



# UNIX<sup>®</sup> System V/386

SYSTEM ADMINISTRATOR'S GUIDE





**UNIX<sup>®</sup> System V** Release 3.0

**INTEL** 80286/80386

Computer Version

System Administrator's Guide



**PRENTICE HALL, ENGLEWOOD CLIFFS, NEW JERSEY 07632**

© 1988 by AT&T. All Rights Reserved.

### IMPORTANT NOTICE TO USERS

While every effort has been made to ensure the accuracy of all information in this document, AT&T assumes no liability to any party for any loss or damage caused by errors or omissions or statements of any kind in the UNIX® System V/386 System Administrator's Guide © AT&T, its upgrades, supplements, or special editions, whether such errors are omissions or statements resulting from negligence, accident or any other cause. AT&T further assumes no liability arising out of the application or use of any product or system described herein; nor any liability for incidental or consequential damages arising from the use of this document. AT&T disclaims all warranties regarding the information contained herein, whether expressed, implied or statutory, including implied warranties or merchantability or fitness for a particular purpose.

AT&T reserves the right to make changes without further notice to any products herein to improve reliability, function or design.

No part of this publication may be reproduced, transmitted or used in any form or by any means—graphic, electronic, mechanical or chemical, including photocopying, recording in any medium, taping, by any computer or information storage and retrieval systems, etc. without prior permission in writing from AT&T.

UNIX is a registered trademark of AT&T

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-940891-6

Prentice-Hall International (UK) Limited, *London*  
Prentice-Hall of Australia Pty. Limited, *Sydney*  
Prentice-Hall Canada Inc., *Toronto*  
Prentice-Hall Hispanoamericana, S.A., *Mexico*  
Prentice-Hall of India Private Limited, *New Delhi*  
Prentice-Hall of Japan, Inc., *Tokyo*  
Simon & Schuster Asia Pte. Ltd., *Singapore*  
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Copyright© 1987 AT&T  
All Rights Reserved  
Printed in U.S.A.

## NOTICE

The information in this document is subject to change without notice.  
AT&T assumes no responsibility for any errors that may appear in this document.

386/ix is a trademark of Interactive Systems Corporation.  
ACT is a trademark of Micro-Term.  
AnnArbor is a trademark of AnnArbor Terminals.  
Beehive is a trademark of Beehive International.  
Concept is a trademark of Human Designed Systems.  
CrystalWriter is a trademark of Syntactics.  
DATASPEED is a registered trademark of AT&T.  
dBASE II is a registered trademark of Ashton-Tate.  
DEC, PDP, VAX, and VT100 are trademarks of Digital Equipment Corporation.  
DOCUMENTER'S WORKBENCH is a trademark of AT&T.  
Dataphone is a registered trademark of AT&T.  
Develcon is a trademark of Develcon Electronics, Incorporated.  
Diablo is a registered trademark of Xerox.  
Dow Jones News/Retrieval Service is a trademark of Dow Jones.  
Ethernet is a registered trademark of Xerox.  
HP is a registered trademark of Hewlett-Packard, Inc.  
IBM is a trademark of International Business Machines.  
IMAGEN is a trademark of IMAGEN Corporation.  
INFORMIX is a registered trademark of Relational Database Systems.  
INGRES/CS is a trademark of Relational Technology.  
INSTRUCTIONAL WORKBENCH is a trademark of AT&T.  
Intel is a registered trademark of Intel Corporation.  
LSI is a trademark of Lear Siegler.  
MBASIC is a registered trademark of Microsoft.  
MICOM is a registered trademark of MICOM System, Incorporated.  
MS-DOS is a registered trademark of Microsoft Corporation.  
MULTIBUS is a registered trademark of Intel Corporation.  
Micro-Term and MIME are trademarks of Micro-Term.  
Microsoft is a registered trademark of Microsoft.  
Multiplan is a registered trademark of Microsoft.  
Official Airline Guide is a trademark of Official Airline Guide, Inc.  
PC-Interface is a registered trademark of Locus Computing.  
Penril is a trademark of Penril Corporation.  
RM/COBOL is a trademark of Ryan-McFarland.  
SuperCalc3 is a trademark of Sorcim/IUS Micro Software.  
Syntactics is a trademark of Syntactics.  
TEKTRONIX and TEKTRONIX 4010 are registered trademarks of Tektronix, Inc.  
TELETYPE is a registered trademark of AT&T.  
TeleVideo is a registered trademark of TeleVideo Systems.  
Teleray is a trademark of Research Inc.  
TermiNet is a trademark of General Electric.  
UNIX is a registered trademark of AT&T.  
UltraCalc is a trademark of OLYMPUS Software.  
Unify is a registered trademark of Unify.  
Ventel is a trademark of Ven-Tel, Incorporated.  
Versatec is a registered trademark of Versatec Corporation.  
WE is a registered trademark of AT&T.  
WRITER'S WORKBENCH is a trademark of AT&T.  
Weitek is a trademark of Weitek Corporation.  
XED is a trademark of Computer Concepts.  
Xenix is a registered trademark of Microsoft Corporation.



---

## AT&T Products and Services

- To order documents from the Customer Information Center:
  - within the continental United States, call 1-800-432-6600
  - outside the continental United States, call 1-317-352-8557
  - send mail orders to:
    - AT&T Customer Information Center
    - Customer Service Representative
    - P.O. Box 19901
    - Indianapolis, Indiana 46219
- To sign up for UNIX system or AT&T computer courses:
  - within the continental United States, call 1-800-221-1647
  - outside the continental United States, call 1-609-639-4593
  - TELEX: 1-609-639-4756  
Attention: Training Registration
- For information on Intel hardware and software, contact the Intel sales office nearest you.
- To find out about UNIX system source licenses:
  - within the continental United States, except North Carolina, call 1-800-828-UNIX
  - in North Carolina and outside the continental United States, call 1-919-855-2737
  - or write to:
    - Software Licensing
    - Guilford Center
    - Salem Bldg. 4th Floor
    - P.O. Box 25000
    - Greensboro, NC 27420

---

## Table of Contents

### Part 1: Procedures

System Identification and Security Procedures	P1-1
Procedure 1.1: Check Console Terminal Configuration	P1-2
Procedure 1.2: Set Time and Date	P1-4
Procedure 1.3: Establish or Change System and Node Names	P1-6
Procedure 1.4: Assign Special Administrative Passwords	P1-9
Procedure 1.5: Forgotten Root Password Recovery	P1-12
User Services Procedures	P2-1
Procedure 2.1: Add Users or Groups	P2-2
Procedure 2.2: Modify User Information	P2-5
Procedure 2.3: Delete Users or Groups	P2-11
Procedure 2.4: List Users or Groups	P2-14
Procedure 2.5: Write to All Users	P2-17
Processor Operations Procedures	P3-1
Procedure 3.1: Powerup	P3-2
Procedure 3.2: Powerdown	P3-4

## Table of Contents

---

Procedure 3.3: Shutdown to Single User	P3-8
Procedure 3.4: Return to Multiuser	P3-9
Procedure 3.5: Recovery from System Trouble	P3-12
Procedure 3.6: Reload the Operating System	P3-13
Disk/Tape Management Procedures	P4-1
Procedure 4.1: Format Diskettes	P4-2
Procedure 4.2: Duplicating Diskettes	P4-5
File System Administration Procedures	P5-1
Procedure 5.1: Format and Create File Systems on Diskette	P5-2
Procedure 5.2: Create File Systems on Second Hard Disk	P5-6
Procedure 5.3: Maintain File Systems	P5-11
Procedure 5.4: File System Backup and Restore	P5-17
Performance Management Procedures	
System Reconfiguration Procedures	P6-1
Procedure 6.1: Unbootable Operating System Recovery	P6-2

LP Spooler Administration Procedures	P7-1
Procedure 7.1: Install the LP Spooler	P7-2
Procedure 7.2: Shut Down for Maintenance	P7-3
Procedure 7.3: Restart the Scheduler	P7-4
Procedure 7.4: Adding Another LP Printer	P7-6
TTY Management Procedures	P8-1
Procedure 8.1: Check TTY Line Settings	P8-2
Procedure 8.2: Make TTY Line Settings	P8-6
Procedure 8.3: Modify TTY Line Characteristics	P8-8
Basic Networking Procedures	P9-1
Procedure 9.1: Install Basic Networking Software	P9-2
Procedure 9.2: Set Up Basic Networking Files	P9-5
Procedure 9.3: Basic Networking Maintenance	P9-18
Procedure 9.4: Basic Networking Debugging	P9-22
Procedure 9.5: Remove BNU Software	P9-25
Procedure 9.6: Set Up BNU STARLAN NETWORK (Basic)	P9-28
Procedure 9.7: Set Up BNU STARLAN NETWORK (Special)	P9-34

## Table of Contents

---

Remote File Sharing Procedures	P10-1
Procedure 10.1: Domain Setup (Primary)	P10-2
Procedure 10.2: Host Setup	P10-12
Procedure 10.3: Advertise Resources	P10-20
Procedure 10.4: Mount Remote Resources	P10-23

## Part 2: Support

### 1. System Identification and Security

Introduction	1-1
Important Security Guidelines	1-2
Logins and Passwords	1-3
Set-UID and Set-GID	1-10

### 2. User Services

Introduction	2-1
Login Administration	2-2
The Users' Environment	2-6
User Communications Services	2-11
Anticipating User Requests	2-15



3. Processor Operations	
Introduction	3-1
Operating Levels	3-8
4. Disk/Tape Management	
Introduction	4-1
Device Types	4-2
Identifying Devices to the Operating System	4-4
Formatting and Partitioning	4-8
Other Disk Operations	4-11
The Bad Block Handling Feature	4-13
5. File System Administration	
Introduction	5-1
How the File System is Organized	5-1
The Relationship Between the File System and the Storage Device	5-9
How the File System Works	5-12
Administering the File System	5-21
Maintaining a File System	5-26
What Can Go Wrong With a File System	5-37
How to Check a File System for Consistency	5-39

## 6. Performance Management

Introduction	6-1
General Approach to Performance Management	6-2
Improving Performance	6-4
Samples of General Procedures	6-10
Performance Tools	6-14
Tunable Parameters	6-41

## 7. LP Spooler Administration

Introduction	7-1
Administrative Commands	7-3
Printer Interface Programs	7-12
Files and Directories	7-16

## 8. TTY Management

Introduction	8-1
The TTY System	8-3

## 9. Basic Networking

Introduction	9-1
Networking Hardware	9-2
Networking Commands	9-3

Daemons	9-5
Supporting Data Base	9-7
Administrative Files	9-33
Direct Links	9-36
10. Remote File Sharing	
Overview	10-1
Sharing Resources	10-10
Mapping Remote Users	10-23
Maintenance	10-33
Appendix A: Diskette Naming Conventions	A-1
Appendix B: Directories and Files	B-1
Appendix C: UNIX System Error Messages	C-1
Glossary	G-1
Index	I-1



---

## List of Figures

Figure 1-1: Password Aging Character Codes	1-4
Figure 2-1: A Default <code>/etc/profile</code>	2-7
Figure 2-2: Environment Array for a Typical User	2-8
Figure 3-1: System States	3-10
Figure 3-2: A Look at System Initialization	3-14
Figure 3-3: A Look at the System Life Cycle	3-18
Figure 4-1: Directory Listing Extracts: Regular and Device Files	4-5
Figure 5-1: A UNIX System File System	5-1
Figure 5-2: Adding the <code>/usr</code> File System	5-2
Figure 5-3: The UNIX System View of a File System	5-3
Figure 5-4: The Super Block	5-5
Figure 5-5: The File System Address Chain	5-7
Figure 5-6: Disk Partitions, 40-Megabyte Drive	5-11
Figure 5-7: The System I-Node Table	5-13
Figure 5-8: File System Tables and Their Pointers	5-15
Figure 5-9: Interrecord Gap and Blocks/Cylinder Recommendations	5-22
Figure 6-1: Outline of Typical Troubleshooting Procedure	6-12
Figure 6-2: Example of <code>sag</code> Output	6-34
Figure 6-3: Output from <code>sadp</code> : Cylinder Access Histogram	6-38
Figure 6-4: Output from <code>sadp</code> : Seek Distance Histogram	6-39
Figure 6-5: Suggested Parameter Values: Release 3.0 Systems	6-43
Figure 7-1: User Commands for the LP Spooling System	7-2
Figure 7-2: Administrative Commands for the LP Spooling System	7-3
Figure 7-3: Dumb Line Printer Interface Program	7-15



## List of Figures

---

Figure 8-1: <b>gettydefs</b> Entries	8-4
Figure 8-2: <b>getty</b> Entries from <b>/etc/inittab</b>	8-7
Figure 10-1: Example - Sharing Resources	10-3
Figure 10-2: Example - Remote File Sharing Security	10-7
Figure 10-3: Output from <b>sar -Dc</b>	10-54
Figure 10-4: Output from <b>sar -Du</b>	10-56
Figure 10-5: Output from <b>sar -S</b>	10-57
Figure 10-6: Output from <b>fusage</b>	10-59
Figure 10-7: Output from <b>df</b>	10-60

---

## Introduction

This guide describes procedures used in the administration of your computer running the UNIX System V Release 3.0 operating system. It is designed to accomplish the following objectives:

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

## 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 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 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.

## 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 UNIX system fundamentals as the directory structure and the shell. We also expect you to feel comfortable using your 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 UNIX system manuals you may want to look at before starting on this one.

## How This Guide is Organized

There are two main parts to this guide:

- |                              |   |
|------------------------------|---|
| Part 1 - Procedures          | Part 1 contains ten sets of step-by-step procedures that tell you how to keep your computer in operation. Each set of tasks is related to a general topic, such as User Services or Processor Operations. |
| Part 2 - Support Information | Part 2 contains ten chapters of more detailed information about each of the ten sets of procedures. The chapters are numbered to parallel the procedures of Part 1.                                       |

In the back of the book there are three appendices, a glossary, and an index.

## How to Use This Guide

1. **Use the procedures in Part 1 to begin with.**

They lead you through administrative tasks without requiring preliminary knowledge or experience in that area.

2. **Use the chapters in Part 2 to learn more about what the procedures do.**

They explain what is going on in the procedures and provide background information about the basic elements of the UNIX system.

3. **Use the Index for navigation.**

It provides a cross-reference so you can uncover all the places in the guide where a topic is discussed.

4. **Use the Glossary to look up definitions of terms that are unfamiliar.**

5. **As you gain experience, use the guide for reference.**

## **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:

<b>Purpose</b>	Summary of what the procedure is used for.
<b>When Performed</b>	When you should schedule the procedure.
<b>Starting Conditions</b>	The state the computer should be in when you begin the procedure. Any special login requirements.
<b>sysadm menu</b>	The part of the System Administration Menu package that contains the subcommands to perform the procedure.
<b>Commands</b>	The commands used to perform the procedure.
<b>Bootable Programs</b>	The names of programs used to boot the system.
<b>Media</b>	Diskettes or tapes used in the procedure.
<b>Time</b>	Approximately how long the procedure takes.
<b>Caution</b>	Special instructions you must follow before or during the procedure to ensure the integrity of your system software and user files.
<b>Reference</b>	The chapter and section in Part 2 where this topic is more fully discussed. Other UNIX system manuals where additional information can be found.



## 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. It is described in the *Owner/Operator Manual*.

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
```

## 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 the section on "Operating Levels," in Chapter 3, "Processor Operations", for more information on system states.

## 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 1, "System Security", for a list of these logins).

## 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 UNIX 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
```

## Notation 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 UNIX 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 **constant width** 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.

## 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)        *System Administrator's Reference Manual*

Sections (1), (2), (3), (4), (5)    *Programmer's Reference Manual*

## 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 UNIX system software may cause small differences in what appears on your terminal.



---

## System Identification and Security Procedures

Procedure 1.1: Check Console Terminal Configuration	P1-2
Procedure 1.2: Set Time and Date	P1-4
Procedure 1.3: Establish or Change System and Node Names	P1-6
System Administration Menu— <b>nodename</b>	P1-7
<b>Command—uname</b>	P1-7
Procedure 1.4: Assign Special Administrative Passwords	P1-9
Procedure 1.5: Forgotten Root Password Recovery	P1-12





---

# System Identification and Security Procedures

The following procedures are covered in this section:

- Procedure 1.1    **Check Console Terminal Configuration**  
To assure that the system console terminal is properly configured.
- Procedure 1.2    **Set Time and Date**  
To set the time and date of the internal system clock.
- Procedure 1.3    **Establish or Change System Node Names**  
To define the formal system name, especially for the computer to be a node in a network.
- Procedure 1.4    **Assign Special Administrative Passwords**  
To assign special passwords to administrative and system logins.
- Procedure 1.5    **Forgotten Root Password Recovery**  
To recover from forgetting or the corruption of the root password.

---

## Procedure 1.1: Check Console Terminal Configuration

<b>Purpose</b>	To assure that communication with the system is maintained.
<b>References</b>	<i>Owner/Operator Manual</i> The Operator's Guide 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 shut down to the firmware mode (see Chapter 3, "Processor Operations", for an explanation of firmware mode and other operating modes).

### For MB1 Systems

Here are some things you might want to do to make sure your console terminal is configured properly. Use the Operator's Guide 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 in the firmware mode, make sure the input/output terminal speed option is set to 9600 and hit **RETURN** (see also the Troubleshooting Information in the *Owner/Operator Manual*).

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. The console printer should be independent of the LP Spooler system.

## For AT286/AT386 Systems

The console is the keyboard and the display part of your computer. Connect the keyboard and the display according to your computer installation manual. The system will work with any display: either Monochrome Adapter, CGA (Color Graphics Adapter), or EGA (Extended Graphics Adapter) type.

A **setup** program (provided with your computer) needs to be executed before UNIX System V is installed. Select the proper adapter type for your console when prompted by the **setup** program.

---

## Procedure 1.2: Set Time and Date

<b>Purpose</b>	To synchronize system time with clock time or to reset the system time after it has been corrupted.
<b>Starting Conditions</b>	System state —multiuser, for synchronizing system time with current clock time —single-user, for setting the date ahead Login— <b>root</b> , to reset clock with <b>date(1)</b>
<b>sysadm menu</b>	SYSTEM SETUP
<b>Commands</b>	<b>sysadm datetime(1)</b> <b>date(1)</b> —requires logging in as <b>root</b>
<b>Caution</b>	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 multiuser 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 UNIX operating system is booted, you will be asked to set the time-of-day clock. You may also have to set the clock when you

- first get the system
- notice that the date/time is incorrect.

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.
```

- 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
```

---

## Procedure 1.3: Establish or Change System and Node Names

<b>Purpose</b>	To define the system and node names —for a new computer —when reconfiguring the system to include basic networking
<b>Starting Conditions</b>	System state—multiuser or single-user Login—authorized login or <b>root</b>
<b>sysadm menu</b>	SYSTEM SETUP
<b>Commands</b>	<b>sysadm nodename(1)</b> <b>uname(1) -S(1)</b> —need <b>root</b> login
<b>Caution</b>	Do not change system and node names arbitrarily. Changes must be coordinated with your networking connections.
<b>References</b>	Chapter 9, “Basic Networking”

The system and node names of the 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 6, “Performance Management”, for more details).

Choose one of the following methods for establishing or changing the system and node name.

