NFILE parameter 7-34
nice command 7-10
shift workload to off-peak hours 7-10
NINODE parameter 7-33
NMOUNT parameter 7-34
Node name, changing P1-4, P1-5
NODE parameter 7-36
NOTICE error messages C-1
NPACK devconfig entries, create P9-20
NPACK devices entry, create, networking P9-19
NPACK dialers entry, networking P9-21
NPACK NETWORK listener, set up P9-20
NPACK systems entry, create, networking P9-18
NPBUF parameter 7-35
NPROC parameter 7-34

O

Open file table 6-11
Open
acccessing a file 6-12
Operating levels
entering multi-user state 4-11
general 4-5
init controls the system state 4-6
Optional domain setup P10-6
Optional host setup, RFS P10-11
Organization of file system free list 7-5
improving performance 7-5
Other networking files P9-11

P

Paging parameters 7-36
PANIC error messages C-5
Parameter tuning, RFS 11-46, 11-47
Parameters
kernel 7-33
log driver 7-39
message 7-40
paging 7-36
remote file sharing 7-42
semaphore 7-41
shared memory 7-42
streams 7-37
suggested values for GENIX V.3 systems 7-30
tunable 7-28
tunable, modifying P6-2
Partitioning 5-5
hard disk 5-5
hard disk, change 5-5
hard disk default, single A-4
increase swap space 5-6
Partitions 6-8
diskette A-3
hard disk default A-3
typical device, examples A-2
Password 2-2
Password, RFS, change host 11-31
Password
aging 2-2
aging character codes 2-3
change by root only 2-5
change every two weeks 2-4
change one time only 2-4
changing entries 3-2
deleting entries 3-2
entries 2-4
RFS 11-29
Passwords, security
administrative, special P1-6
Password
special administrative 2-5
Pathname conversion 6-14
Performance management 7-1
finding problems 7-2
fixing problems 7-2
general approach 7-1
improving performance 7-3
outline of typical troubleshooting procedure 7-8
performance tools 7-11
tunable parameters 7-28
Performance tools
performance management 7-11
sadp command 7-26
sag command 7-24
sar command 7-12
timex command 7-25
Permissions file, networking 10-15, P9-9
considerations 10-16

entries are structured 10-15
options 10-16
Policies, need for 6-19
Poll file, networking 10-21
pollmgmt, networking P9-8
port, TTY 9-1
portmgmt, networking P9-6
Powerdown P3-2
multi-user, from P3-2, P3-4
single-user P3-5
Powering up 4-11
Powerup P3-1
Prerequesites, removing BNU software P9-16
Print commands 4-19
firmware monitor information 4-19
Printer interface programs 8-8
model interface programs 8-8
writing interface programs 8-8
Processor operations 4-1
Processor operations procedures P3-1
powerdown P3-2
powerdown, multi-user, from P3-2, P3-4
powerdown, single-user P3-5
powerup P3-1
return to multi-user P3-6
return to multi-user, from single-user P3-6
return to multi-user, from single-user P3-6
system trouble, perform a system dump P3-8
system trouble, recovery from P3-7
Processor operations
administrative directories 4-2
administrative files 4-2
general operating policy 4-1
important system files 4-3
maintaining a system log 4-2
operating levels 4-5
root directories 4-2
Program execution commands
firmware monitor information 4-22
Program interrupts 6-27
Protocols, networking 10-9
ps command 7-6
improving performance 7-6
PUTBUFSZ parameter 7-36