## 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 80386
# cat /etc/rc.d/nodename
#     Node name changed 02/01/86 14:55:49.
#
```

## Command—uname

The following shows how to display and change the system and node names.



### Procedure 1.3

---

```
# uname -a
unix unix 3.0 80386
# uname -S abcdefghijk
uname: name must be <= 8 letters
# uname -S ABcd5678
# uname -a
ABcd5678 ABcd5678 3.0 2 80386
#
```

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.

---

## Procedure 1.4: Assign Special Administrative Passwords

<b>Purpose</b>	To permit controlled access to various administrative and special system functions.
<b>Starting Conditions</b>	Multiuser or single-user state, any login.
<b>sysadm menu</b>	SYSTEM SETUP
<b>Commands</b>	<b>sysadm admpasswd(1)</b> <b>sysadm syspasswd(1)</b>
<b>Reference</b>	"Special Administrative Passwords" in Chapter 1, "System Security"

After you have set up your computer you should assign passwords to the special administrative and system logins (see Chapter 1, "System Security", for definitions of these logins). The administrative logins are: **setup**, **power-down**, **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.

## Procedure 1.4

---

```
# 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
```

```
Password unchanged.
```

```
The login 'powerdown' already has a password.
```

```
Do you want to change the password, delete it, or skip it?
```

```
[c, d, s, q, ?] q
```

```
Password unchanged.
```

```
#
```

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 'daemon' 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 1.5: Forgotten Root Password Recovery



Restoring the original operating system is a rather drastic way of recovering from a forgotten root password. Make every effort to remember or discover the root password before performing this procedure.

<b>Purpose</b>	To recover from a forgotten <b>root</b> password by restoring the original operating system (before the password was assigned).
<b>Starting Conditions</b>	Power off, not logged in.
<b>Commands</b>	<code>cpio(1)</code> , <code>mount(1M)</code> , <code>mt(1M)</code> , <code>umount(1M)</code>
<b>Bootable Program</b>	<code>/unix</code> —boot program (from diskettes)
<b>Media</b>	The core system diskettes or for MB1 systems installation tape.

The following steps are necessary to recover the ability to log in as **root** based on the assumption that you are not able to log in (as a conventional user) and restore the `/etc/passwd` file from another login.

### For the MULTIBUS I Systems

You should power up the machine and all associated peripherals and follow one of the two console sequences depending on the level of boot EPROMs in the system:

First console sequence:

```
DEBUGMonitor - 386 d011
[....]
Copyright 1986 Intel Corporation.
>
```

Second console sequence:

After the console displays "xxxx", enter <Shift>U.  
After system diagnostics and the following display:

break to DMON-386 Monitor (y or [n]) ?

Answer 'y' to this in 30 seconds.  
(otherwise it times out to iSDM Monitor,  
then you need to restart from the beginning)

```
DEBUG Monitor - 386 d016 (or d015)
[....]
Copyright 1986 Intel Corporation
>
```

Insert the release tape and type:

```
b ":wta0:#fs=1"
```

Note that the double-quote marks and the leading and trailing colons are required. br

Step 1: On the Multibus 1 (MB1) system, place the distribution tape in the tape drive, press 'reset'. At the '>' prompt, type:

```
b ":wta0:#fs=1"
```

This will put the machine in 'maintenance' mode with a root file system on the RAMDISK.

Step 2: cd / and type:

```
# mount /dev/dsk/0s0 /mnt
cd /mnt
mt -f /dev/rnmt0 fsf 4
cpio -icBvdu < /dev/rmt0 etc/passwd
```

## Procedure 1.5

---

```
umount /dev/dsk/0s0
```

This will replace the `passwd` file with the original distribution `/etc/passwd` file.

Press the reset button and reboot from the hard disk as outlined in procedure 3.1.

- Step 3: Execute `passwd` to change the `root` password.
- Step 4: Store the installation media in a safe place.

## For the AT386 and 286 Systems

- Step 1: Insert the boot diskette

At the 'boot:', type <CR>

- Step 2: After the UNIX system is booted-up, mount the hard disk root file system.

```
# mount /dev/dsk/0s1 /mnt
```

- Step 3: Copy file `/etc/passwd` into the hard disk root file system.

```
# cp /etc/passwd /mnt/etc/passwd
```

- Step 4: Unmount the hard disk file system

```
# umount /mnt
```

- Step 5: Shut down and reboot the system.

---

## **User Services Procedures**

User Services Procedures	P2-1
Procedure 2.1: Add Users or Groups	P2-2
Procedure 2.2: Modify User Information	P2-5
Procedure 2.3: Delete Users or Groups	P2-11
Procedure 2.4: List Users or Groups	P2-14
Procedure 2.5: Write to All Users	P2-17





---

## User Services Procedures

The following procedures are covered in this section:

- Procedure 2.1    **Add Users or Groups**  
To add information about new users of the system or to name groups.
- Procedure 2.2    **Modify User Information**  
To change information about users or groups.
- Procedure 2.3    **Delete Users or Groups**  
To remove users or groups from the system.
- Procedure 2.4    **List Users or Groups**  
To display information about users or groups.
- Procedure 2.5    **Write to All Users**  
To send a message to all users logged in.

---

## Procedure 2.1: Add Users or Groups

<b>Purpose</b>	To identify new users or groups of users to the system.
<b>Starting Conditions</b>	System state—multiuser
<b>sysadm menu</b>	USER MANAGEMENT
<b>Commands</b>	<b>sysadm adduser(1)</b> <b>sysadm addgroup(1)</b>
<b>Reference</b>	"Login Administration" in Chapter 2, "User Services"

Step 1: Enter one of the following commands:

```
$ sysadm adduser
```

```
Password:
```

or

```
$ sysadm addgroup
```

```
Password:
```

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 46145) [?, 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:      46145

  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:

## Procedure 2.1

---

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 45201) [?, q]: <CR>

*(Accepting default by entering <CR>)*

This is the information for the new group:

Group name:       seventy7

group ID:         45201

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**

---

## Procedure 2.2: Modify User Information

<b>Purpose</b>	To change stored information about users.
<b>Starting Conditions</b>	System state—multiuser
<b>sysadm menu</b>	USER MANAGEMENT
<b>Commands</b>	<b>sysadm modadduser(1)</b> <b>sysadm modgroup</b> <b>sysadm moduser(1)</b> <b>sysadm chgloginid(1)</b> <b>sysadm chgpasswd(1)</b> <b>sysadm chgshell(1)</b>
<b>Reference</b>	"Login Administration" in Chapter 2, "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**)

## Procedure 2.2

---

Function 1, Step 1: Enter the command:

```
$ sysadm modadduser  
Password:
```

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.

\$



## Procedure 2.2

---

Function 2, Step 1: To change the value of a group ID name, enter the command:

```
$ sysadm modgroup  
Password:
```

Function 3, Step 1: To change the values for an individual user's login, enter the command:

```
$ sysadm moduser  
Password:
```

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:

## Procedure 2.2

---

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**.

---

## Procedure 2.3: Delete Users or Groups

<b>Purpose</b>	To clear the system of an inactive user. To eliminate a group name that is no longer needed.
<b>Starting Conditions</b>	System state—multiuser
<b>sysadm menu</b>	USER MANAGEMENT
<b>Commands</b>	<b>sysadm deluser(1)</b> <b>sysadm delgroup(1)</b>
<b>Caution</b>	When you delete a user's ID, all the files and directories owned by that ID are deleted too.
<b>Reference</b>	"Login Administration" in Chapter 2, "User Services"

Step 1: Deleting a group ID is done with this command:

```
$ sysadm delgroup  
Password:
```

Step 2: The prompt sequence is as follows:

## Procedure 2.3

---

```
Which group name do you wish to delete?[q] seventy7
Do you want to delete group name 'seventy7', group ID 45201?[y, n, ?, q] y
seventy7 has been deleted
Do you want to delete any other group?[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
```

```
Password:
```

```
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
```

---

## Procedure 2.4: List Users or Groups

<b>Purpose</b>	To see what users or groups are known to the system.
<b>Starting Conditions</b>	System state—multiuser
<b>sysadm menu</b>	USER MANAGEMENT
<b>Commands</b>	<b>sysadm lsuser(1)</b> <b>sysadm lsgroup(1)</b>

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  
Password:
```

which produces a report with these column headings:

Groups currently in the computer  
 (press <RETURN> to start listing each time you hear the bell)

group name	group number	logins permitted to become 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:

Users currently in the computer  
 (press <RETURN> to start listing each time you hear the bell)



## Procedure 2.4

---

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)
sysadmin     general system administration
trouble       trouble(0000)
umountfsys   unmount diskette file system
uucp          0000-uucp(0000)
uucpa        Uucp login
$
```

---

## Procedure 2.5: Write to All Users

<b>Purpose</b>	To send urgent messages to all users logged in.
<b>Starting Conditions</b>	System state—multiuser Login— <b>root</b> required to prevent users from blocking messages.
<b>Commands</b>	<b>wall(1M)</b>
<b>Reference</b>	"Write to All Users" in Chapter 2, "User Services"

Step 1: For times when it is necessary to communicate with all users on the system at once, the UNIX 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:

**Procedure 2.5**

---

Broadcast Message from root: System coming down in  
ten minutes. Please log off.

---

## Processor Operations Procedures

Processor Operations Procedures	P3-1
Procedure 3.1: Powerup	P3-2
Procedure 3.2: Powerdown	P3-4
From Multiuser	P3-4
From Single User	P3-7
Procedure 3.3: Shutdown to Single User	P3-8
Procedure 3.4: Return to Multiuser	P3-9
From Single User	P3-9
From Firmware	P3-10
Procedure 3.5: Recovery from System Trouble	P3-12
Determining the System Trouble	P3-12
Procedure 3.6: Reload the Operating System	P3-13
Instructions for Updating Selected Files from the Release	P3-15
MB1 Release	P3-15
AT386 Release	P3-16
Full System Restore	P3-16



---

## Processor Operations Procedures

The following procedures are covered in this section:

- Procedure 3.1    **Powerup**  
To power up the system to the multiuser state.
- Procedure 3.2    **Powerdown**  
To halt the system and turn the power off.
- Procedure 3.3    **Shutdown to Single-User**  
To bring the system to the single-user state to do administrative tasks.
- Procedure 3.4    **Return to Multiuser**  
To return the system to the multiuser state after it was brought to another state for administrative purposes.
- Procedure 3.5    **Recovery from System Trouble**  
To handle system troubles due to hardware problems and software errors.
- Procedure 3.6    **Reload the Operating System**  
To reload the system from diskettes, if the system itself has been severely damaged, or to change the disk partitions. Procedures include partial restore, full restore, and full restore with hard disk repartitioning.

---

## Procedure 3.1: Powerup

<b>Purpose</b>	To turn on the system and make it available for use.
<b>Starting Conditions</b>	System state—(power off).
<b>Time</b>	Approximately 20 seconds (depending on configuration).
<b>Reference</b>	"Powering Up" in Chapter 3, "Processor Operations"

To power up your computer from a halted state (power is off), perform the following procedure:

- Step 1: Turn on the console and wait for the cursor to appear.
- Step 2: Turn the latch on the diskette drive to the horizontal position to make sure no diskette is in the drive.
- Step 3: Turn power switch on.

Here is a typical example of what appears at the system console:

On an **MB1** system:

```
xxxx xxxxx x
xxxx xxxxx x
Copyright 1986 Intel Corporation.
>
```

On an **AT386** system:

```
xxxx:xxxx: x
xxxx:xxxx: x
512 Base Memory, 02048 Extended
boot:
```

Step 4: Boot the system. On the **MB1** system enter '**b**' and a carriage return:

**>b <CR>**

On the **AT386** enter a carriage return:

**boot: <CR>**



---

## Procedure 3.2: Powerdown

<b>Purpose</b>	To halt the system and turn off the power.
<b>Starting Conditions</b>	System state—multiuser or single-user. You must <b>mount /usr</b> to run this procedure in single-user mode. Login—authorized user or root
<b>sysadm menu</b>	MACHINE MANAGEMENT
<b>Commands</b>	<b>sysadm whoson(1)</b> <b>shutdown(1M)</b> —root login only
<b>Caution</b>	Do not pull the plug until the powerdown procedure is completely finished.
<b>Reference</b>	"Turning the System Off" in Chapter 3, "Processor Operations"

There are differences in the procedure depending on whether you are in multiuser or single-user state.

### From Multiuser

The best way to turn off the computer while the system is in the multiuser state is to enter the **shutdown** 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:

```
These users are currently logged in:  
  ID   terminal number sign-on time  
-----  
root   console           May19 18:06  
jaf    tty22                May19 22:30
```

Step 2: Notify any users that the system is shutting down via the `/etc/wall(1M)` command (see Procedure 2.5). For example:

## Procedure 3.2

---

```
# /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:

```
$ cd /
shutdown -y -i0 -g300
```

The system will proceed to shut itself down after the allotted time, and when the **Reset the CPU to reboot** message appears, the system may be turned off. On an AT386 system, the usual <CTL> <ALT> <DEL> key combination can be pressed, or the system can be turned off and then turned on again in order to reboot at this point. On a Multibus system, the interrupt button or key can be used, or the power can be turned off and on.

## From Single User

If the system is in the single-user state, use the following command to power down 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.
```

```
Reset the cpu to reboot.
```

All services are stopped.

---

## Procedure 3.3: Shutdown to Single User

<b>Purpose</b>	To perform administrative tasks that should be done when no other users are on the system, such as <ul style="list-style-type: none"><li>—software installation</li><li>—file backup and restore</li><li>—hard disk formatting</li><li>—system reconfiguration</li></ul>
<b>Starting Conditions</b>	System state—multiuser. Login—root
<b>Commands</b>	<b>shutdown(1M)</b>
<b>Reference</b>	"Going to Single-User Mode" in Chapter 3, "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 -is
```

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.

---

## Procedure 3.4: Return to Multiuser

<b>Purpose</b>	To make the system available to users after administrative duties have been performed in either the single-user state or the firmware mode.
<b>Starting Conditions</b>	System state—single-user or firmware. Login— <b>root</b>
<b>Commands</b>	<b>init(1M)</b> <b>/unix</b> (boot program)
<b>Reference</b>	"A Look at Entering the Multiuser State" in Chapter 3, "Processor Operations"

There are three system states from which you can return the system to the multiuser 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.

### From Single User

After administrative tasks are finished, you can bring the system back to the multiuser 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 multiuser state. The following is displayed:

### Procedure 3.4

---

```
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 multiuser state.

## From Firmware

When the machine is first powered up, you can bring the system back from the firmware mode by executing the boot program **/unix** from the hard disk.

On the **MB1** system enter 'b' and a <CR>.

```
xxxx xxxxx x
xxxx xxxxx x
Copyright 1986 Intel Corporation.
>
```

On the AT386 enter a <CR>.

```
xxxx xxxxx x
xxxx xxxxx x
512 Base Memory, 02048 Extended
boot:
```

After the the sanity of the root file system is checked [via **fsstat(1M)**], a file system check is performed if necessary [via **fsck(1M)**], the system configuration is printed out, and the system is placed in the multiuser state. Observe the prompt:

Console Login:

You may log in with an appropriate system or user login.



---

## Procedure 3.5: Recovery From System Trouble

<b>Purpose</b>	To return the system to a usable state
<b>Starting Conditions</b>	Variable – running but throughput is degraded – halted or in an unknown state Login – root
<b>sysadm menu</b>	NONE
<b>Commands</b>	<code>/unix</code> – boot program <code>/etc/shutdown(1M)</code>

### Determining the System Trouble

Step 1: Consult the listing of error messages in Appendix C for a description of what happened and what to do about it.

Consult the section on Troubleshooting in the Owner/Operator Manual for problems that occurred during the first time setup of your 80386 computer.

Call your Service Representative if you cannot determine what course of action to take.

Step 2: If the operating system is running, copy any error messages into the system log\*, run shutdown, and reboot the system.

Step 3: If the operating system is not running, copy any error messages into the system log\*, and reboot the system.

\* Log book of system operations, changes, problems, etc.

---

## Procedure 3.6: Reload the Operating System

<b>Purpose</b>	To partially restore the system —to bring it to a usable state —to remove a forgotten root password  To fully restore the system —if there is a new system or disk —if you need to increase the size of the swap, root or /usr partition
<b>Starting Conditions</b>	System state—multiuser or single-user.
<b>Commands</b>	<b>shutdown(1)</b>
<b>Media</b>	The core system diskettes or distribution tape. Any backup data that has to be reloaded.
<b>Caution</b>	A full restore erases everything on the disk. If you use the procedure to change partition sizes, do a complete backup first (Procedure 5.4).
<b>Reference</b>	Release Notes "Instructions for Updating Select Files From the Release". Procedure 3.2 "Powerdown".

Just as file systems have backup versions, so does the operating system: the core diskettes or cartridge tape used to install the system originally. There are two ways to reload the operating system if it is necessary.

- A partial restore replaces (overwrites) the core system files on the hard disk with those originally distributed. These files include the Essential Utilities. The "user" files are not affected by a partial restore.
- A full restore erases everything on the integral hard disk, reformats it, and then loads the core system files.

### **Procedure 3.6**

---

Read through the procedures for reloading the operating system before you do the procedure. If you understand all of the steps, then reload the operating system. If you do not understand the reload procedure, contact your service representative for help.

---

# Instructions for Updating Selected Files from the Release

## MB1 Release

Occasional disk failures may make it necessary to read in some of the files from the release tape without reading the whole release. To do this, it is necessary to read past the first four files on the release. The first four files of the release are binary images, first of the bootstrap, the bootstrap operating system, and two ram-disk images. The *cpio*(1) image of the installed system commands and data follows these.

If you want to load selected files from the release, you need to position the tape to the *cpio* image and *cpio* in whichever files you require. A regular expression can be used to read in whole directories or subtrees. The **-u** option to *cpio* should be used to replace defective executables to assure the file is written regardless of the date in the defective file. This sequence should be performed as **root**. Load the release tape into the tape drive and advance the tape to the end of volume 4 by performing the following sequence of commands:

```
su root
cd /
mt -f /dev/rnmt0 fsf 4
cpio -icBvd </dev/rmt0 regular-expression ...
```

The regular-expression should be the full path name of the files requested, without the initial /. There is no particular order to the files written on the release tape, so you should read the whole tape to make sure you get the files you desire.

For example, to load contents of the directory **/usr/lib/uucp**, you should specify:

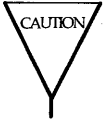
```
cpio -icBvd </dev/rmt0 'usr/lib/uucp/*'
```

## AT386 Release

Diskettes are used instead of cartridge tape. Lists are available for determining which installation diskettes should be searched for the desired files.

## Full System Restore

A full restore erases everything on the hard disk and then loads the system files. Refer to section "Installation and Boot Procedure" in the Release Notes (307-646).



A full system restore should not be performed until you copy any files you want to keep onto a floppy diskette or tape. All files are destroyed in this procedure.

---

## **Disk/Tape Management Procedures**

Disk/Tape Management Procedures	P4-1
Procedure 4.1: Format Diskettes	P4-2
Procedure 4.2: Duplicate Diskettes	P4-5



---

## Disk/Tape Management Procedures

The following procedures are covered in this section:

Procedure 4.1    **Format Diskettes**

To prepare diskettes for use and verify they are usable.

Procedure 4.2    **Duplicate Diskettes**

To make exact copies of diskettes.



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.



---

## Procedure 4.1: Format Diskettes

<b>Purpose</b>	To format a diskette so it can be used by the system.
<b>Starting Conditions</b>	System state—multiuser or single-user You must <b>mount /usr</b> to run this procedure in single-user mode (if <b>/usr</b> is a separate file system). You must be at the computer to insert and remove the diskettes or tapes. Login—an authorized login
<b>sysadm menu</b>	DISK MANAGEMENT TAPE MANAGEMENT
<b>Commands</b>	<b>sysadm format(1)</b>
<b>Media</b>	Unformatted diskettes
<b>Reference</b>	"Formatting Disks" in Chapter 4, "Disk/Tape Management"

Step 1: Enter the following command:

```
$ sysadm format  
Password:
```

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:

```
Insert disk. Press <RETURN> key when ready.  
Do you want each format verified?[(default: yes)[y, n, ?, q]:y
```

#### Procedure 4.1

---

During the format process, you will see a message like the following:

```
Formatting in progress...
Formatted 79 tracks: 0 through 78,
interleave 4
```

Step 3: At the conclusion of the formatting process a message like the following one is displayed on your terminal:

```
The removable disk is now formatted and can be removed.
```

At this point you can remove the diskette or cartridge tape.

---

## Procedure 4.2: Duplicate Diskettes

<b>Purpose</b>	To make a copy of the contents of a diskette.
<b>Starting Conditions</b>	System state—multiuser or single-user You must <b>mount /usr</b> to run this procedure in single-user mode. You must be at the computer to insert and remove diskettes. Login—an authorized login
<b>sysadm menu</b>	DISK MANAGEMENT
<b>Commands</b>	<b>sysadm cpdisk(1)</b>
<b>Media</b>	A new formatted diskette for each one to be copied.
<b>Reference</b>	"Duplicating Disks" in Chapter 4, "Disk/Tape 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 computer is equipped with two diskette drives, duplication is done drive-to-drive.

Step 1: From the console, enter the command:

```
$ sysadm cpdisk  
Password:
```

## Procedure 4.2

---

Step 2: A prompt sequence like this appears on your terminal:

*(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 into the diskette drive.  
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  
insert a writable medium into the diskette drive.  
Press <RETURN> when ready[q]: <CR>  
The original is being copied out onto the duplicate medium.  
Copy out complete  
You may now remove the medium from the diskette 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 a writable medium 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.



---

# File System Administration Procedures

File System Administration Procedures	P5-1
Procedure 5.1: Format and Create File Systems on Diskette	P5-2
Procedure 5.2: Format and Create File Systems on Second Hard Disk	P5-6
Procedure 5.3: Maintain File Systems	P5-11
File System Checking	P5-12
Monitoring Disk Usage	P5-14
Procedure 5.4: File System Backup and Restore	P5-17
Complete Backup	P5-18
Incremental Backup	P5-21
Selective Backup	P5-23
Restore	P5-26
Backup Schedule Reminders	P5-28





---

## File System Administration Procedures

The following procedures are covered in this section:

- Procedure 5.1    **Format and Create File Systems on Diskette**  
To define a file system on a diskette.
- Procedure 5.2    **Format and Create File Systems on Second Hard Disk**  
To define additional file systems when more than one disk device is available.
- Procedure 5.3    **Maintain File Systems**  
To check and possibly repair file systems.  
To monitor disk space usage.  
To reorganize disk space.  
To compress file systems using cartridge tape.
- Procedure 5.4    **File System Backup and Restore**  
To provide a storage copy of active files.  
To archive unneeded files.  
To bring files and file systems back from storage.



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.

---

## Procedure 5.1: Format and Create File Systems on Diskette

<b>Purpose</b>	<p>To define file systems that are removable for reasons of privacy or security.</p> <p>To write identifying labels on the magnetic medium so the system can know what is brought on line.</p> <p>To bring a file system under UNIX system control (<b>mount</b>), or to release it so it can be removed from the system (<b>unmount</b>).</p>
<b>Starting Conditions</b>	<p>System state—(multiuser) or (single-user)</p> <p>You must <b>mount /usr</b> to run this procedure in single-user mode (if <b>/usr</b> is a separate file system).</p> <p>You must be at the computer to insert and remove diskettes or tapes.</p> <p>Login—an authorized login</p>
<b>sysadm menus</b>	DISK MANAGEMENT
<b>Commands</b>	<p><b>sysadm diskmgmt(1)</b></p> <p><b>sysadm makefsys(1)</b></p> <p><b>makefsys(1M)</b></p> <p><b>sysadm mountfsys(1)</b></p> <p><b>sysadm umountfsys(1)</b></p>
<b>Media</b>	<p>One formatted diskette for each file system to be created.</p> <p>Either medium with a file system that needs to be mounted.</p>
<b>Reference</b>	Chapter 5, 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
```

```
Password:
```

```
or
```

```
$ makefsys
```

```
Password:
```

## Procedure 5.1

---

Step 2: 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 diskette drive.  Press <RETURN> when ready: <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 or tape cartridge.)

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 200)[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 3: 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
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]
Disk 'fsys01', file system '/dirfs1', mount it? [y, n, q] y
/dirfs1 mounted. DO NOT REMOVE THE MEDIUM UNTIL IT IS UNMOUNTED!
#
```

- Step 4: 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**.

---

## Procedure 5.2: Format and Create File Systems on Second Hard Disk

<b>Purpose</b>	<p>To define additional file systems on a second hard disk device</p> <ul style="list-style-type: none"><li>—to give a user group a dedicated portion of the disk space</li><li>—to balance the distribution of data on the disk</li></ul> <p>To write identifying labels on the magnetic medium so the system can know what is being brought on line.</p> <p>To bring a file system under UNIX system control (<b>mount</b>), or to release it so it can be removed from the system (<b>umount</b>).</p>
<b>Starting Conditions</b>	<p>System state—(single-user) Login—<b>root</b></p>
<b>Commands</b>	<p><b>mkfs(1M)</b> <b>mkdir(1)</b> <b>mkpart(1M)</b> <b>mount(1M)</b> <b>umount(1M)</b></p>
<b>Media</b>	<p>A second hard disk device.</p>
<b>Reference</b>	<p>"Using <b>mkfs</b>" in Chapter 5, "File System Administration" and the <b>mkfs(1M)</b> and <b>mkpart(1M)</b> manual pages in the System Administrator's Reference Manual</p>

Your computer is delivered with the file systems **root** and **/usr** already defined on the hard disk device.

- Step 1: Install the second hard disk. The second disk is normally daisy-chained to the first disk. This requires reconnecting disk cables, rejumping, and terminating the disk drives. Consult the installation manual for your particular disk drive type.

Step 2: After the second disk drive is installed, boot the system up and create the following device nodes.

Enter:

```
# mknode /dev/rdsk/1s0 C 0 16
# mknode /dev/rdsk/1s1 C 0 17
# mknode /dev/rdsk/1s2 C 0 18
# mknode /dev/rdsk/1s3 C 0 19
```

NOTE

Some systems may have only one file system, '/', and /usr will be a directory on this system.

Step 3: Execute the command 'diskadd' by entering

**#/etc/diskadd**

The system will prompt the user for information about the location of the disk (controller number and drive on that controller), physical characteristics [number of cylinders and (Multibus 1 only) number of heads], and bad disk sectors. It then allows partitioning of the disk into a **tmp** file system, additional swap/paging space, and one to four user file systems.

The user is then asked whether a complete or partial surface analysis should be done and whether the file systems on the new disk should be mounted automatically. Finally, the disk is formatted, verified, and file systems are created. On AT386 systems, the *fdisk* program is run to create a DOS-style partition table which reserves the entire disk for the UNIX system. If automatic mounting was specified, directories are created in the root file system to hold the new file systems, they are mounted, and **/etc/fstab** is updated to remount them on subsequent bootups of the system. Device and partition stanzas for the new disk are appended to the **/etc/partitions** file for use with *mkpart(1M)*.



## Procedure 5.2

---

During the execution of the *mkpart* command, warnings about having no root partition from which to boot will be issued. These may be ignored, as added disks are not bootable.

If swap space is added on the new drive, it must be made available for system use with the *swap(1M)* program. The **diskadd** command doesn't do this, as there is no automatic facility for adding swap space on bootup.

NOTE

Some systems may have only one file system, `/`, and `/usr` will be a directory on this file system.

Step 4: **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
cyl 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 5: Mount the new file system with this command:

```
# mount /dev/dsk/1s1 /usr2
```

*(The arguments on the command line are /dev/dsk/1s1, the name of the device on which the file system resides, and /usr2, the mount point directory.)*

Step 6: 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:

## Procedure 5.2

---

```
#          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 7: The file system is now available for use. This procedure may also be used to define file systems on diskettes.

---

## Procedure 5.3: Maintain File Systems

<b>Purpose</b>	<p>To check and possibly repair removable file systems so the integrity of the file system is assured.</p> <p>To be informed on how disk space is being used so adequate resources can be provided to users.</p>
<b>When Performed</b>	<p>Before mounting the file system.</p> <p>On a schedule appropriate for your circumstances.</p>
<b>Starting Conditions</b>	<p>System state—(multiuser) or (single-user)</p> <p>For checking, the file system must NOT be mounted. For other maintenance, the file system must be mounted.</p> <p>To run this procedure in single-user mode, you must <b>mount /usr</b></p> <p>You must be at the computer to insert and remove the media.</p>
<b>sysadm menus</b>	<p>FILE MANAGEMENT</p> <p>DISK MANAGEMENT</p>
<b>Commands</b>	<p><b>sysadm checkfsys(1)</b></p> <p><b>sysadm diskuse(1)</b></p> <p><b>sysadm fileage(1)</b></p> <p><b>sysadm filesize(1)</b></p>
<b>Media</b>	<p>Diskette or tape that contains file system to be checked.</p>
<b>Reference</b>	<p>"Maintaining a File System" in Chapter 5, "File System" Administration</p>

## File System Checking

Step 1: Insert the diskette into its drive and close the latch.

Step 2: Enter the command:

```
$ sysadm checkfsys  
Password:
```

Step 3: You are then asked to choose a type of checking from this display:

```
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 4: 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.



The error messages are in Chapter 5 under "Running fsck."

Step 5: When the check is completed, a message like this appears:

```
27 files 94 blocks 1266 free
You may now remove the medium from the diskette drive.
```

The operation is finished.

## 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  
Password:
```

Step 2: This display appears on your screen:

```
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 path name 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.



**Procedure 5.3**

---

The heading for the information displayed is:

Owner	File size (characters)	Date last access	File name
-----	-----	-----	-----

The information in the display depends on your application.

---

## Procedure 5.4: File System Backup and Restore

<b>Purpose</b>	To store file systems or parts of file systems. —to guard against loss of data —to free up space on the disk
<b>When Performed</b>	On a schedule developed to fit the needs of your system.
<b>Starting Conditions</b>	System state—(single-user) You must <b>mount /usr</b> to run this procedure in single-user mode. You must be at the computer to insert and remove the media. Login— <b>root</b>
<b>sysadm menu</b>	FILE MANAGEMENT
<b>Commands</b>	<b>sysadm backup(1)</b> <b>sysadm store(1)</b> <b>sysadm restore(1)</b> <b>sysadm bupsched(1)</b> <b>mount(1M)</b>
<b>Media</b>	Cartridge tapes or formatted diskettes in enough quantities to hold the files or file systems you are backing up. Multibus 1 and low density AT diskettes hold 720 blocks. High density diskettes hold 2400 blocks. Tapes hold a minimum of 60,000 blocks.
<b>Reference</b>	"File System Backup and Restore" in Chapter 5, "File System Administration"

## 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 /usr**, if it is a separate file system:

```
# mount /dev/dsk/0s3 /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

Select which drive to use:

1 ctape1 2 diskette1

Enter a number, a name, the initial part of a name, or

? for HELP, q to QUIT: 2

*(Select the storage medium to be used. In the example, we have specified the diskette. If your system includes a cartridge tape drive, select **ctape1** for complete and incremental backups.)*

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.

.  
.
   
.

## Procedure 5.4

---

```
.  
. .  
. .  
    (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 Multiuser).

## Incremental Backup

Incremental backups can be used after an initial full backup has been done. See the discussion on backup strategies in Chapter 5 under "File System Backup and Restore".

- Step 1: Log in 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/0s3 /usr
```

- Step 4: Execute the System Administration backup command (**sysadm backup**) and follow the displayed instructions.

## Procedure 5.4

---

```
# 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>

Select which drive to use:
  1 ctape1      2 diskette1
Enter a number, a name, the initial part of a name, or
? for HELP, q to QUIT:

      (Select the storage medium to be used. If your system
includes a cartridge tape drive, select ctape1 for
complete and incremental backups.)

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 drive. Press <RETURN> when ready. [q] <CR>
/usr2/abc/fil01
.
.
.
83 blocks

Incremental backup of /usr2 has 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 Multiuser).

## 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 /usr**, if it is a separate file system:

```
# mount /dev/dsk/0s3 /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
```



## Procedure 5.4

---

The display on your terminal is

Running subcommand 'store' from menu 'filemgmt',  
FILE MANAGEMENT

*(Prompt for device)*

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

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:

```
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 diskette drive. Press <RETURN> when ready. <CR>
Verification complete. You may remove the medium.
Should the stored files be removed from the built-in disk[y,n,q,?]
```

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.

## Restore

- Step 1: Log in 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 if it is a separate file system:

```
# mount /dev/dsk/0s3 /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 which drive to use

1 diskette 2 tape0

Enter a number, a name, the initial part of a name, or  
? for HELP, q to QUIT: 2

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 tape0 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 the 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 tape0 drive.

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,?]: q

#

## Procedure 5.4

---

- Step 5: Return the system to the normal operating configuration (see Procedure 3.4, Return to Multiuser).
- Step 6: Store the backup diskettes in a safe place.

## Backup Schedule Reminders

You may do this procedure with any authorized login.

- Step 1: Enter the command:

```
$ sysadm bupsched  
Password:
```

- 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 for 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.

## Procedure 5.4

---

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.



## Procedure 5.4

---

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.



---

## **Performance Management Procedures**

System Reconfiguration Procedure P6-1

Procedure 6.1: Unbootable Operating System  
Recovery P6-2



---

## **System Reconfiguration Procedure**

The following procedure is covered in this section:

**Procedure 6.1 Unbootable Operating System Recovery**

To recover if you create an unbootable operating system after attempting a system reconfiguration.

---

## Procedure 6.1: Unbootable Operating System Recovery

<b>Purpose</b>	To recover from an unbootable <b>/unix</b> . To get a viable version of the system running after an unsuccessful attempt at reconfiguring the system.
<b>Starting Conditions</b>	System state—variable. Login— <b>root</b>
<b>sysadm menu</b>	MACHINE MANAGEMENT
<b>Bootable Programs</b>	<b>/oldunix</b> —boot program (from hard disk)
<b>Reference</b>	Chapter 6, "Performance Management"

If you create a **/unix** that is unbootable or operates so poorly that recovery while operating in that version is impossible, do the following steps.

Step 1: On the AT386, type the following at boot time:

```
boot: /oldunix <CR>
```

On the MB1, type in the following at boot time:

```
>b":w0:oldunix" <CR>
```



If you did not make this copy, reload the operating system using the core system diskettes or installation tape via the partial restore procedure (see Procedure 3.6, Reload the Operating System).

Step 2: 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 3: 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 4: If you cannot determine what went wrong, document what happened as thoroughly as you can, and then contact your service representative.





---

## **LP Spooler Administration Procedures**

LP Spooler Administration Procedures	P7-1
Procedure 7.1: Install the LP Spooler	P7-2
Procedure 7.2: Shut Down for Maintenance	P7-3
Procedure 7.3: Restart the Scheduler	P7-4
Procedure 7.4: Add Another LP Printer	P7-6



---

## LP Spooler Administration Procedures

The following procedures are covered in this section:

- Procedure 7.1    **Install the LP Spooler**  
To install the LP Spooler Utilities and an LP printer in the computer: references to pertinent documentation.
- Procedure 7.2    **Shut Down for Maintenance**  
To either turn off the scheduler or a printer for administrative or maintenance purposes.
- Procedure 7.3    **Restart the Scheduler**  
To restart the LP scheduler and enable any printers.
- Procedure 7.4    **Add Another LP Printer**  
To add another LP printer to the current configuration.

---

## Procedure 7.1: Install the LP Spooler

<b>Purpose</b>	Provide checklist of essential information needed to install the LP Spooler.
<b>References</b>	Owner/Operator Manual Appropriate printer manuals

---

## Procedure 7.2: Shut Down for Maintenance

<b>Purpose</b>	To turn off either the scheduler or a printer.
<b>Starting Conditions</b>	Multiuser or single-user state Login—an authorized login ( <b>root</b> )
<b>Commands</b>	<b>lpstat(1)</b> <b>/usr/lib/lpshut(1M)</b>
<b>Reference</b>	"Administrative Commands" in Chapter 7, "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.

---

## Procedure 7.3: Restart the Scheduler

<b>Purpose</b>	To make the LP system available again after administrative tasks or maintenance.
<b>Starting Conditions</b>	Multiuser or single-user state Login—an authorized login
<b>Commands</b>	<code>/usr/lib/lpsched(1M)</code> <code>enable(1)</code> <code>/usr/lib/accept(1M)</code>
<b>Reference</b>	"Administrative Commands" in Chapter 7, "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
```



---

## Procedure 7.4: Add Another LP Printer

<b>Purpose</b>	To meet the need for expanded output facilities.
<b>Starting Conditions</b>	Multiuser or single-user state You must <b>mount /usr</b> to run this procedure in single-user mode. Login— <b>root</b>
<b>Commands</b>	<b>lpstat(1)</b> <b>/usr/lib/lpshut(1M)</b> <b>lpadmin(1M)</b> <b>/usr/lib/lpshed(1M)</b> <b>/usr/lib/accept(1M)</b> <b>enable(1)</b>
<b>Reference</b>	"Administrative Commands" in Chapter 7, "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 -pprinter -vdevice [-eprinter!-iinterface!-mmodel]
```

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.



---

## **TTY Management Procedures**

TTY Management Procedures	P8-1
Procedure 8.1: Check TTY Line Settings	P8-2
Procedure 8.2: Make TTY Line Settings	P8-6
Procedure 8.3: Modify TTY Line Characteristics	P8-8



---

## TTY Management Procedures

- Procedure 8.1    **Check TTY Line Settings**  
To tell what line settings are defined.
- Procedure 8.2    **Make TTY Line Settings**  
To create new TTY line settings and hunt sequences.
- Procedure 8.3    **Modify TTY Line Characteristics**  
To change the characteristics of TTY lines. To turn lines on or off.



The procedures described in this section apply to Release 2.1 and later releases.

---

## Procedure 8.1: Check TTY Line Settings

<b>Purpose</b>	To tell what line settings are defined.
<b>Starting Conditions</b>	System state—multiuser or single-user Login—an authorized login
<b>sysadm menu</b>	TTY MANAGEMENT
<b>Commands</b>	<b>sysadm lineset(1)</b>
<b>Reference</b>	"How the TTY System Works" in Chapter 8, "TTY Management"

Step 1: Enter this command to go directly to the lineset display:

```
$ sysadm lineset  
Password:
```

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.



## Procedure 8.1

---

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
BRKINT     Signal interrupt on break
CREAD      Enable receiver
CS8        8 bit character
ECHO       Enable echo
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

Press the RETURN key to see the ttygmt menu [?, q]: q

---

## Procedure 8.2: Make TTY Line Settings

<b>Purpose</b>	To create new line settings and hunt sequences.
<b>Starting Conditions</b>	System state—multiuser or single-user Login—an authorized login
<b>sysadm menu</b>	TTY MANAGEMENT
<b>Commands</b>	<b>sysadm mklineset(1)</b>
<b>Reference</b>	"How to Create New Line Settings and Hunt Sequences" in Chapter 8, "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
Password:
```

Step 2: This sequence of prompts appears on your terminal:

```

Running subcommand 'mklineset' from menu 'ttypgmt',
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.

Press the RETURN key to see the ttypgmt menu [?, q]: q

```

---

## Procedure 8.3: Modify TTY Line Characteristics

<b>Purpose</b>	To modify TTY line settings or turn line on or off.
<b>Starting Conditions</b>	System state—multiuser or single-user Login—an authorized login
<b>sysadm menu</b>	TTY MANAGEMENT
<b>Commands</b>	<b>sysadm modtty(1)</b>
<b>Reference</b>	"How to Modify TTY Line Characteristics" in Chapter 8, "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  
Password:
```

Step 2: This sequence of prompts appears on your terminal:

```

Running subcommand 'modtty' from menu 'ttypgmt',
TTY MANAGEMENT

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      Line
  ---      ---      ---      ---      ---
  contty   on         60         4800
  tty11    off        off        9600
  tty12    off        off        9600
  tty13    off        off        9600
  tty14    off        off        9600
  tty15    off        off        9600

Continue (default: y)? [y, n, q] <CR>

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]: 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.)
.
.
.

```

## Procedure 8.3

.  
.  
.