R

read, accessing a file 6-13
Rebuilding the operating system P6-2
Reconfigure the system P6-1
Recovery, RFS 11-37
primary goes down 11-38
rfuadmin 11-39
rfudaemon 11-39
secondary goes down 11-38
server goes down 11-39
Regular data blocks 6-35
REL parameter 7-36
Remote disk space, RFS 11-45
Remote file sharing mode (init 3) 11-34
adding scripts 11-37
entering 11-35
processing 11-35
processing, changing 11-36
Remote file sharing parameters 7-42
Remote file sharing procedures P10-1
advertise resources P10-14
basic domain setup P10-2
basic domain setup, configuration P10-2
basic domain setup, prerequisites P10-2
basic host setup P10-8
basic host setup, configuration P10-9
basic host setup, prerequisites P10-8
domain setup P10-1
host setup P10-8
mount remote resources P10-16
optional domain setup P10-6
optional host setup P10-11
Remote file sharing, (RFS) 11-1
administration 11-4
advertise resources 11-8
aliases resources 11-9
automatic remote mounts 11-13
connect security 11-5
connect security, group 11-7
connect security, mount 11-5
connect security, user 11-7
domain advertise table 11-11
domain mount table 11-15
domains 11-3
forced unmount 11-16
host mount table 11-14
local advertise table 11-10
maintenance 11-25
maintenance, domain 11-32
maintenance, domain, add new domains 11-34
maintenance, domain, add new hosts 11-33
maintenance, domain adminstration 11-26
maintenance, domain, change name server 11-33
maintenance, domain, name server, reassign 11-34
maintenance, domain, remove a host 11-33
maintenance, domain, update group files 11-34
maintenance, domain, update user files 11-34
maintenance, host 11-27
maintenance, host adminstration 11-25
maintenance, host, change password 11-31
maintenance, host, passwords 11-29
maintenance, host, rfstart options 11-30
maintenance, host, shut down 11-31
maintenance, host, start up 11-28
maintenance, host, update group information 11-32
maintenance, host, update user information 11-32
maintenance, host, verification 11-29
maintenance, init 3 11-34
maintenance, init 3, adding scripts 11-37
maintenance, init 3, entering 11-35
maintenance, init 3, processing 11-35
maintenance, init 3, processing, changing 11-36
maintenance, monitoring 11-26, 11-40
maintenance, monitoring, CPU time 11-42
maintenance, monitoring, remote disk space 11-45
maintenance, monitoring, remote system calls 11-40
maintenance, monitoring, resource usage 11-44
maintenance, monitoring, server processes 11-42
maintenance, monitoring, servers, too few 11-42
maintenance, monitoring, servers, too many 11-44
maintenance, recovery 11-37
maintenance, recovery, primary goes down 11-38
maintenance, recovery, rfuadmin 11-39
maintenance, recovery, rfudaemon 11-39
maintenance, recovery, secondary goes down 11-38

maintenance, recovery, server goes down 11-39
maintenance, tuning 11-26
mapping remote users 11-17
mapping remote users, components 11-18
mapping remote users, group files 11-21
mapping remote users, host password 11-21
mapping remote users, how it works 11-17
mapping remote users, idload command 11-20
mapping remote users, rules files 11-18
mapping strategy, choosing 11-21
mapping strategy, map all 11-23
mapping strategy, map name 11-24
mapping strategy, mapping remote IDs 11-22
mapping strategy, mapping remote names 11-23
mapping strategy, no mapping 11-21
mount remote resources 11-12
mounting guidelines 11-13
mounting rules 11-13
parameter tuning 11-46, 11-47
resource security 11-9
resource sharing 11-1
security 11-5
sharing resources 11-7
unadvertise 11-12
unmounting 11-16
Remote file sharing security, example 11-6
Remote system calls, RFS 11-40
Remove a host, RFS 11-33
Remove BNU software, networking P9-16
Removing inactive files 6-21
Reporting bad blocks 5-11
Rescan for more DUPS, phase 1B 6-39
Resource security, RFS 11-9
Resource sharing, RFS 11-1
Resource usage, RFS 11-44
Restart the scheduler, LP P7 2
Restore, file system 6-23
Restore
file system P5-13
Restoring good file system organization
improving performance 7-5
Restricted Shell 3-7
Return to multi-user P3-6
from single-user P3-6
RFS, parameters 11-47
rfstart options, RFS 11-30
rfuadmin, RFS 11-39
rfudaemon, RFS 11-39
Root directories 4-2
Rules files, RFS 11-18
Run level 3 4-15
Run level 5 4-16
Run level 6 4-16
Run level change 4-12
Run level directories 4-13
Run levels B-6
Run levels, changing 4-13
Run sysadm installpkg P9-2
Running fsck 6-35