Available line settings:

console	console4	contty2	pty	1200H	4800	9600	19200H
console1	console5	contty3	300	1200	4800H	9600H	1200300
console2	contty	contty4	300H	2400	4800	19200	3001200
console3	contty1	contty5	1200	2400H			

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	
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	

Continue (default: y)? [y, n, q] q

---

# Basic Networking Procedures

Basic Networking Procedures	P9-1
Procedure 9.1: Install Basic Networking Software	P9-2
Run <code>sysadm installpkg</code>	P9-2
Procedure 9.2: Set Up Basic Networking Files	P9-5
Set Up Devices File - <code>devicemgmt</code>	P9-7
Set Up <code>/etc/inittab</code> - <code>portmgmt</code>	P9-8
Set Up Systems File - <code>systemmgmt</code>	P9-10
Set Up Poll File - <code>pollmgmt</code>	P9-12
Set Up Permissions File	P9-14
Set Up <code>Devconfig</code> File	P9-14
Set Up <code>Sysfiles</code> File	P9-15
Add <code>uucp</code> logins	P9-16
Other Networking Files	P9-16
Procedure 9.3: Basic Networking Maintenance	P9-18
Automated Networking Maintenance ( <code>cron</code> )	P9-18
<code>uudemon.poll</code>	P9-18
<code>uudemon.hour</code>	P9-19
<code>uudemon.admin</code>	P9-19
<code>uudemon.cleanup</code>	P9-20
Manual Maintenance	P9-20
Procedure 9.4: Basic Networking Debugging	P9-22
Check for Faulty ACU/Modem	P9-22
Check Systems File	P9-22
Debug Transmissions	P9-23
Check Error Messages	P9-24



## Table of Contents

---

Check Basic Information	P9-24
<b>Procedure 9.5: Remove BNU Software</b>	P9-25
Prerequisites	P9-25
Single-User Mode	P9-26
Run <b>sysadm removepkg</b>	P9-27
Return to Multiuser Mode	P9-27
<b>Procedure 9.6: Set Up BNU STARLAN NETWORK (Basic)</b>	P9-28
Prerequisites	P9-28
Create STARLAN <b>Systems</b> Entry	P9-29
Create STARLAN <b>Devices</b> Entry	P9-30
Create STARLAN <b>Devconfig</b> Entries	P9-30
Set Up STARLAN NETWORK Listener	P9-31
STARLAN <b>Dialers</b> Entry	P9-33
<b>Procedure 9.7: Set Up BNU STARLAN NETWORK (Special)</b>	P9-34

---

## Basic Networking Procedures

The following procedures are covered in this section:

- Procedure 9.1    **Install Basic Networking Software**  
To place basic networking software on the hard disk.
- Procedure 9.2    **Set Up Basic Networking Files**  
To configure basic networking files.
- Procedure 9.3    **Basic Networking Maintenance**  
To maintain basic networking files and operations.
- Procedure 9.4    **Basic Networking Debugging**  
To track down problems in basic networking.
- Procedure 9.5    **Remove BNU Software**  
To remove basic networking software from the hard disk.
- Procedure 9.6    **Set Up BNU on STARLAN NETWORK (Basic)**  
To show how BNU files are created for a STARLAN NETWORK.
- Procedure 9.7    **Set Up BNU on STARLAN NETWORK (Special)**  
To show how BNU files are created for a STARLAN NETWORK if **cu** and **uucico** services are to be handled differently.

---

## Procedure 9.1: Install Basic Networking Software

<b>Purpose</b>	To place basic networking software on the hard disk.
<b>Starting Conditions</b>	System state—2 (multiuser) or 1 (single-user) You must be at the computer to insert and remove diskettes. You must <b>mount /usr</b> to run this procedure in single-user mode. Login— <b>root</b>
<b>sysadm menu</b>	SOFTWARE MANAGEMENT
<b>Commands</b>	<b>sysadm installpkg(1)</b>
<b>Media</b>	The basic networking utilities diskette.
<b>Reference</b>	<i>Basic Networking Utilities Guide</i>

This procedure describes Basic Networking installation. The Basic Networking Utilities for the 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.

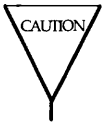
### Run **sysadm installpkg**

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 computer will display its current node name and ask if you want to define or rename it.



Changing the node name of your 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 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 file names 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.

## Procedure 9.1

---



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.

---

## Procedure 9.2: Set Up Basic Networking Files

<b>Purpose</b>	To configure basic networking files. To ensure proper communications links.
<b>When Performed</b>	Initial setup and when adding new devices or remote systems.
<b>Starting Conditions</b>	System state—2 (multiuser) or 1 (single-user). You must <b>mount /usr</b> to run this procedure in single-user mode. Login—an authorized login
<b>sysadm menu</b>	PACKAGE MANAGEMENT
<b>Commands</b>	<b>sysadm uucpmgmt(1)</b> <b>sysadm devicemgmt(1)</b> <b>sysadm portmgmt(1)</b> <b>sysadm systemmgmt(1)</b> <b>sysadm pollmgmt(1)</b>
<b>Reference</b>	<i>Basic Networking Utilities Guide</i>

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 setup of several optional files is described.

## Procedure 9.2

---

Display the **uucp** System Administration submenu by entering:

```
$ sysadm uucp
```

```
Password:
```

```
BASIC NETWORKING UTILITIES MANAGEMENT
```

- 1 devicemgt      manage devices (list, add, delete)
- 2 pollmgt        manage poll entries (list, add, delete)
- 3 portmgt        manage I/O ports (list, modify)
- 4 systemmgt     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 in the following text can be accessed from this display or directly from the shell.

## 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 9 in Part 2 of this book.

Step 1: To add entries into the **Devices** file, type **sysadm devicemgmt**, then select 2 (add):

```
$ sysadm devicemgmt
```

```
Password:
```

```
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
```



## Procedure 9.2

---

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.

### 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 3, "Processor Operations".

Step 1: To add BNU entries to the **inittab** file, type **sysadm portmgmt**, then select **2** (modify).

```
$ sysadm portmgmt
```

```
Password:
```

```
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
```

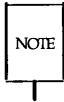
The subcommand lists the ports available to be used by Basic Networking, then asks you to choose which one you want to modify. It then prompts you for the following information:

|                   |                                                                                                                                                              |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 <b>incoming</b> only, <b>outgoing</b> only, or <b>bidirectional</b> . |
| baud rate         | Enter the speed (baud rate) of the selected port.                                                                                                            |

## Procedure 9.2

---

After you have entered the requested information, it will be displayed to you before it is entered into the `/etc/inittab` file.



Since adding a device (**sysadm devicemgmt**) automatically creates a port entry in `/etc/inittab`, you may only need to use **sysadm portmgmt** for modifications.

## 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 computer, refer to Chapter 9 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
```

```
Password:
```

```
Running subcommand 'systemmgmt' from menu 'uucprgmt',  
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.              |

## Procedure 9.2

---

|              |                                                                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| phone number | The telephone number of the remote machine. Special symbols can be embedded in the phone number, including abbreviations from the <b>Dialcodes</b> file ( <b>/usr/lib/uucp/Dialcodes</b> ). |
| login ID     | Used by <b>uucp</b> 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.

## 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 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
```

```
Password:
```

```
Running subcommand 'pollmgmt' from menu 'uucpnmgt',  
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.

## Set Up Permissions File

The default `/usr/lib/uucp/Permissions` file provides the maximum amount of security for your 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 9 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.

## Set Up Devconfig File

The `/usr/lib/uucp/Devconfig` file is only needed if you are using BNU over an AT&T STARLAN NETWORK or some other Streams-based transport provider that conforms to the AT&T Transport Interface (TI). If you are using an AT&T STARLAN NETWORK, the two entries shown in the **Devconfig** file are all you need in this file.

```
service=cu      device=STARLAN  push=ntty:tirdwr:ld0
service=uucico  device=STARLAN  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 STARLAN 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 STARLAN.)

**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.

## 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 9. 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.

## 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**.

## 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 UNIX 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.)

---

## Procedure 9.3: Basic Networking Maintenance

|                            |                                                                               |
|----------------------------|-------------------------------------------------------------------------------|
| <b>Purpose</b>             | To keep files related to basic networking from consuming too much disk space. |
| <b>When Performed</b>      | Automatically with <b>cron(1M)</b> or as needed.                              |
| <b>Starting Conditions</b> | System state—2 (multiuser) 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.

### 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 your computer is in run state 2 (multiuser), **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.

#### **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"
```

## **uudemon.hour**

The **uudemon.hour** shell script you receive with your machine does the following:

- Calls the **uusched** program to search the spool directories for work files (C.) that have not been processed and schedules these files for transfer to a remote machine.
- Calls the **uuxqt** daemon to search the spool directories for execute files (X.) that have been transferred to your computer and were not processed at the time they were transferred.

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.

## **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.

## Procedure 9.3

---

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
```

## uudemon.cleanup

The delivered `uudemon.cleanup` 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.) 2 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
```

## 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 multiuser state.

---

## Procedure 9.4: Basic Networking Debugging

|                            |                                                                                                                                                      |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>             | To use available monitoring tools to solve basic networking problems.                                                                                |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 1 (single-user)                                                                                                        |
| <b>Commands</b>            | <code>uustat(1)</code><br><code>cu(1)</code><br><code>Uutry(1)</code><br><code>uuname(1M)</code><br><code>uulog(1)</code><br><code>uucheck(1)</code> |

These procedures describe how to go about solving common problems that may be encountered with Basic Networking Utilities.

### Check for Faulty ACU/Modem

You can check if the automatic call units or modems are not working properly in several ways.

- Run `uustat -q`. This will give counts and reasons for contact failure.
- Run `cu -d -lline`. This will let you call over a particular line and print debugging information on the attempt. The line must be defined as Direct in the devices file. (You must add a telephone number to the end of the command line if the line is connected to an autodialer or the device must be set up as **direct**.)

### Check Systems File

Check that you have up-to-date information in your systems file if you are having trouble contacting a particular machine.

Some things that may be out of date for a machine are its:

- Phone number
- Login
- Password

## Debug Transmissions

If you are unable to contact a particular machine, you can check out communications to that machine with **Utry** and **uucp**.

Step 1: To simply try to make contact, run:

```
$ /usr/lib/uucp/Utry -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 **Utry** 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 **Utry** again. If you still cannot solve the problem, you may need to call support personnel. Save the debugging output; it will help diagnose the problem.



## Check 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 your computer attempts to communicate with. These files contain status information on the attempted communication and whether it was successful.

## Check Basic Information

There are several commands you can use to check for basic networking information.

- |                   |                                                                                                                                                                                                              |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>uuname</b>     | Use this command to list those machines your machine can contact.                                                                                                                                            |
| <b>uulog</b>      | Use this command to display the contents of the log directories for particular hosts.                                                                                                                        |
| <b>uucheck -v</b> | Run this command to check for the presence of files and directories needed by <b>uucp</b> . This command also checks the <b>Permissions</b> file and outputs information on the permissions you have set up. |

---

## Procedure 9.5: Remove BNU Software

|                            |                                                                                                                                        |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>             | To remove the software from the hard disk                                                                                              |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 1 (single-user)<br>You must be at the computer to insert and remove diskettes.<br><br>Login— <b>root</b> |
| <b>sysadm menu</b>         | SOFTWARE MANAGEMENT                                                                                                                    |
| <b>Commands</b>            | <b>sysadm removepkg(1)</b>                                                                                                             |
| <b>Media</b>               | The basic networking utilities diskette.                                                                                               |

This procedure describes how to remove Basic Networking Utilities software from your 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.



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**.

### 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.

## Procedure 9.5

---

```
/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.

## 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.)*

## 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.

## Return to Multiuser 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 computer is available for use with the exception of the Basic Networking Utilities.

---

## Procedure 9.6: Set Up BNU STARLAN NETWORK (Basic)

|                            |                                                                                   |
|----------------------------|-----------------------------------------------------------------------------------|
| <b>Purpose</b>             | To show how to set up BNU files for an AT&T STARLAN NETWORK using <b>sysadm</b> . |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 1 (single-user)<br>Login— <b>root</b>               |
| <b>Commands</b>            | <b>sysadm uucpmgmt(1)</b>                                                         |

This procedure contains an example of setting up BNU files for a STARLAN NETWORK. Though the example is particular to setting up BNU on a STARLAN NETWORK, the concepts could be applied to setting up BNU to run on a transport provider that is compatible with the AT&T 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 STARLAN NETWORK consists of trusted machines.

### Prerequisites

Before running this procedure to set up BNU to run on a STARLAN NETWORK, you must install these packages in the order shown:

- Networking Support Utilities
- STARLAN NETWORK
- Basic Networking Utilities

## Create STARLAN 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 STARLAN NETWORK. (The convention is to use node name. However, some networks use 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 **STARLAN** to choose that name as the network name. (The word **STARLAN** 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, STARLAN 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 STARLAN 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.

## Create STARLAN 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 **starlan** as the name of the port. (In this case you must use the literal **starlan**. 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 **STARLAN** 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.

## Create STARLAN 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=STARLAN  push=ntty:tirdwr:ld0
service=uucico  device=STARLAN  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.

## Set Up STARLAN NETWORK Listener

STARLAN and other TLI networks running on the UNIX 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 STARLAN NETWORK. The following steps show you how to check that STARLAN NETWORK is set up properly to support BNU and what to enter in case it isn't.

Step 1: Type:

```
nlsadmin -v starlan
```



## Procedure 9.6

---

You should see:

```
1  ENABLED  NOMODULES  /usr/sbin/lib/ttysrv  comment
102  ENABLED  NOMODULES  /usr/sbin/lib/ttysrv  comment
```

If you don't see the above, then type:

```
# nlsadmin -a 1 -c "/usr/sbin/lib/ttysrv" -y "tty login service" -m starlan
# nlsadmin -a 102 -c "/usr/sbin/lib/ttysrv" -y "tty login service" starlan
```

Step 2: Type:

```
nlsadmin -x
```

You should see:

```
starlan ACTIVE
```

If you see:

```
starlan INACTIVE
```

then type:

```
nlsadmin -s starlan
```

to start it up. If you don't see either `starlan ACTIVE` or `starlan INACTIVE`, then the STARLAN NETWORK was not properly installed.

## STARLAN Dialers Entry

You do not need to create an entry in the **Dialers** file for standard BNU services over STARLAN as shown in the previous procedure.

BNU is now ready to communicate over a STARLAN network.

---

## Procedure 9.7: Set Up BNU STARLAN NETWORK (Special)

|                            |                                                                                                                         |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>             | To show how to set up BNU files for an AT&T STARLAN NETWORK to handle <b>uucico</b> and <b>cu</b> services differently. |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 1 (single-user)<br>Login— <b>root</b>                                                     |
| <b>Commands</b>            | Any text editor [ <b>ed(1)</b> , <b>vi(1)</b> ]                                                                         |

In the previous procedure, both **uucico** and **cu** services go through a login procedure to connect to other computers on a STARLAN NETWORK. To improve performance of **uucico** services, you can have **uucico** requests connect directly to remote machines on the STARLAN NETWORK without going through login.

You should only set up **uucico** as described above if you trust all machines on your STARLAN 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 file names you want. The ones shown above are only given as an example.

- Step 2: The systems you want to transfer files to on your STARLAN 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.

```
3b2abc Any ListenCico - 3b2abc.serve
```

In this example, **3b2abc** 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 **3b2abc.serve** is the STARLAN NETWORK address of **3b2abc**.

- 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 starlan - - TLI \D listencico
```

## Procedure 9.7

---

In this example, **ListenCico** is the device name, **starlan** identifies the network, the two dashes are field place holders, **TLI** is the network type, **\D** says read in the network address (**3b2abc**), and **listencico** is the dialer type.



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\Nc
```

- 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" starlan
```

This entry says to answer requests for service code 101 with **uucico** directly as it is used for a **TLI** (AT&T Transport Interface) network. Notice that the default **nuucp** login ID is used; you can use a different one if you choose.

---

## Remote File Sharing Procedures

|                                        |        |
|----------------------------------------|--------|
| Remote File Sharing Procedures         | P10-1  |
| Procedure 10.1: Domain Setup (Primary) | P10-2  |
| Basic Domain Setup                     | P10-2  |
| Prerequisites                          | P10-2  |
| Configuration                          | P10-3  |
| Optional Domain Setup                  | P10-9  |
| Procedure 10.2: Host Setup             | P10-12 |
| Basic Host Setup                       | P10-12 |
| Prerequisites                          | P10-12 |
| Configuration                          | P10-13 |
| Optional Host Setup                    | P10-16 |
| Procedure 10.3: Advertise Resources    | P10-20 |
| Procedure 10.4: Mount Remote Resources | P10-23 |



---

## Remote File Sharing Procedures

The following procedures are covered in this section.

- Procedure 10.1     **Domain Setup (primary)**  
To set up the Remote File Sharing primary name server for the domain.
- Procedure 10.2     **Host Setup**  
To set up a host computer and connect it to the Remote File Sharing domain.
- Procedure 10.3     **Advertise Resources**  
To make your directories available to other hosts.
- Procedure 10.4     **Mount Remote Resources**  
To make remote directories available to your host.



---

## Procedure 10.1: Domain Setup (primary)

|                            |                                                                                                                                              |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>             | To set up the primary domain name server for a Remote File Sharing Domain.                                                                   |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 1 (single-user)<br>You must <b>mount /usr</b> to run this procedure in single-user mode.<br>Login— <b>root</b> |
| <b>Commands</b>            | <b>uname(1M)</b><br><b>nlsadmin(1M)</b><br><b>dname(1M)</b><br><b>rfadmin(1M)</b><br><b>rfstart(1M)</b><br><b>adv(1M)</b>                    |
| <b>Reference</b>           | Chapter 10, "Remote File Sharing"                                                                                                            |

Before beginning this procedure, read Chapter 10, "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.

### 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.

---

## Procedure 10.1: Domain Setup (primary)

These are defined simply to keep the name server running temporarily, should the primary fail.

- UNIX System V Release 3.0 software, Remote File Sharing software, Networking Support Utilities, and network software must be installed and running. (See the *Remote File Sharing Release Notes* and the network manuals that accompany the product for installation instructions.)

STARLAN 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 AT&T Transport Interface specification.

- Log in as **root**. All domain administration must be done with **root** permission from the primary domain name server.

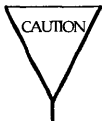
### Configuration

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 AT&T STARLAN 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.



Changing the node name of your 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 STARLAN NETWORK and Remote File Sharing in the order described in the *Remote File Sharing Release Notes*, you can skip this step. The

## Procedure 10.1: Domain Setup (primary)

---

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 STARLAN NETWORK listener is improperly set up, this step shows how to manually set up the listener. In the following example STARLAN NETWORK is used. To set up the listener for other networks compatible with the AT&T Transport Interface, you would replace **starlan** 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 starlan
```

to initialize the files needed for the listener process for the network specified, in this case **starlan**.

Next type (on one line):

```
nlsadmin -a 105 -c /usr/net/servers/rfs/rfsetup -y \  
"rfsetup" starlan
```

to add the Remote File Sharing service (**rfsetup**) to the list of services available to the **starlan** listener.

Use the following command line to report the status of the **starlan** listener process installed on this machine (ENABLED or DISABLED):

```
nlsadmin -x starlan
```

Next type:

```
nlsadmin -l "nodename.serve" -t "nodename" starlan
```

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 different address types.]

To start the listener, type:

```
nlsadmin -s starlan
```

Normally, it will be started automatically when your machine enters multiuser 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: To identify the network, you must tell Remote File Sharing the network (transport provider) it should use. (In our example, this is **starlan** for STARLAN NETWORK.)

```
# dname -N starlan
```

This command indicates the device, relative to the **/dev** directory, that is used for the transport provider.

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**, **fie**, and **foe**. Adding each machine's domain name (**graph**) to its node name, separated by a period, forms its full Remote File Sharing host name.

## Procedure 10.1: Domain Setup (primary)

---

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 domain **graph** another secondary is **graph.foe**.
- For host **graph.fee** the network address is **fee.serve**.
- For host **graph.fie** the network address is **fie.serve**.
- For host **graph.foe** the network address is **foe.serve**.

(The addresses shown are an example of STARLAN NETWORK addresses. These addresses should be in the form *nodename.serve*.)

|           |   |           |
|-----------|---|-----------|
| graph     | p | graph.fee |
| graph     | s | graph.fie |
| graph     | s | graph.foe |
| graph.fee | a | fee.serve |
| graph.fie | a | fie.serve |
| graph.foe | a | foe.serve |

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*).

---

## Procedure 10.1: Domain Setup (primary)

- 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:

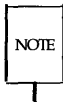
```
# rfadmin -a domain.nodename  
Enter password for nodename:  
Re-enter password for nodename:
```

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.)

## Procedure 10.1: Domain Setup (primary)

---

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.



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 "Remote File Sharing Mode" in Chapter 10 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.

## 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.



## Procedure 10.1: Domain Setup (primary)

---

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 the "Mapping Remote Users" section of Chapter 10 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**. The following example shows the **rfmaster** file created earlier in this section, with information added to contact a domain called **docs**.