S

sadp command
cylinder access historgram, output 7-27
performance tools 7-26
seek distance historgram, output 7-29
sag command
output, example 7-24
performance tools 7-24
Salvage free list
phase 6 6-47
sar command, RFS
-Dc 11-40
-Du 11-42
-S 11-42
sar command
—a option 7-12
—A option 7-22
—b option 7-13
—c option 7-14
—d option 7-15
—m option 7-16
—p option 7-20
—q option 7-16
—r option 7-21
—u option 7-17
—v option 7-18
—w option 7-19
—y option 7-22
performance tools 7-12
Search time 6-14
Security, connect, RFS 11-5
Security, group, RFS 11-7
Security, mount, RFS 11-5
Security, RFS 11-5
Security, user, RFS 11-7
Security
admistrative passwords, special P1-6
console terminal configuration P1-1
guidelines 2-1
node name, changing P1-4, P1-5
procedures P1-1
set date P1-2
set time P1-2
system 2-1
system administration menu P1-4
system name, changing P1-4, P1-5
SEMAEM parameter 7-41
Semaphore parameters 7-41
Semaphore tunable parameters 7-41
SEMMAP parameter 7-41
SEMMNI parameter 7-41
SEMMNS parameter 7-41
SEMMNU parameter 7-41
SEMMSL parameter 7-41
SEMOPM parameter 7-41
SEMUME parameter 7-41
SEMVMX parameter 7-41
Server processes, RFS 11-42
Servers, RFS
too few 11-42
too many 11-44
Set date, security P1-2
Set time, security P1-2
Set up basic networking files P9-3
Set up BNU NPACK network P9-21

Set up devices file, devicemgmt P9-4
Set up systems file P9-7
Set-GID 2-8
Set-group identification (set-GID) 2-8
Setting text-bit (stick-bit) 7-4
  improving performance 7-4
Set-UID 2-8
  other file systems, check 2-10
  owned by root, check 2-8
  root file system, check 2-9
Set-user identification (set-UID) 2-8
sh command 3-4
Shared memory tunable parameters 7-42
Shared memory 7-42
  parameters 7-42
Sharing resources, RFS 11-7
  advertise resources 11-8
  aliases resources 11-9
  automatic remote mounts 11-13
  domain advertise table 11-11
  domain mount table 11-15
  example 11-2
  forced unmount 11-16
  host mount table 11-14
  local advertise table 11-10
  mount remote resources 11-12
  mounting guidelines 11-13
  mounting rules 11-13
  resource security 11-9
  unadvertise 11-12
  unmounting 11-16
Shell scripts
  file system administration 6-19
Shell
  default 3-7
  restriced 3-7
SHMALL parameter 7-42
SHMMAX parameter 7-42
SHMMIN parameter 7-42
SHMMNI parameter 7-42
SHMSEG parameter 7-42
Shutdown for maintenance, LP P7-2
Shutdown host, RFS 11-31
Single-user mode, going to 4-15
Single-user mode, networking P9-17
Size limitations 6-9
Space users, identifying large 6-22
Special administrative passwords 2-5
Stand-alone diagnostics 4-25
Start up host, RFS 11-28
Storage blocks 6-7
Storage device 6-7
  file system, relationship 6-7
Stream parameters 7-37
stty, TTY 9-5
Super-block 6-3, 6-5
Swap space
  changing partitions to increase 5-6
Swapping, check for excess
  general procedures, samples 7-9
sync command 6-27
Synchronization 6-14
SYS parameter 7-36
sysadm fileage command 6-22
sysadm removepkg P9-17
sysdump, example P3-8
sysfiles file, networking 10-22, P9-10
System administration menu, security P1-4
System dump, perform P3-8
System file table 6-10
System files, important 4-3
System identification P1-1
System I-Node table 6-11
System intialization 4-8
System life cycle 4-13, 4-14
  changing run levels 4-13
  run level 3 4-15
  run level 5 4-16
  run level 6 4-16
  run level directories 4-13
  single-user mode 4-15
  turning the system off 4-16
System log, maintaining a 4-2
System name, changing P1-4, P1-5
System reconfiguration procedures P6-1
  modifying the tunable parameters P6-2
  rebuilding the operating system P6-2
  reconfigure the system P6-1
  unbootable operating system recovery P6-4
System reconfiguration
  general procedures, samples 7-10
System security 2-1
System states 4-7
System trouble, recovery from P3-7
  perform a system dump P3-8
  sysdump, example P3-8
System
  init controls 4-6
systemmgmt, networking P9-7
Systems file entries, making, networking 10-28
Systems file, networking 10-11