---

## Procedure 10.1: Domain Setup (primary)

|             |   |              |
|-------------|---|--------------|
| graph       | p | graph.fee    |
| graph       | s | graph.fie    |
| graph       | s | graph.foe    |
| graph.fee   | a | fee.serve    |
| graph.fie   | a | fie.serve    |
| graph.foe   | a | foe.serve    |
|             |   |              |
| docs        | p | docs.big     |
| docs        | s | docs.little  |
| docs.big    | a | big.serve    |
| docs.little | a | little.serve |

---

## Procedure 10.2: Host Setup

|                            |                                                                                                                                              |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b>             | To set up a host on a Remote File Sharing Domain.                                                                                            |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 1 (single-user)<br>You must <b>mount /usr</b> to run this procedure in single-user mode.<br>Login— <b>root</b> |
| <b>Commands</b>            | <b>uname(1M)</b><br><b>nlsadmin(1M)</b><br><b>dname(1M)</b><br><b>rfstart(1M)</b><br><b>mount(1M)</b>                                        |
| <b>Reference</b>           | 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.

### 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. UNIX System V Release 3.0 software, Remote File Sharing software, Networking Support Utilities, and network software must be installed and running. (See the *Remote File Sharing Release*

*Notes* and the network manuals that accompany the product for installation instructions.)

STARLAN 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 AT&T Transport Interface specification.

- Log in as **root**. All host administration must be done with **root** permission on your host.

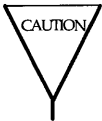
### Configuration

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 AT&T STARLAN 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.



Changing the node name of your 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 STARLAN NETWORK and Remote File Sharing in the order described in the *Remote File Sharing Release 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 STARLAN NETWORK listener is improperly set up, this step shows how to manually set up the *listener*.

## Procedure 10.2: Host Setup

---

In the following example STARLAN NETWORK is used. To set up the listener for other networks compatible with the AT&T Transport Interface, you should replace **starlan** 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 starlan
```

to initialize the files needed for the listener process for the network specified, in this case **starlan**.

Next type (on one line):

```
nlsadmin -a 105 -c /usr/net/servers/rfs/rfsetup -y \  
"rfsetup" starlan
```

This will add the Remote File Sharing service (**rfsetup**) to the list of services available to the **starlan** listener.

```
nlsadmin -x starlan
```

will report the status of the starlan listener process installed on this machine.

Next type:

```
nlsadmin -l "nodename.serve" -t "nodename" starlan
```

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 different address types.]

To start the listener, type:

```
nlsadmin -s starlan
```

Normally, it will be started automatically when your machine enters multiuser 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 **starlan**.)

```
# dname -N starlan
```

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. (For STARLAN NETWORK, the address should be in the form *nodename.serve*, where *nodename* is replaced by the communications node name of the primary name server.)

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.

## Procedure 10.2: Host Setup

---

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 "Remote File Sharing Mode" in Chapter 10 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.

## Optional Host Setup

The following steps are needed only 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



## Procedure 10.2: Host Setup

---

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 10 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 "Mapping Remote Users" in Chapter 10.)

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.

---

## Procedure 10.3: Advertise Resources

|                            |                                                                   |
|----------------------------|-------------------------------------------------------------------|
| <b>Purpose</b>             | Share your resources with other hosts.                            |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 3 (RFS state)<br>Login— <b>root</b> |
| <b>Commands</b>            | <b>adv(1M)</b>                                                    |
| <b>Reference</b>           | Chapter 10, "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 UNIX 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 UNIX 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 path name 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 10 for the format of **host.alias**.)

### Procedure 10.3: Advertise Resources

---

Here 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 10 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
#
```

---

## Procedure 10.4: Mount Remote Resources

|                            |                                                                   |
|----------------------------|-------------------------------------------------------------------|
| <b>Purpose</b>             | Use remote resources on your machine.                             |
| <b>Starting Conditions</b> | System state—2 (multiuser) or 3 (RFS state)<br>Login— <b>root</b> |
| <b>Commands</b>            | <b>nsquery</b> (1M)<br><b>mkdir</b> (1)<br><b>mount</b> (1M)      |
| <b>Reference</b>           | Chapter 10, "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.

## Procedure 10.4: Mount Remote Resources

---

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.

---

# Chapter 1: System Identification and Security

|                                           |      |
|-------------------------------------------|------|
| Introduction                              | 1-1  |
| Important Security Guidelines             | 1-2  |
| Logins and Passwords                      | 1-3  |
| Password Aging                            | 1-3  |
| Sample <code>/etc/passwd</code> Entries   | 1-5  |
| Change Every Two Weeks                    | 1-5  |
| Change One Time Only                      | 1-5  |
| Change by <code>root</code> Only          | 1-6  |
| Locking Unused Logins                     | 1-7  |
| Special Administrative Passwords          | 1-7  |
| Set-UID and Set-GID                       | 1-10 |
| Check Set-UIDs Owned by <code>root</code> | 1-10 |
| Check Set-UIDs in the Root File System    | 1-11 |
| Check Set-UIDs in Other File Systems      | 1-13 |





---

## Introduction

This chapter deals with maintaining the security of your 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.

---

## 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 tamper-proof. 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**, **/sbin**, 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**.

---

## Logins and Passwords

The discussion of logins and passwords covers

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

### 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 **M**aximum duration of the password (in weeks, following the notation in Figure 1-1).
- m* The **m**inimum time interval before the existing password can be changed by the user (in weeks, following the notation in Figure 1-1).

## Logins and Passwords

---

*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.

All times are specified in weeks (0 through 63) by a 64-character alphabet. Figure 1-1 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.

| 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   |

Figure 1-1: Password Aging Character Codes

---

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.

## 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:
```

## Logins and Passwords

---

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.

## 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:
```

## 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).



## Logins and Passwords

---

The commands, which are also logins, that can be password-protected perform functions that might be needed by a number of users on your computer:

| Function          | Use                                                                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <b>setup</b>      | This command is used to set up your computer. Once the machine has been set up, you do not want anyone doing it again without your knowledge. |
| <b>sysadm</b>     | This command allows access to many useful administrative functions that do not require a user to log in as root.                              |
| <b>powerdown</b>  | This command powers the system down.                                                                                                          |
| <b>checkfsys</b>  | This command initiates a file system check on the specified file systems.                                                                     |
| <b>makefsys</b>   | This command makes a new file system on the specified media.                                                                                  |
| <b>mountfsys</b>  | This command mounts the specified file system for use.                                                                                        |
| <b>umountfsys</b> | 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 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.

| <b>Login</b>   | <b>Use</b>                                                                                                                                                                                                                                 |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>root</b>    | 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 <b>root</b> login should be very carefully protected. |
| <b>sys</b>     | This login has the power of a normal user login over the files it owns, which are in <b>/usr/src</b> .                                                                                                                                     |
| <b>bin</b>     | This login has the power of a normal user login over the files it owns, which are in <b>/bin</b> .                                                                                                                                         |
| <b>adm</b>     | This login has the power of a normal user login over the object files it owns, which are in <b>/usr/adm</b> .                                                                                                                              |
| <b>uucp</b>    | This login owns the object and spooled data files in <b>/usr/lib/uucp</b> .                                                                                                                                                                |
| <b>nuucp</b>   | This login is used by remote machines to log into the system and initiate file transfers via <b>/usr/lib/uucp/uucico</b> .                                                                                                                 |
| <b>rje</b>     | This login has an entry in <b>/etc/passwd</b> . There are no files currently associated with this login.                                                                                                                                   |
| <b>daemon</b>  | This is the login of the system daemon, which controls background processing.                                                                                                                                                              |
| <b>trouble</b> | This login has the power of a normal user login over the files it owns, which are in <b>/usr/lib/trouble</b> .                                                                                                                             |
| <b>lp</b>      | This login owns the object and spooled data files in <b>/usr/spool/lp</b> .                                                                                                                                                                |

---

## 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 switch 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)**.

### 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-sr-x  1 root  bin      33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-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-sr-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
? d
#

```

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.

## 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/0s1**). 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.

## Set User and Group IDs

---

```
# ls -l 'ncheck -s /dev/dsk/0s1 | 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
-rwsr-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.

## 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/0s3**, 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.



**/usr** may not be a separate file system on your system.

## Set User and Group IDs

---

```
# ncheck -s /dev/dsk/0s3 | cut -f2
/dev/dsk/c1d0s2:
/bin/at
/bin/crontab
/bin/sh1
/bin/sadp
/bin/timex
/bin/cancel
/bin/disable
/bin/enable
/bin/lp
/bin/lpstat
/bin/ct
/bin/cu
/bin/uucp
/bin/uname
/bin/uustat
/bin/uux
/lib/mv_dir
/lib/exreserve
/lib/exrecover
/lib/accept
/lib/lpadmin
/lib/lpmove
/lib/lpsched
/lib/lpshut
/lib/reject
/lib/mailx/rmail
/lib/sa/sadc
/lib/uucp/uucico
/lib/uucp/usched
/lib/uucp/uuxqt
/rar/bin/sh
#
```

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

---

## Chapter 2: User Services

|                                     |      |
|-------------------------------------|------|
| Introduction                        | 2-1  |
| Login Administration                | 2-2  |
| Adding Users                        | 2-2  |
| Changing or Deleting Passwd Entries | 2-3  |
| Group IDs                           | 2-4  |
| The User's Environment              | 2-6  |
| Environment Variables               | 2-8  |
| <b>umask</b>                        | 2-9  |
| Default Shell and Restricted Shell  | 2-10 |
| User Communications Services        | 2-11 |
| Message of the Day                  | 2-11 |
| <b>news</b>                         | 2-11 |
| <b>write</b> to All Users           | 2-13 |
| <b>mail</b> and <b>mailx</b>        | 2-13 |
| Anticipating User Requests          | 2-15 |
| Trouble Reporting                   | 2-15 |





---

## Introduction

This chapter deals with providing a variety of services to the users of your 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.

---

# Login Administration

## 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 log in 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:gQQ3xsv05bWuM,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 <b>passwd9</b> , 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 "Password Aging" in Chapter 1, "System Security". |
| user id    | The user-ID number ( <b>uid</b> ) 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 ( <b>gid</b> ) number as to the <b>uid</b> , 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 <b>/usr</b> . The <b>modadduser</b> 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 <b>/bin/sh</b> . This field is most commonly used to invoke a special shell, such as <b>/bin/rsh</b> (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 1 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 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.

## Changing or Deleting Passwd 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.

## Login Administration

---

Ever so often a user will forget his or her password. When that happens (let's say to user **abc**), you can log in 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.

## 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 : jcp, 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. |

---

## 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 2-1.

```
# 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 2-1: A Default `/etc/profile`

---

Several interesting things are contained in the profile:

- Some environment variables are exported (see "Environment Variables" later in this chapter).
- A file named `/etc/motd` is `cat`-ted (see "Message of the Day" later in this chapter).



- If the user is not **root**, the names of news items are displayed (**news -n**; see **news** later in this chapter).
- If the user has mail (**mail -e**), a message about it is displayed (see **Mail** later in this chapter).

For information on the shell programming commands used in Figure 2-1, see **sh(1)** in the *User's Reference Manual*.

## 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 2-2.

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

Figure 2-2: Environment Array for a Typical User

---

The environment variables shown in Figure 2-2 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.

## **umask**

A system default controls the permissions mode of any files or directories created by a user. The 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**.

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

---

## User Communications Services

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

### 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`.

### 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 <b>news</b> command 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 <b>news</b> command is entered with the names of one or more items as arguments, only those items selected are printed.                                                                                    |
| read and delete   | After the <b>news</b> 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 <b>/usr/news</b> 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.

## 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.

## mail and mailx

The UNIX 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*.

---

## **Anticipating User Requests**

As the system administrator for your computer you can expect users to look to you to help solve any number of problems. In addition to the system log described in Chapter 3, 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.

## **Trouble Reporting**

Another technique that is strongly recommended is an organized way for users to report problems. Figure 2-3 shows a sample Trouble Report that can be used to record and keep track of system problems.



TROUBLE REPORT

Machine \_\_\_\_\_

Program running \_\_\_\_\_

Production or development \_\_\_\_\_

Type \_\_\_\_\_

Symptoms \_\_\_\_\_

Scope \_\_\_\_\_

Error Messages \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Person reporting \_\_\_\_\_ login \_\_\_\_\_

Location \_\_\_\_\_ Phone \_\_\_\_\_

Figure 2-3: Sample Trouble Report

---

---

## Chapter 3: Processor Operations

|                                           |      |
|-------------------------------------------|------|
| Introduction                              | 3-1  |
| General Operating Policy                  | 3-1  |
| Maintaining a System Log                  | 3-2  |
| Administrative Directories and Files      | 3-2  |
| Root Directories                          | 3-3  |
| Important System Files                    | 3-3  |
| <br>                                      |      |
| Operating Levels                          | 3-8  |
| General                                   | 3-8  |
| How <b>init</b> Controls the System State | 3-11 |
| A Look at Entering the Multiuser State    | 3-13 |
| Powering Up                               | 3-13 |
| Early Initialization                      | 3-15 |
| Preparing the Run Level Change            | 3-15 |
| A Look at the System Life Cycle           | 3-17 |
| Changing Run Levels                       | 3-17 |
| Run Level Directories                     | 3-19 |
| Going to Single-User Mode                 | 3-20 |
| Run Level 3                               | 3-21 |



---

## Introduction

This chapter deals with the day-to-day operations of your 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.

## General Operating Policy

Many administrative tasks require the system to be shut down to a run level other than the multiuser 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 multiuser 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 multiuser 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.

## Maintaining a System Log

In a multiuser environment it is strongly recommended that a complete set of records be maintained. A system log book can be a valuable tool when troubleshooting 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.

## 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 UNIX System V manual pages in the *Programmer's Reference Manual*.

## Root Directories

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

|                   |                                                                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>bck</b>        | Directory used to mount a backup file system for restoring files.                                                                         |
| <b>bin</b>        | Directory that contains public commands.                                                                                                  |
| <b>dev</b>        | Directory containing special files that define all of the devices on the system.                                                          |
| <b>etc</b>        | Directory that contains administrative programs and tables.                                                                               |
| <b>install</b>    | Directory used by the System Administration Menu package to mount utilities packages for installation and removal (/install file system). |
| <b>lib</b>        | Directory that contains public libraries.                                                                                                 |
| <b>lost+found</b> | Directory used by <b>fsck(1M)</b> to save disconnected files.                                                                             |
| <b>mnt</b>        | Directory used to temporarily mount file systems during the restore of the operating system from floppy disks.                            |
| <b>save</b>       | Directory used by the System Administration Menu package for saving data on diskettes.                                                    |
| <b>tmp</b>        | Directory used for temporary files.                                                                                                       |
| <b>usr</b>        | Directory used to mount the <b>/usr</b> file system. (See Chapter 5 for a description of this file system.)                               |

## Important System Files

The following files and directories are important in the administration of your computer.

|                       |                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------|
| <b>/etc/checklist</b> | File used to define a default list of file system devices to be checked by <b>/etc/fsck</b> . |
|-----------------------|-----------------------------------------------------------------------------------------------|

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

|                      |                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>/etc/rc2</b>      | File executed by <b>/etc/init</b> that executes shell scripts in <b>/etc/rc2.d</b> and <b>/etc/rc.d</b> on transitions to system run-level 2.                                                                                                                                                                                                                                               |
| <b>/etc/rc2.d</b>    | Directory containing files executed by <b>/etc/rc2</b> for transitions to system run-levels 2 and 3. Files in this directory are linked from files in the <b>/etc/init.d</b> directory and begin with either a <b>K</b> or an <b>S</b> . <b>K</b> indicates processes that should be stopped and <b>S</b> indicates processes that should be started when entering run-levels 2 or 3.       |
| <b>/etc/rc.d</b>     | Directory containing executable files that do the various functions needed to initialize the system to run-level 2; they are executed when <b>/etc/rc2</b> is run. (Files contained in this directory prior to UNIX System V Release 3.0 were moved to <b>/etc/rc2.d</b> . This directory is only maintained for compatibility.)                                                            |
| <b>/etc/rc3</b>      | File executed by <b>/etc/init</b> that executes shell scripts in <b>/etc/rc3.d</b> on transitions to system run-level 3 (Remote File Sharing state).                                                                                                                                                                                                                                        |
| <b>/etc/rc3.d</b>    | Directory containing files executed by <b>/etc/rc3</b> for transitions to system run-level 3 (Remote File Sharing mode). Files in this directory are linked from the <b>/etc/init.d</b> directory and begin with either a <b>K</b> or an <b>S</b> . <b>K</b> indicates processes that should be stopped, and <b>S</b> indicates processes that should be started when entering run-level 3. |
| <b>/etc/rstab</b>    | 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).                                                                                                                                                                                                    |
| <b>/etc/save.d</b>   | Directory containing files that are used by the System Administration Menu commands associated with backing-up data on floppy diskettes.                                                                                                                                                                                                                                                    |
| <b>/etc/shutdown</b> | File containing a shell script that gracefully shuts down the system in preparation for system backup or for scheduled downtime.                                                                                                                                                                                                                                                            |



|                                 |                                                                                                                                                                                                                                                                                                                     |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>/etc/shutdown.d</b>          | 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 UNIX System V Release 3.0 were moved to <b>/etc/rc0.d</b> . This directory is only maintained for compatibility.) |
| <b>/etc/TIMEZONE</b>            | File used to set the time zone shell variable TZ.                                                                                                                                                                                                                                                                   |
| <b>/etc/utmp</b>                | File containing the information on the current run-state of the system.                                                                                                                                                                                                                                             |
| <b>/etc/wtmp</b>                | File containing a history of system logins.                                                                                                                                                                                                                                                                         |
| <b>/usr/adm/sulog</b>           | File containing a history of <b>su</b> command usage. This file should be checked periodically for size.                                                                                                                                                                                                            |
| <b>/usr/lib/cron/log</b>        | File containing a history of all the actions taken by <b>/etc/cron</b> . This file should be checked periodically for size.                                                                                                                                                                                         |
| <b>/usr/lib/help/HELPLIST</b>   | File containing a history of all the actions taken by the <b>/usr/bin/help</b> (if it is enabled on the system).                                                                                                                                                                                                    |
| <b>/usr/lib/spell/spellhist</b> | File containing a history of all words that <b>spell</b> fails to match (if the Spell Utilities are installed on the system).                                                                                                                                                                                       |
| <b>/usr/news</b>                | Directory containing news files. This directory should be checked periodically, and old files should be discarded.                                                                                                                                                                                                  |
| <b>/usr/options</b>             | 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.

---

# Operating Levels

## General

After you have set up your computer for the first time (plugging it in, hooking all the hardware together, installing the system software, booting it, running the setup programs) as documented, you and other users can use the system. Whenever you turn it on (including the first time), the system comes up in a multiuser 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 multiuser 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 multiuser state.

Not all activities, however, can be performed in the multiuser 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 Figure 3-1), 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. Figure 3-1 defines each of them as they pertain to the computer.

## Operating Levels

---

| 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 <b>s</b> and <b>1</b> are both used to go to single-user state, <b>s</b> only kills processes spawned by <b>init</b> and does not unmount file systems. State <b>1</b> unmounts everything except root and kills all user processes, except those that relate to the console. |
| 2          | Multuser mode is the normal operating mode for the system. The default is that the <b>root (/)</b> and user ( <b>/usr</b> ) file systems are mounted in this mode. When the system is powered up, it is put in multuser mode.                                                                                                                                                                                    |
| 3          | Multuser/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 10, "Remote File Sharing".)                                                                                                                                                                                  |
| 4          | User-defined run state.                                                                                                                                                                                                                                                                                                                                                                                          |

Figure 3-1: System States

---

## How **init** Controls the System State

UNIX 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 multiuser 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 3-2.

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



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.

```
bchk::bootwait:/etc/bcheckrc </dev/console >/dev/console 2>&1
brc::bootwait:/etc/brc 1> /dev/console 2>&1
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
is:2:initdefault:
r0:0:wait:/etc/rc0 1> /dev/console 2>&1 </dev/console
r1:1:wait:/etc/rc1 1> /dev/console 2>&1 </dev/console
r2:23:wait:/etc/rc2 1> /dev/console 2>&1 </dev/console
r3:3:wait:/etc/rc3 1> /dev/console 2>&1 </dev/console
co:12345:respawn:/etc/getty console console
00:23:off:/etc/getty tty00 9600
01:23:off:/etc/getty tty01 9600
i0:23:off:/etc/getty ttyi0 9600 # iSBC 546 Terminal Controller
i1:23:off:/etc/getty ttyi1 9600 # iSBC 546 Terminal Controller
i2:23:off:/etc/getty ttyi2 9600 # iSBC 546 Terminal Controller
i3:23:off:/etc/getty ttyi3 9600 # iSBC 546 Terminal Controller
```

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 multiuser 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.

|                    |                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------|
| <b>wait</b>        | when going to <i>level</i> , start <i>process</i> and wait until it's finished.                                       |
| <b>initdefault</b> | when <b>init</b> starts, it will enter <i>level</i> ; the <i>process</i> field for this <i>action</i> has no meaning. |
| <b>once</b>        | run <i>process</i> once and don't start it again if it finishes.                                                      |
| <b>powerfail</b>   | tells <b>init</b> to run <i>process</i> whenever a direct powerdown of the computer is requested.                     |
| <b>respawn</b>     | if process does not exist, start it, wait for it to finish, and then start another.                                   |
| <b>ondemand</b>    | synonymous with <b>respawn</b> , but used only with <i>level a, b, or c</i> .                                         |
| <b>off</b>         | when in <i>level</i> , 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**.

## A Look at Entering the Multiuser State

### Powering Up

When you power up your system, it will enter multiuser state by default. (You can change the default by modifying the **initdefault** line in your **inittab** file.) In effect, going to the multiuser state follows these broad lines (see Figure 3-2):

1. You turn on the computer.
2. The operating system is loaded and the early system initializations are started by **init**.



## Operating Levels

---

3. The run level change is prepared by the `/etc/rc2` procedure.
4. Finally the system is made public via the spawning of `gettys` along the terminal lines.

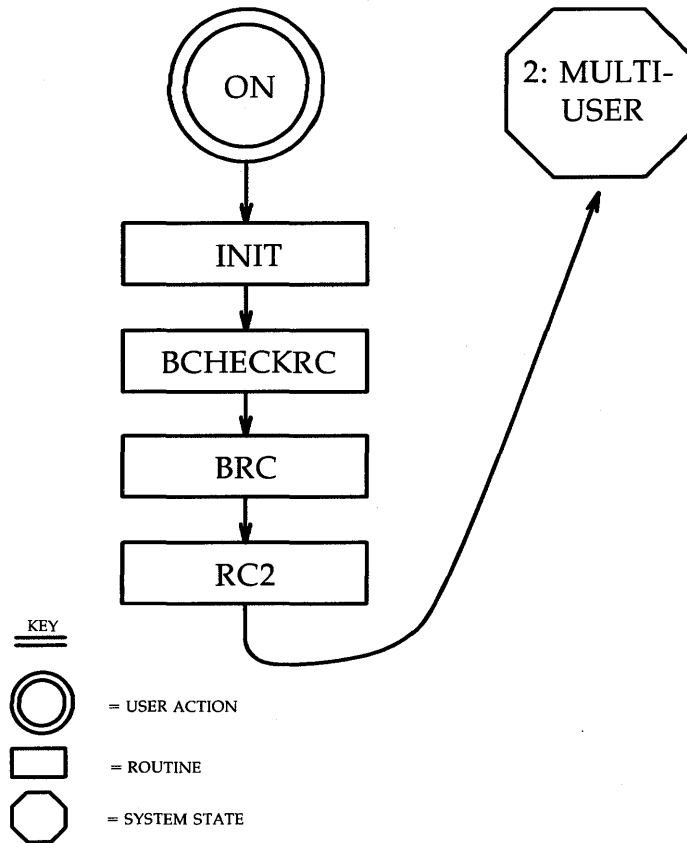


Figure 3-2: A Look at System Initialization

---

## Early Initialization

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

```
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
```

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 multiuser state.

## 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 multiuser state) as the level to select and execute other entries:

## Operating Levels

---

```
is:2:initdefault:
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
ct:234:off:/etc/getty contty 9600
he:234:respawn:sh -c 'sleep 20 ; exec /etc/hdlogger >/dev/console 2>&1'
11:234:respawn:/etc/getty -t60 tty11 1200
12:234:respawn:/etc/getty -t60 tty12 1200
13:234:respawn:/etc/getty -t60 tty13 1200
```

The other entries shown above specify the actions necessary to prepare the system to change to the multiuser 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 multiuser environment is established, and your system is available for users to log in (see Procedures 3.1 and 3.4).

## A Look at the System Life Cycle

### Changing Run Levels

In effect, changing run levels follows these broad lines (see Figure 3-3):

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.

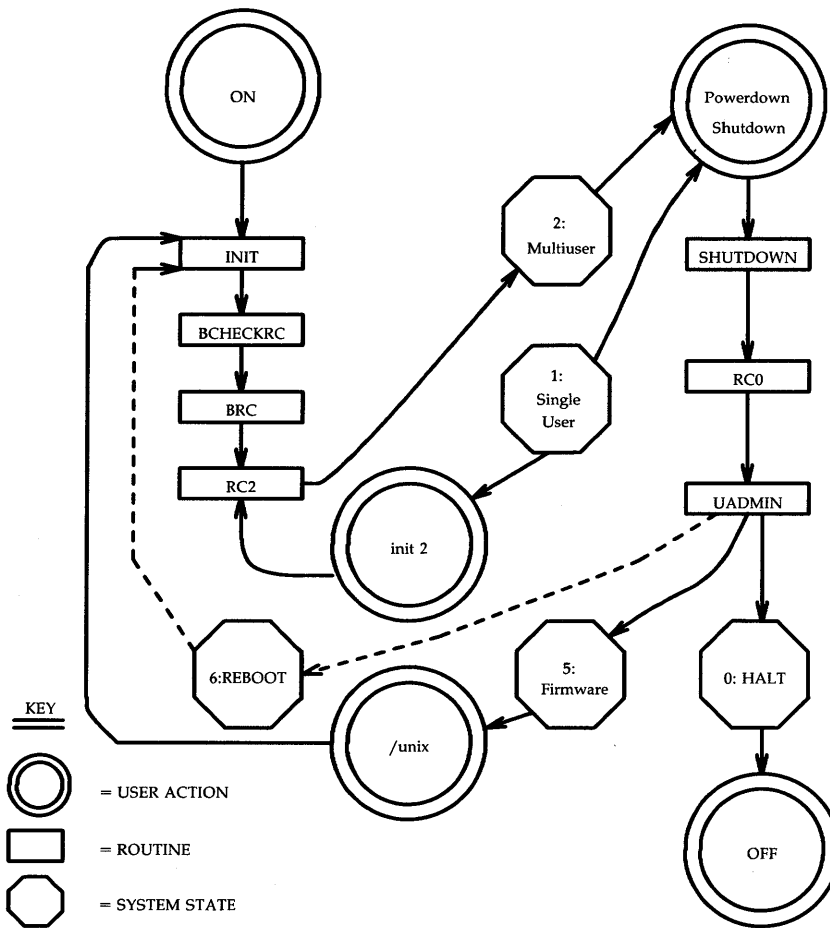


Figure 3-3: A Look at the System Life Cycle

---

## 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 file names look like this:

*S00name*

or

*K00name*

The file names can be split into three parts:

|                      |                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>S</b> or <b>K</b> | The first letter defines whether the process should be started ( <b>S</b> ) or stopped ( <b>K</b> ) upon entering the new run level.                                   |
| <i>00</i>            | 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.). |
| <i>name</i>          | The rest of the file name is the <b>/etc/init.d</b> file name 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.

## 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/rc.d** directory by calling the **/etc/rc1** 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 entry for single-user processing in the sample **/etc/inittab** is:

```
r1:1:wait:/etc/rc1 1>/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 **r1** entry, and it starts the shutdown processing.

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

### 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 multiuser 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 10.





---

## Chapter 4: Disk/Tape Management

|                                             |      |
|---------------------------------------------|------|
| Introduction                                | 4-1  |
| Device Types                                | 4-2  |
| Hard Disk Devices                           | 4-2  |
| Diskette Drives                             | 4-2  |
| Cartridge Tape Drive                        | 4-3  |
| Identifying Devices to the Operating System | 4-4  |
| Block and Character Devices                 | 4-6  |
| Defining a New Special File                 | 4-7  |
| Formatting and Partitioning                 | 4-8  |
| Formatting Disks                            | 4-8  |
| Hard Disk Partitioning                      | 4-9  |
| Planning to Change Hard Disk Partitions     | 4-9  |
| Changing Partitions to Increase Swap Space  | 4-10 |
| Other Disk Operations                       | 4-11 |
| Duplicating Disks                           | 4-11 |
| Verifying Usability                         | 4-11 |
| The Bad Block Handling Feature              | 4-13 |
| When Is a Block Bad?                        | 4-13 |
| When Are Bad Blocks Detected?               | 4-15 |
| How Bad Block Handling Works                | 4-16 |
| A Bad Block Handling Scenario               | 4-16 |
| Replacing New Bad Blocks                    | 4-17 |
| Dealing With Data Loss                      | 4-18 |



---

## Introduction

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

- Disk device types and sizes
- Tape device sizes
- Making devices known to the operating system
- Formatting disks
- Duplicating disks
- Verifying disk or tape 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 5, "File System Administration".

---

## Device Types

Your 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 computer permit you to have two or more hard disk units. A cartridge tape drive is available on some models.

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. The cartridge tape device is used primarily for high-speed file system backups.

## Hard Disk Devices

Hard disks come in several sizes. The file `/etc/partitions` contains partitioning information for some of these. The partitioning information for your disk can be found after the line which reads `disk0:`.

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 your 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 computer are housed within the cabinet.

## Diskette Drives

Your computer comes with one integral diskette drive. An Expansion Module that adds another diskette drive is available as an option. The drive accommodates 5-1/4 inch flexible diskettes. (You sometimes hear them called "floppy disks.") A diskette holds 360 kilobytes of data on a Multibus Machine, and 1.2 megabytes of data on an AT386 Machine. 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(s) 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.

## **Cartridge Tape Drive**

Your computer may be equipped with a cartridge tape drive. Cartridge tapes provide a more efficient way of storing data files and file systems. Cartridge tape sizes vary (approximately 10 MB per 100 feet).

Because you can dump a lot of data to a tape fairly quickly, having a cartridge tape device may influence the way you do file system backups. Complete backups may work better for you than the combination of complete and incremental backups. (See Chapter 5, "File System Administration", for a discussion of backups.)

---

## Identifying Devices to the Operating System

Before a disk or tape device can be used on a computer running the UNIX 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 4-1 shows excerpts from the output of commands on a user's directory and the `/dev` directory structure.

```

(a regular file)
-rw-r----- 1 abc      dsg      1050 Nov 23 08:14 dm.ol

(A directory)
drw-r----- 1 dsg      dsg      50, Nov 20 09:10 dr.1

(Hard disk device files)
brw----- 1 root      root      17, 0 Nov 23 15:07 /dev/dsk/0s0
brw----- 2 root      root      17, 1 Nov 23 15:07 /dev/dsk/0s1

crw----- 1 root      root      17, 0 Nov 23 15:07 /dev/rdisk/0s0
crw----- 2 root      root      17, 1 Nov 23 15:07 /dev/rdisk/0s1

(Dloppy diskette files)
brw-rw-rw- 1 root      root      0, 34 Nov 23 15:08 /dev/dsk/f0d16t
brw-rw-rw- 2 root      root      0, 33 Nov 23 15:08 /dev/dsk/f0d4d
brw-rw-rw- 1 root      root      0, 32 Nov 23 15:08 /dev/dsk/f0d8d
brw-rw-rw- 1 root      root      0, 35 Nov 23 15:08 /dev/dsk/f0d8dt
brw-rw-rw- 1 root      root      0, 36 Nov 23 15:08 /dev/dsk/f0d9dt
brw-rw-rw- 2 root      root      0, 33 Nov 23 15:08 /dev/sa/diskette

cwr-rw-rw- 1 root      root      0, 34 Nov 23 15:08 /dev/rdisk/f0d16t
cwr-rw-rw- 2 root      root      0, 33 Nov 23 15:08 /dev/rdisk/f0d4d
cwr-rw-rw- 1 root      root      0, 32 Nov 23 15:08 /dev/rdisk/f0d8d
cwr-rw-rw- 1 root      root      0, 35 Nov 23 15:08 /dev/rdisk/f0d8t
cwr-rw-rw- 1 root      root      0, 36 Nov 23 15:08 /dev/rdisk/f0d9dt
cwr-rw-rw- 2 root      root      0, 33 Nov 23 15:08 /dev/rSA/diskette

(cartridge tape device files)
crw-rw-rw- 2 root      root      3, 48 Nov 23 15:08 /dev/zmt0
crw-rw-rw- 1 root      root      3, 49 Nov 23 15:08 /dev/zmnt0

```

Figure 4-1: Directory Listing Extracts: Regular and Device Files



## Identifying Devices

---

The extracts from directory listings in Figure 4-1 show a regular file [indicated by the dash (-) in the first position of the line] with these characteristics:

- It has 1050 characters.
- The file name 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).

Notice that such pairs of files have different file names or are in different directories (for example, **/dev/mt/0m** and **/dev/rmt/0m**).

- There are alias names, for example, file **diskette1** is an alias for **f0d4d**.
- 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.)

## 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 UNIX 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 4-1, where the character device version of the diskette drive is in directory `/dev/rdisk`. The character-at-a-time method is used by some file maintenance utilities.

## Defining a New Special File

The need to define new special device files occurs infrequently. When the need does occur, however, there is a UNIX 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*.)

---

## Formatting and Partitioning

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

### Formatting Disks

Before a disk or diskette can be used for 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 both disks and diskettes, formatting maps both sides of the disk into tracks and sectors that can be addressed by the disk controller. On hard disks, a portion of the disk is reserved for data having to do with the specific disk. The volume table of contents (VTOC) shows how the partitions on the disk are allocated. An additional use of the reserved area is to map portions of the disk that may not be usable. Formatting a previously used disk or diskette, 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. If the box is open, it means all diskettes in it are formatted. If you tend to use several different diskette formats, you should label the box with the device name the diskette was formatted on.

Diskettes are formatted by the UNIX system **format(1M)** command or the **sysadm format(1)** subcommand.

The **mkpart(1M)** command is used to format and maintain the VTOC information on hard disks. This is done automatically for you during the installation process for the primary hard disk. If you wish to add an additional hard disk, you can use the UNIX system **adddisk(1M)** command.

## Hard Disk Partitioning

The UNIX system **mkpart(1M)** procedure establishes hard disk partitions according to information in a partitions file. The default, located in */etc/partitions*, 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. For more information about how to change your partitions, read the *mkpart(1M)* manual page (in the System Administrator's Reference Manual), which gives information on changing partitions using **mkpart**.

The file */etc/partitions* contains information for hard disk partitioning. The default information for your hard disk can be found after the line which reads "disk0:."

## 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*, if present) for which the default is all the available space. 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 others?
- Have we added, or are we planning to add, system software that changes our thinking about space requirements?

If you decide that a re-partitioning is needed, it must be done with a full system backup and restore. The **adddisk(1M)** command can be used to re-define partitions on disks other than the root disk if they are first backed up and then restored.

## 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).

Additional swap space is added to the primary disk's swap space (usually without affecting any secondary disk's data) by a full backup and restore procedure. Extra swap area can be allocated on a secondary disk (instead of your primary disk) using **adddisk(1M)**. If this procedure is used, you must inform the UNIX system kernel of the existence of the new swap space using the **swap(1M)** command. Before you run either procedure, there are three things that you should do first:

1. Find out about your present partitions using the **mkpart(1M)** command, or use the sysadm hard disk 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 that you have to reduce the size of one of the other partitions.)
3. Do a *complete* backup (see procedure 5.4) of the disk you intend to change. (The process of changing partitions is going to erase everything that is already there.)

---

## Other Disk Operations

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

### 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 multiuser mode (run-level 2). No matter which directory is specified, a minimum of 2500 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.

### Verifying Usability

An option, **-E**, of the **format(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

## Other Disk Operations

---

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.

---

## The Bad Block Handling Feature



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

The 80386 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 blocks that are no longer usable
- Restoring the usability of the disk in spite of the bad blocks that exist

It should be pointed out that new bad blocks are seldom found, particularly with sealed disks, 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 that 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. Regular system backups provide protection against operational errors as well as protection against lost data due to bad blocks. A discussion of backup procedures can be found in Chapter 5, "File System Administration".

### 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 a disk, 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 a service representative. Several distinct failures occurring about the same time should also prompt you to contact your service representative to check them out.

### **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.

### **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 UNIX 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.

### **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.

### **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.

### **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.

### How Bad Block Handling Works

The bad block handling feature provides the mechanisms to detect and add new bad blocks to the existing map. This procedure is not totally automated, and requires that the system operators keep an exact log of hard disk error messages as they are printed on the system console.



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

- An absolute sector number is an integer.
- A drive number is an integer number denoting a particular drive on a disk controller board.
- A controller number is an integer number denoting a particular controller board on an 80386 machine.

All bad block error messages appearing on the console will contain these three numbers.

### A Bad Block Handling Scenario

Sometime during normal operations, an error message appeared on the console listing an absolute sector number, a drive number, and a controller number. This message was carefully logged by the operator. Eventually, the same message shows up several times in the log and the System Administrator decides the sector is bad (after exhausting the other possibilities mentioned earlier in this chapter).

The absolute sector number was 14000, the drive was 0, and the controller was 0.

## Replacing New Bad Blocks

In order to add this bad block to the existing mapping, the System Administrator uses the **sysadm addbadblocks** subcommand.

He goes through the following procedure:

```
sysadm addbadblocks
```

```
Running subcommand addbadblocks from menu harddiskmgmt
```

```
Any time you want to quit, type "q".
```

```
If you are not sure how to answer a prompt, type "?" for help,  
or see the appropriate manual.
```

```
If a default appears in the question, press <RETURN> for the default.
```

```
enter the drive number: 0
```

```
enter the controller number: 0
```

```
Enter absolute sector number [?,q]: 14000
```

```
This is the information for the bad area:
```

```
Drive # :          0  
Controller # :     0  
Absolute Sector # : 14000  
Disk Name:         disk0
```

```
Do you want to install or skip this entry [i, s, q]? i
```

```
Do you wish to add another bad block? [y, n, q]? n
```

## Disk Identification

These mechanisms support both single-disk and multi-disk models of your 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.

### 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. Under rare circumstances, you might also need to have your disk reformatted if you lose the Volume Table of Contents (VTOC) block. You may need to restore the special code (the program) for booting the system (it is not in a file system).