T

Table overflows, check for potential 7-9
  general procedures, samples 7-9
Tables in memory 6-10
tail command 6-21
Telephone lines, networking 10-1
Terminal options, how to set, TTY 9-5
Terminal options, TTY 9-1
Test sequence 4-22
  firmware monitor information 4-22
timex command, performance tools 7-25
Trouble reporting 3-9, 3-10
Troubleshooting procedure, outline of typical 7-8
TTY line 9-1
TTY management 9-1
  baud rate 9-1
  getty entries 9-5
  gettydefs entries 9-3
  hunt sequence 9-1
  inittab entry 9-4
  line settings 9-1
  line settings, check P8-1
  make line settings P8-4
  mode 9-1
  modify line characteristics P8-6
  port 9-1
  procedures P8-1
  stty 9-5

sysadm modtty 9-4
terminal options 9-1
terms, definition of 9-1
TTY line 9-1
TTY system 9-2
how it works 9-2
line settings, create new 9-3
line settings, modify characteristics 9-4
line settings, what are defined 9-2
terminal options, how to set 9-5
Tunable parameters
kernel 7-33
log driver 7-39
message 7-40
modifying P6-2
paging 7-36
performance management 7-28
remote file sharing 7-42
semaphore 7-41
semaphores 7-41
shared memory 7-42
streams 7-37
Tuning, RFS 11-26
Turning the system off 4-16
Type, fsck 6-32

U

umask 3-6
Unadvertise, RFS 11-12
uname command, security P1-5
Unbootable operation system recovery P6-4
Unmounting file systems 6-17
Unmounting, RFS 11-16
Update group files, RFS 11-34
Update group information, RFS 11-32
Update user files, RFS 11-34
Update user information, RFS 11-32
User commands for the LP spooling system 8-2
User communications services 3-7
User $PATH variables 7-7
improving performance 7-7
User programs, networking 10-2
ct 10-2
cu 10-2
uucp 10-2
uupick 10-2
uustat 10-2
uuto 10-2
uux 10-2
User requests, anticipating 3-9
User security, RFS 11-7
User services 3-1
add groups P2-1
add users P2-1
anticipating user requests 3-9
delete groups P2-7
delete users P2-7
list groups P2-9
list users P2-9
login administration 3-1
modify user information P2-4
procedures P2-1
user communications services 3-7
user's environment 3-4
write to all users P2-11
User's environment 3-4
Utilities, fsck 6-28
uucheck command, networking 10-3
uucico command, networking 10-3
uucleanup command, networking 10-3
uucp command, networking 10-2
uucp logins, add, networking P9-10
uudemon.admin, networking P9-13
uudemon.cleanup, networking P9-13
uudemon.hour, networking P9-12
uudemon.poll, networking P9-12
uugetty command, networking 10-4
uulog command, networking 10-3
uupick command, networking 10-2
uusched command, networking 10-4
uustat command, networking 10-2
uuto command, networking 10-2
Uutry command, networking 10-3
uux command, networking 10-2
uuxqt command, networking 10-3

V

Variables
environment 3-4
user $PATH, improving performance 7-7
VER parameter 7-36
Verification, RFS 11-29
Verifying usability 5-7

W

wall command 3-9
WARNING error messages C-4
Workload, shift to off-peak hours
batch command 7-10
general procedures, samples 7-10
nice command 7-10
write, accessing a file 6-13
Write to all users 3-9, P2-11
Writing interface programs, LP 8-8

National
Semiconductor

MICROCOMPUTER
SYSTEMS DIVISION

READER'S COMMENT FORM

In the interest of improving our documentation, National Semiconductor invites your comments on this manual.

Please restrict your comments to the documentation. Technical Support may be contacted at:

(800) 538-1866 - U.S. non CA
(800) 672-1811 - CA only
(800) 223-3248 - Canada only

Please rate this document according to the following categories. Include your comments below.

	EXCELLENT	GOOD	ADEQUATE	FAIR	POOR
Readability (style)	☐	☐	☐	☐	☐
Technical Accuracy	☐	☐	☐	☐	☐
Fulfills Needs	☐	☐	☐	☐	☐
Organization	☐	☐	☐	☐	☐
Presentation (format)	☐	☐	☐	☐	☐
Depth of Coverage	☐	☐	☐	☐	☐
Overall Quality	☐	☐	☐	☐	☐

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Do you require a response? ☐ Yes ☐ No PHONE _____

Comments:

_____

_____

_____

_____

_____

_____

FOLD, STAPLE, AND MAIL

424510771-110A

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 409 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

National Semiconductor Corporation
Microcomputer Systems Division
Technical Publications Dept. 8278, M/S 7C261
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-9968