---

## Chapter 5: File System Administration

|                                                                 |      |
|-----------------------------------------------------------------|------|
| Introduction                                                    | 5-1  |
| How the File System is Organized                                | 5-1  |
| Block 0                                                         | 5-4  |
| Block 1: the Super Block                                        | 5-4  |
| I-Nodes                                                         | 5-5  |
| Storage Blocks                                                  | 5-7  |
| Free Blocks                                                     | 5-8  |
| Summary                                                         | 5-8  |
| <br>                                                            |      |
| The Relationship Between the File System and the Storage Device | 5-9  |
| Disk Format                                                     | 5-9  |
| Partitions                                                      | 5-10 |
| Size Limitations                                                | 5-11 |
| <br>                                                            |      |
| How the File System Works                                       | 5-12 |
| Tables in Memory                                                | 5-12 |
| The System I-Node Table                                         | 5-12 |
| The System File Table                                           | 5-13 |
| The Open File Table                                             | 5-14 |
| System Steps in Accessing a File                                | 5-15 |
| Open                                                            | 5-16 |
| Create                                                          | 5-17 |
| Reading and Writing                                             | 5-17 |
| Files Used by More Than One Process                             | 5-18 |
| Path Name Conversion                                            | 5-18 |
| Synchronization                                                 | 5-18 |
| Search Time                                                     | 5-19 |
| Holes in Files                                                  | 5-19 |
| Summary                                                         | 5-20 |

|                                                       |      |
|-------------------------------------------------------|------|
| <b>Administering the File System</b>                  | 5-21 |
| Creating a File System and Making it Available        | 5-21 |
| Using <b>mkfs</b>                                     | 5-21 |
| Relating the File System Device to a File System Name | 5-22 |
| Mounting and Unmounting File Systems                  | 5-23 |
| Summary                                               | 5-25 |
| <br>                                                  |      |
| <b>Maintaining a File System</b>                      | 5-26 |
| The Need for Policies                                 | 5-26 |
| Shell Scripts for File System Administration          | 5-27 |
| Checking for File System Consistency                  | 5-27 |
| Monitoring Disk Usage                                 | 5-27 |
| Monitoring Percent of Disk Space Used                 | 5-28 |
| Monitoring Files and Directories that Grow            | 5-28 |
| Identifying and Removing Inactive Files               | 5-29 |
| Identifying Large Space Users                         | 5-30 |
| File System Backup and Restore                        | 5-31 |
| Complete Backup                                       | 5-32 |
| Incremental Backup                                    | 5-33 |
| Selective Backup                                      | 5-35 |
| Backup Schedule Reminder                              | 5-35 |
| Restoring a File System from Backup                   | 5-36 |
| <br>                                                  |      |
| <b>What Can Go Wrong With a File System</b>           | 5-37 |
| Hardware Failure                                      | 5-37 |
| Program Interrupts                                    | 5-37 |
| Human Error                                           | 5-37 |

|                                                   |      |
|---------------------------------------------------|------|
| <b>How to Check a File System for Consistency</b> | 5-39 |
| The <b>fsck</b> Utility                           | 5-39 |
| The <b>fsck</b> Command                           | 5-40 |
| Sample Command Use                                | 5-42 |
| File System Components Checked by <b>fsck</b>     | 5-42 |
| Super Block                                       | 5-43 |
| I-Nodes                                           | 5-44 |
| Indirect Blocks                                   | 5-47 |
| Directory Data Blocks                             | 5-47 |
| Regular Data Blocks                               | 5-48 |
| Running <b>fsck</b>                               | 5-48 |
| Initialization Phase                              | 5-49 |
| General Errors                                    | 5-50 |
| Meaning of Yes/No Responses                       | 5-50 |
| Phase 1: Check Blocks and Sizes                   | 5-51 |
| Phase 1B: Rescan for More DUPS                    | 5-53 |
| Phase 2: Check Path Names                         | 5-53 |
| Phase 3: Check Connectivity                       | 5-56 |
| Phase 4: Check Reference Counts                   | 5-57 |
| Phase 5: Check Free List                          | 5-61 |
| Phase 6: Salvage Free List                        | 5-63 |
| Cleanup Phase                                     | 5-63 |





---

## Introduction

### How the File System is Organized

A primary function of the UNIX operating system is to support file systems. In the UNIX 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 5-1 shows the relationship between directories and files in a UNIX file system. The circles represent directories.

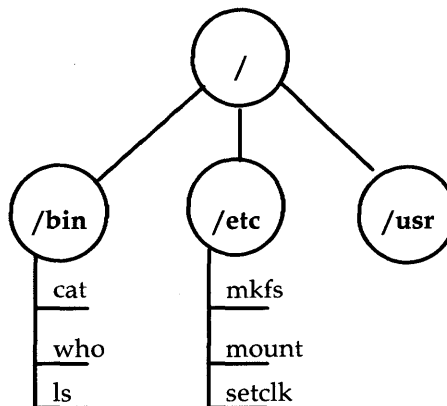


Figure 5-1: A UNIX System File System

---

The starting point of any UNIX file system is a directory that serves as the root. In the UNIX 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 5-1, 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 5-2.

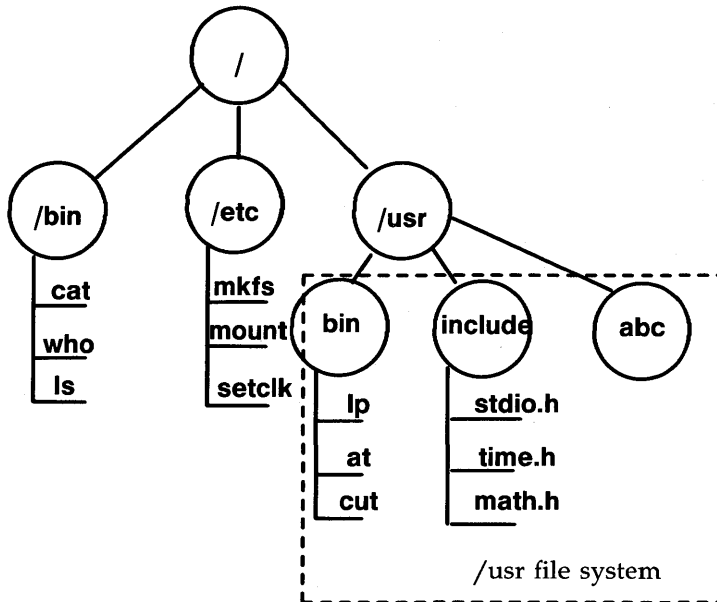


Figure 5-2: Adding the `/usr` File System

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

The diagrams in Figures 5-1 and 5-2 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 UNIX 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 5-3.

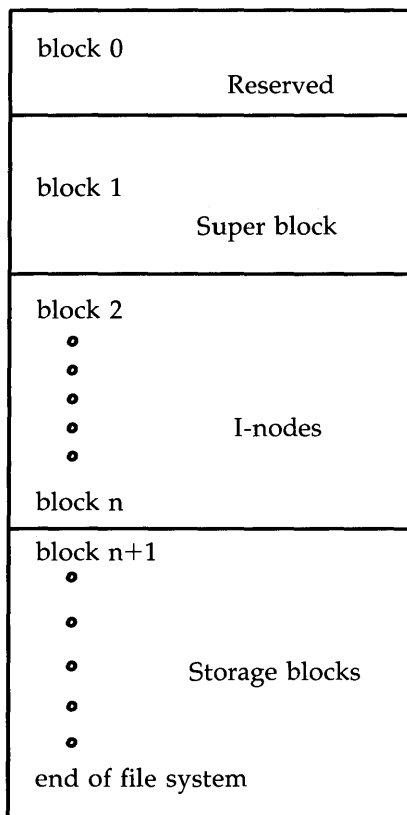


Figure 5-3: The UNIX System View of a File System

---

## 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.

## 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 5-4. 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.

|                                                                                                                                                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Miscellaneous Information</p> <ul style="list-style-type: none"><li>- logical/physical disk sizes</li><li>- super block modified flag</li><li>- read-only system flag</li><li>- current date of last update</li><li>- label or name</li></ul> |
| <p>I-node Information</p> <ul style="list-style-type: none"><li>- total number of i-nodes</li><li>- total free i-nodes</li><li>- array of free i-node numbers</li><li>- index into inumber array</li></ul>                                       |
| <p>Block Information</p> <ul style="list-style-type: none"><li>- total number of free blocks</li><li>- array of free block numbers</li><li>- index into array of free block numbers</li></ul>                                                    |

Figure 5-4: The Super Block

---

## 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.

## Introduction

---

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
- 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 UNIX system file is well beyond the limits of the amount of disk storage on your computer. Figure 5-5 illustrates this chaining of address blocks stemming from the i-node.

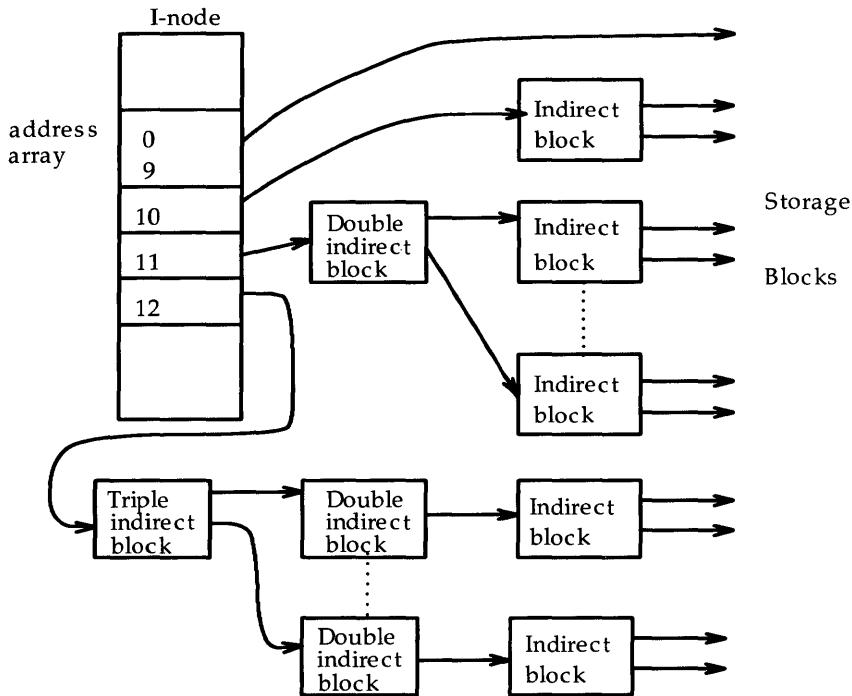


Figure 5-5: The File System Address Chain

## 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 file name of the member file or subdirectory.



## 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.

## Summary

What we have described thus far is an abstract view of a UNIX 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.

---

## The Relationship Between the File System and the Storage Device

In the UNIX 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 4, "Disk/Tape Management".

### Disk Format

Before a disk can be used by the UNIX system it must be formatted into addressable sectors.

A disk sector is a portion of the storage medium that can be addressed by the disk controller. On the AT386, that portion is 512 bytes, and on the Multibus machine the portion is 1024 bytes.

**NOTE** Hard disk units on your 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:

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

The **mkpart(1M)** command is used to define a volume table of contents (VTOC) for a formatted hard disk (see "Partitions" which follows).

## Partitions

The next level in disk formatting is partitions. On the 80386 computers, up to 16 partitions can be defined on a hard disk device; up to 5 partitions on a diskette. On a hard disk, the **mkpart(1M)** command is used to tie the starting points of partitions to sector numbers (sectors are numbered from 0 to however many the disk holds). 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. Some partition tags are reserved. The list below shows tags that are reserved on the 80386 computer.

| Name             | Number |
|------------------|--------|
| ENTIRE DEVICE    | 0      |
| ROOT             | 1      |
| SWAP             | 2      |
| USR              | 3      |
| USR2             | 4      |
| DOS              | 5      |
| BOOT             | 6      |
| ALTERNATE SECTOR | 7      |
| DIAGNOSTICS      | 8      |
| AVAILABLE        | 9-15   |

When you first get your 80386 computer, the primary hard disk device has already been formatted, and the partitions assigned are something like this: (the example shown is for a 40MB Quantum disk drive)

TYPE "mkpart -t -p disk0" TO GET THE INFORMATION YOU NEED TO FILL IN THE CHART ON PAGE 5-11

REMEMBER:

`/dev/[r]dsk/0sn` REFERS TO DRIVE 0 PARTITION *n*.

`/dev/[r]dsk/1sn` REFERS TO DRIVE 1 PARTITION *n*.

FOR THE SECOND DISK CONTROLLER, PREPEND "c1d" TO THE DRIVE/PARTITION NAME (i.e., `/dev/dsk/c1d0s1` IS CONTROLLER #1, DRIVE 0, PARTITION 1)

| 40-Megabyte Hard Disk Drive<br>(blocks per cylinder = 72)<br>(rotational gap = 2) |            |              |       |         |
|-----------------------------------------------------------------------------------|------------|--------------|-------|---------|
| Disk Partition                                                                    | Use        | Sector Start | Size* | I-Nodes |
| 0s0                                                                               | DISK       | 0            | 36864 | —       |
| 0s1                                                                               | ROOT       | 4464         | 32184 | 8032    |
| 0s2                                                                               | SWAP       | 144          | 4320  | —       |
| 0s3                                                                               | unassigned | —            | —     | —       |
| 0s4                                                                               | unassigned | —            | —     | —       |
| 0s5                                                                               | unassigned | —            | —     | —       |
| 0s6                                                                               | BOOT       | 0            | 18    | —       |
| 0s7                                                                               | ALTERNATES | 18           | 62    | —       |
| 0s8                                                                               | NONUNIX    | 36648        | 216   | —       |

\* Size is 1024-byte blocks on the Multibus 1 and 512-byte blocks on the AT386.

Figure 5-6: Disk Partitions, 40-Megabyte Drive

---

The illustration shows two file systems defined on this drive: the **root** file system and one called **/usr**. Tables showing the default partitioning for all supported devices are in Appendix A.

## 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, on an AT386 machine, is 512 bytes. The size of a block on a Multibus I machine is 1024 bytes. On both computers 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.

---

## How the File System Works

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

### Tables in Memory

When a file system is identified to the UNIX 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.

### The System I-Node Table

The UNIX 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 5-7.

|                             |
|-----------------------------|
| I-node chain pointers       |
| Free-list chain             |
| Flag                        |
| Waiting count for i-node    |
| Reference count             |
| Device where i-node resides |
| I-number                    |
| Mode                        |
| Number of links             |
| User-ID of owner            |
| Group-ID of owner           |
| Size of file                |

Figure 5-7: The System I-Node Table

---

### 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:

## How the File System Works

---

- 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

## 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 5-8 shows how these tables point to each other.

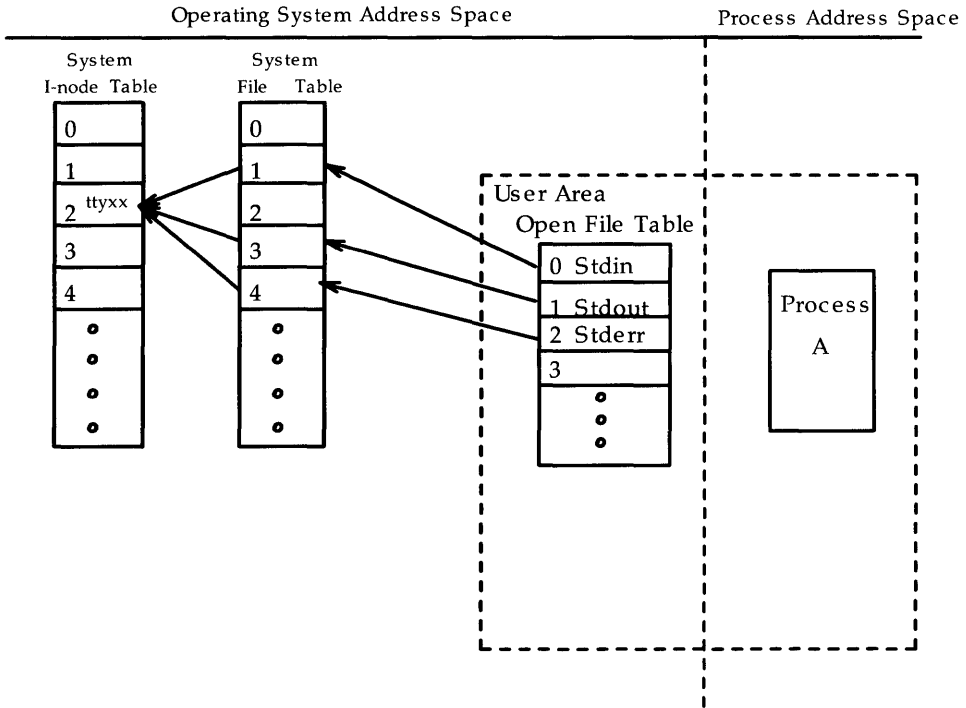


Figure 5-8: File System Tables and Their Pointers

---

## 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.



### Open

Suppose we give the path name `/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 path name 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.]

## 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 path name into the directory that is the next to last portion of the path name. The i-number is stored with it.

## Reading and Writing

Both the **read(2)** and **write(2)** system calls 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.

### 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.

### Path Name Conversion

The directory search and path name conversion take 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.

## Synchronization

The above description, while complex, may seem rather neat and orderly. The situation is complicated, however, by the fact that the UNIX system is a multitasking 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.

## 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 UNIX system is locating a file to be opened, it searches linearly through all the directories in the path name. 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 its 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.

## 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.

## **Summary**

We have tried in this section to give you an understanding of how the UNIX 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.

---

# Administering the File System

## 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.

### Using **mkfs**

The **mkfs** command has two formats:

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

```
mkfs special_file prototype [gap blocks/cyl]
```

Notice that in neither format is the file system actually given a name; it is identified by the file name 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. Figure 5-9 shows the recommended values for use with the **mkfs** command for devices supported on the 80386 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.

| Device         | Physical Blks/Cyl |        | Gap Size |
|----------------|-------------------|--------|----------|
|                | MB1               | AT/386 |          |
| 40M Quantum    | 72                | 136    | 2        |
| 75M Toshiba    | 90                | 170    | 2        |
| 140M Maxtor    | 135               | 255    | 2        |
| 20M Miniscribe | 36                | 68     | 2        |
| 40M Seagate    | 40                | 85     | 2        |
| 40M Micropolis | 54                | 102    | 2        |

Figure 5-9: Interrecord Gap and Blocks/Cylinder Recommendations

---

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 *System Administrator's Reference Manual*. Note that the **sysadm makefsys** subcommand has no provision for the use of prototype files.

## Relating the File System Device to a File System Name

A UNIX file system is generally referred to by the name of the highest level directory in its hierarchy. The file system shown in Figure 5-2 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 UNIX 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.

## Mounting and Unmounting File Systems

For a file system to be available to users, the UNIX 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 multiuser 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 UNIX system table (called the mount table, or **/etc/mnttab**) paired with the directory that is specified. For example, the command

```
# mount /dev/dsk/0s3 /usr
```

tells the system that **/dev/dsk/0s3** contains a file system that begins at directory **/usr**.



The **mount** command has other arguments. See the *System 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 UNIX 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 760 512-byte blocks on a Multibus I, and 2400 512-byte blocks on an AT386. 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, and 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.

## 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.

---

## 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 your 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

## The Need for Policies

As with most other aspects of administering a 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.

## 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.

## 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.

## 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 UNIX system commands that you may use.

## 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.

## 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                                       |
|---------------------------------|-------------------------------------------|
| <b>/etc/wtmp</b>                | history of system logins                  |
| <b>/usr/adm/sulog</b>           | history of <b>su</b> commands             |
| <b>/usr/lib/cron/log</b>        | history of actions of <b>/etc/cron</b>    |
| <b>/usr/lib/help/HELPLOG</b>    | actions of <b>/usr/bin/help</b>           |
| <b>/usr/lib/spell/spellhist</b> | words that <b>spell(1)</b> 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.

## 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 file names 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:

|                                  |                                                                                                                                                                                                         |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>/usr</b>                      | specifies the path name where <b>find</b> is to start. Presumably, your machine is organized in such a way that inactive user files will not often be found in the <b>root</b> file system.             |
| <b>-type</b>                     | tells <b>find</b> to look only for regular files and to ignore special files, directories, and pipes.                                                                                                   |
| <b>-mtime +60</b>                | says you are interested only in files that have not been modified in 60 days.                                                                                                                           |
| <b>-print</b>                    | means that when a file is found that matches the <b>-type</b> and <b>-mtime</b> expressions, you want the path name to be printed.                                                                      |
| <b>&gt; /tmp/deadfiles &amp;</b> | 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).

## 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 path names 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).

## 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). The directories and files are read back to the hard disk by using **sysadm restore**(1) or the appropriate **cpio** command.



Another method of backing up information stored on your computer is to copy file systems to another computer system over a high-speed data link. The link between the machines must be a high-speed data link so that data transfers are done in a timely fashion. The other machine should be a larger system with mass storage capability.

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 under "Backup Schedule Reminder" later in this section.

## 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.

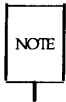
If the backup medium is diskettes, you must have enough formatted diskettes on-hand to contain the directories and files being backed up. As a rule-of-thumb, the number of diskettes needed for a complete backup is equal to the number of data blocks used in the file system divided by 1300, rounded to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

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.

## 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.



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 5-10.

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.

```
# 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 5-10: Sample `/etc/save.d/except` File

---

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.

## 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 1422 blocks (512 bytes per block). Before you copy the selected directory(ies) or file(s), check the number of blocks involved.

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

## 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)**.

## 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 UNIX 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 the discussion on "Selective Backup").

---

## 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.

### 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 maintenance procedures are followed conscientiously.

### Program Interrupts

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

### 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.

## What Can Go Wrong

---

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.

---

## How to Check a File System for Consistency

When the UNIX operating system is brought up, a consistency check of the file systems should always be done. On your 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.

### 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.



## 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][-sX][-SX][-tfile][-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.
- sX** 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         | <i>-sblocks/cylinder:gap</i> (MB1) |
|----------------|------------------------------------|
| 140M Hard Disk | <b>-s135:2</b>                     |
| 75M Hard Disk  | <b>-s90:2</b>                      |
| 40M Hard Disk  | <b>-s72:2</b>                      |
| Diskette       | <b>-s18:4</b>                      |

- SX** 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.
- tfile** 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 file name 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**.

### 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/0s1

/dev/dsk/0s1
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
#
```

### 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 UNIX system file system and to describe the kinds of consistency checks that are applied to them.

## 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

### 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.

### 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.

## Checking for Consistency

---

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.

### 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.

### 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.

## 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

### 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.

#### **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.

#### **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.

## Checking for Consistency

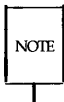
---

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.

### 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.



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.

### 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.

### 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 UNIX system does not fill them in. A check of the file would be required to tell if the error is real or not (see the section on "Holes in Files" earlier in this chapter).

### **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 under "Duplicate Blocks" and "Bad Block Numbers" above are performed for indirect blocks as well as for the direct blocks of an i-node.

### **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

### **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.



## Checking for Consistency

---

### 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.

### 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.

### 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.

### 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.

## 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.

The following abbreviations are used in the **fsck** error messages:

|       |                             |
|-------|-----------------------------|
| 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 5-11: Error Message Abbreviations in **fsck**

---

## 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.

### 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.

### 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.

## 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

### 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.

### 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.

## Checking for Consistency

---

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.

### 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.

### 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.

### 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

### 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

### 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.

## 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 file name *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 file name *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.

### 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

### Types of Error Messages—Phase 3

Phase 3 has 2 types of error messages:

1. information messages
2. messages with a RECONNECT? prompt

### 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.

### 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.

### 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

### 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

### 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.

#### 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.

## Checking for Consistency

---

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 file name *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.

**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

**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

**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.

## Checking for Consistency

---

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.

### 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.

**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.

**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.

**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.



### 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 UNIX (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 UNIX system is not rebooted immediately without **sync**, the work done by **fsck** may be undone by the in-core copies of tables the UNIX 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 6: Performance Management

|                                                         |      |
|---------------------------------------------------------|------|
| Introduction                                            | 6-1  |
| General Approach to Performance Management              | 6-2  |
| Finding Problems                                        | 6-2  |
| Fixing Problems                                         | 6-2  |
| Improving Performance                                   | 6-4  |
| Modifying the Tunable Configuration Parameters          | 6-4  |
| Improving Disk Utilization                              | 6-4  |
| Choosing Buffer Size                                    | 6-5  |
| Setting Text-Bit (Sticky-Bit)                           | 6-5  |
| File System Organization                                | 6-6  |
| Defining Best System Usage Patterns                     | 6-8  |
| <b>ps</b>                                               | 6-8  |
| User \$PATH Variables                                   | 6-9  |
| Samples of General Procedures                           | 6-10 |
| Sample Procedure for Investigating Performance Problems | 6-10 |
| Check for Excess Swapping                               | 6-10 |
| Check for Disk Bottleneck                               | 6-10 |
| Check for Potential Table Overflows                     | 6-11 |
| Shift Workload to Off-Peak Hours                        | 6-11 |
| Sample System Reconfiguration                           | 6-13 |
| Performance Tools                                       | 6-14 |
| <b>sar</b>                                              | 6-14 |
| <b>sar -a</b>                                           | 6-14 |
| <b>sar -b</b>                                           | 6-16 |
| <b>sar -c</b>                                           | 6-18 |

## Table of Contents

---

|                                |      |
|--------------------------------|------|
| <b>sar -d</b>                  | 6-20 |
| <b>sar -m</b>                  | 6-22 |
| <b>sar -q</b>                  | 6-23 |
| <b>sar -u</b>                  | 6-24 |
| <b>sar -v</b>                  | 6-25 |
| <b>sar -w</b>                  | 6-27 |
| <b>sar -p</b>                  | 6-28 |
| <b>sar -r</b>                  | 6-29 |
| <b>sar -y</b>                  | 6-30 |
| <b>sar -A</b>                  | 6-31 |
| <b>sag</b>                     | 6-33 |
| <b>timex</b>                   | 6-35 |
| <b>sadp</b>                    | 6-37 |
| <br>                           |      |
| <b>Tunable Parameters</b>      | 6-41 |
| Kernel Parameters              | 6-45 |
| Paging Parameters              | 6-50 |
| Streams Parameters             | 6-52 |
| Log Driver Parameters          | 6-55 |
| Message Parameters             | 6-55 |
| Semaphore Parameters           | 6-56 |
| Shared Memory Parameters       | 6-58 |
| Remote File Sharing Parameters | 6-58 |

---

## Introduction

This chapter describes ways to monitor and enhance the performance of your 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.

---

## General Approach to Performance Management

Performance management is an activity that may need your attention when you first set up your 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.

### 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.

### 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.

---

## Improving Performance

### Modifying the Tunable Configuration Parameters

The setting of the core system tunable parameters is accomplished by overriding the parameter entries in `/usr/include/sys/kdef.h` or `/etc/conf/modules/*/space.c` files with new definitions in the `/etc/conf/systems/system.std` file. For full definitions of the individual tunable parameters, and suggestions about setting them, refer to the section "Tunable Parameters" near the end of this chapter. (See Figure 6-5 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 the section "Performance Tools" 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 the section "Improving Disk Utilization."

Look at the end of this section for a sample of a typical reconfiguration.

### 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

### Choosing Buffer Size

Use the number of buffers (NBUF and NHBUF) given in Figure 6-5 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 6-5. The value for NHBUF is a power of 2 that is roughly one-quarter the value of NBUF.

Increasing NBUF and NHBUF, up to a point, improves system performance. A system with 2 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 UNIX 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.

### 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



## Improving Performance

---

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.

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.

## 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.

### **Organization of File System Free List**

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. 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.

### **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.

### **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.

## Improving Performance

---

- Step 4: If you have the Cartridge Tape Utilities, run **sysadm compress(1)** or **compress(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.

## 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**

### 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.

### **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.

---

## 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.

### 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 6-1 for an outline of the approach.

#### 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.

#### 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 earlier discussion of this subject under "Setting Text-Bit (Sticky-Bit)."
5. Consider adding an additional disk and balancing the most active file systems across the two disks.

### **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 (see procedure 6.1).

### **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.

## Samples of General Procedures

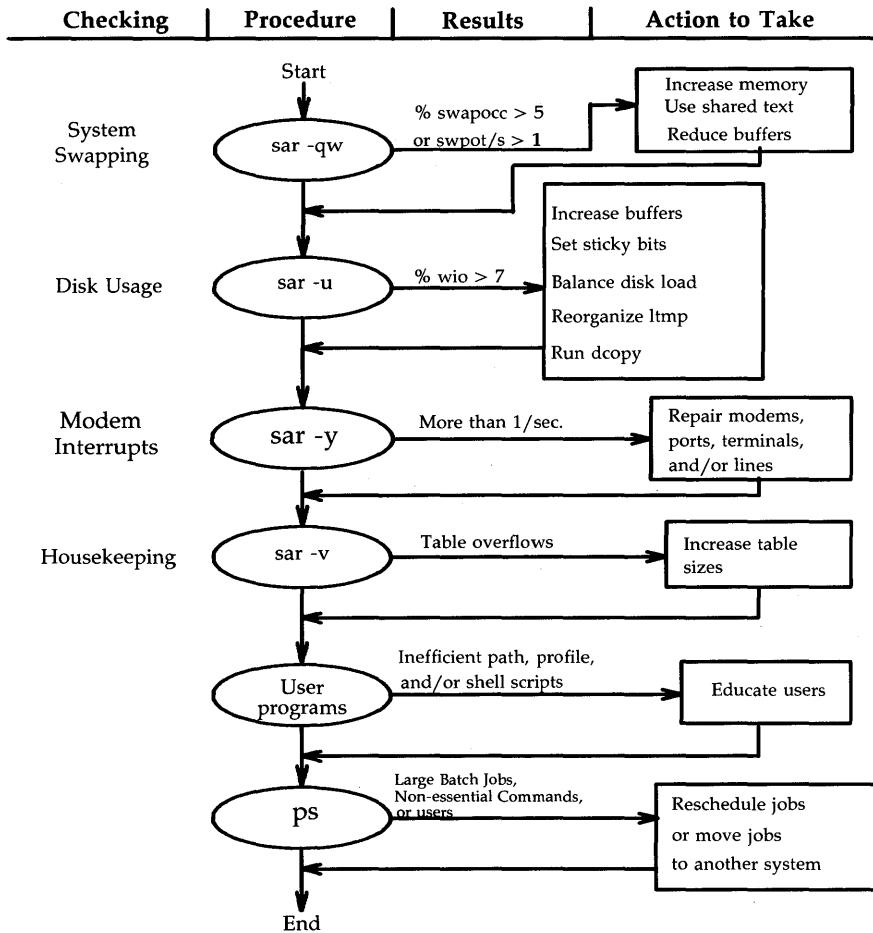


Figure 6-1: Outline of Typical Troubleshooting Procedure

## Sample System Reconfiguration

The following is a typical scenario for reconfiguring the system because of a hardware upgrade from 2 megabytes to 4 megabytes of memory. Because of the additional memory, many tunable parameters should be increased. (See Figure 6-5 for parameter values recommended for your system.) 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.

```
# cp /unix /oldunix
# cd /etc/conf/systems
# cp systems.std system.new
```

*Note: Editing of **system.new** is not shown.*

```
# cd /etc/conf
# mkunix -s system.new
# ln kernels /unix.new.1 /unix
```

When this procedure is complete, reboot the system.



---

## Performance Tools

Internal activity is measured by a number of counters contained in the UNIX 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 UNIX system internal activities described in this section are:

<b>sar</b>	Samples cumulative activity counters internal to the UNIX system and provides reports on various system-wide activities.
<b>sag</b>	Graphically displays the information collected by <b>sar</b> .
<b>sadp</b>	Produces profiles of disk access location and seek distance.
<b>timex</b>	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 UNIX 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.

### **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.

#### **sar -a**

The **sar -a** option reports the use of file access operations. The UNIX 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:

```
unix unix 3.0 2 3B2    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 UNIX 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.

**sar -b**

The **-b** option reports the following buffer activity.

<b>bread/s</b>	Average number of physical blocks read into the system buffers from the disk (or other block devices) per second.
<b>lread/s</b>	Average number of logical blocks read from system buffers per second.
<b>%rcache</b>	Fraction of logical reads found in buffer cache (100% minus the ratio of <b>bread/s</b> to <b>lread/s</b> ).
<b>bwrit/s</b>	Average number of physical blocks written from the system buffers to disk (or other block devices) per second.
<b>lwrit/s</b>	Average number of logical blocks written to system buffers per second.
<b>%wcache</b>	Fraction of logical writes found in buffer cache (100% minus the ratio of <b>bwrit/s</b> to <b>lwrit/s</b> ).
<b>pread/s</b>	Average number of physical read requests per second.
<b>pwrit/s</b>	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:

```
unix unix 3.0 2 3B2      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.

**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:

```
unix unix 3.0 2 3B2 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 10, "Remote File Sharing".

**sar -d**

The **sar -d** option reports the activity of block devices.

<b>device</b>	Name of the block device(s) that <b>sar</b> is monitoring.
<b>%busy</b>	Percent of time the device was servicing a transfer request.
<b>avque</b>	The average number of requests outstanding during the period of time (measured only when the queue is occupied).
<b>r+w/s</b>	Number of read and write transfers to the device per second.
<b>blks/s</b>	Number of 512-byte blocks transferred to the device per second.
<b>await</b>	Average time in milliseconds that transfer requests wait idly in the queue (measured only when the queue is occupied).
<b>avserv</b>	Average time in milliseconds for a transfer request to be completed by the device (for disks this includes seek, rotational latency, and data transfer times).

An example of **sar -d** is as follows:

```

unix unix 3.0 2 3B2      12/30/85
13:46:28 device %busy avque r+w/s blks/s await 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.



**sar -m**

The **sar -m** option reports on interprocess communication activities. Message and semaphore calls are reported as follows:

**msg/s**            Number of message operations (sends and receives) per second.

**sema/s**           Number of semaphore operations per second.

An example of **sar -m** output follows:

```
unix unix 3.0 2 3B2 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.

**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:

```

unix unix 3.0 2 3B2      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.

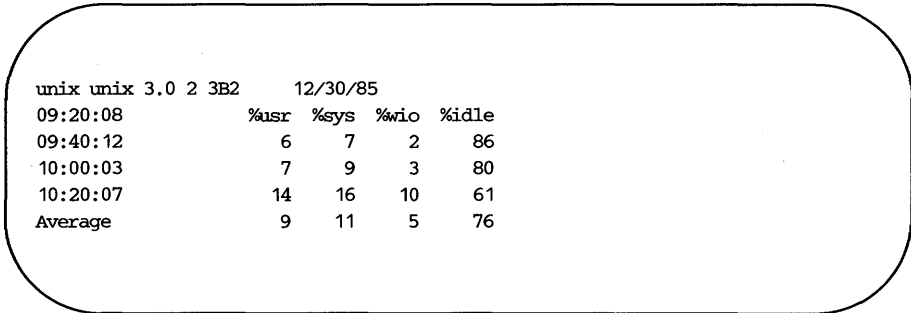
If %runocc is greater than 90 and **runq-sz** is greater than 2, the CPU is heavily loaded and response is degraded. In this case, additional CPU capacity may be required to obtain acceptable system response. If %swpocc is greater than 20, more memory or fewer buffers would help reduce swapping/paging activity.

**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 (**%usr**), 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:



```
unix unix 3.0 2 3B2      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 Chapter 10, "Remote File Sharing".

**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.

<b>proc-sz</b>	Number of process table entries presently being used/allocated in the kernel.
<b>inod-sz</b>	Number of i-node table entries presently being used/allocated in the kernel.
<b>file-sz</b>	Number of file table entries presently being used/allocated in the kernel.
<b>ov</b>	Number of times an overflow occurred (one column for each of the above three items).
<b>lock-sz</b>	The number of shared memory record table entries presently being used/allocated in the kernel.
<b>fhdr-sz</b>	No longer applicable.

The values are given as *level/table size*. An example of **sar -v** follows:

```

unix unix 3.0 2 3B2      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

```

## Performance Tools

---

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.

**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:

```
unix unix 3.0 2 3B2      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.

### **sar -p**

The **-p** option reports paging activity. The following page rates are recorded.

<b>vflt/s</b>	Number of address translation page faults per second (valid page not present in memory).
<b>pflt/s</b>	Number of page faults from protection errors per second (illegal access to page) or "copy-on-writes". <b>pflt/s</b> generally consists entirely of "copy-on-writes."
<b>pgfil/s</b>	Number of <b>vflt/s</b> per second satisfied by a page-in from the file system. (Each <b>pgfil</b> causes two <b>lreads</b> ; see <b>sar -b</b> ).
<b>rclm/s</b>	Number of valid pages per second that the system has reclaimed (added to list of free pages).

An example of **sar -p** output follows:

```
unix unix 3.0 2 3B2 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
```

**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:

```
unix unix 3.0 2 3B2 12/30/85
12:01:51 freemem freeswap
12:56:52      208      5848
```



**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:

```
unix unix 3.0 2 3B2      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
```

**sar -A**

The **sar -A** option is equivalent to **sar -udDqbwcaymprS**. The report includes RFS operations (the **-D** and **-S** options) described in Chapter 10, "Remote File Sharing". The **-A** option provides a view of overall system performance. Use it to get a more global perspective. If data from more than one time slice is shown, the report includes averages.

An example of **sar -A** follows:

## Performance Tools

```

unix unix 3.0 2 3B2    02/19/86

00:00:00  %usr  %sys  %sys  %wio  %idle
              local remote
01:00:00      0      0      0      0     100
00:00:00  device  %busy  avque  r+w/s  blks/s  await  avserv
01:00:00  hdk-0      0     1.5     0      0     10.9   21.1
          hdk-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  madmin/s
01:00:00      0      0      0      0      0      0
00:00:00  proc-sz  ov  inod-sz  ov  file-sz  ov  lock-sz  fndr-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  avg lgth  %avail  avg avail
01:00:00      0     0.0     0     0.0     0

```

## **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 6-2 shows a typical **sag** display. Running **sar** will show what data are available.

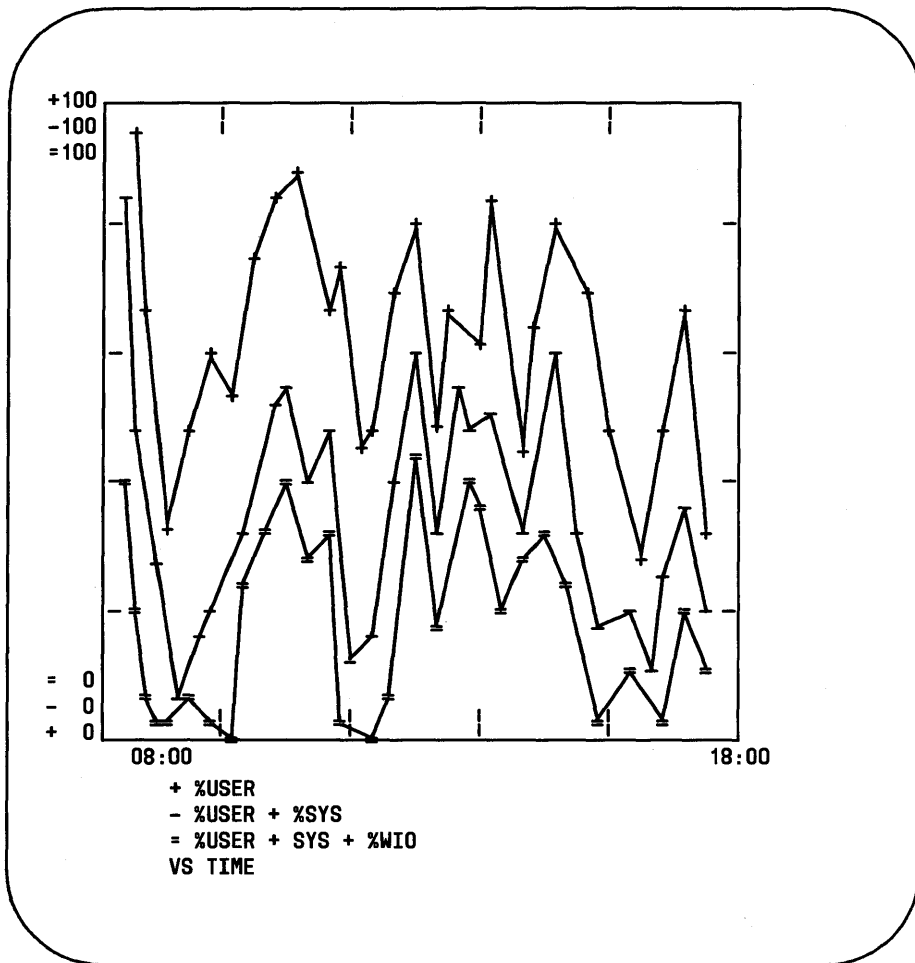


Figure 6-2: Example of **sag** Output

In Figure 6-2, 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.

## **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:

## Performance Tools

```

$ timex -s date
Wed Feb 19 08:32:50 EST 1986
real    0.17
user    0.01
sys     0.09
unix unix 3.0 2 3B2    02/19/86
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  await  avserv
08:32:05 hdisk-0    12    1.0    4    8    0.0    30.0
           hdisk-1    8    1.0    4    8    0.0    20.0
08:32:05 rawch/s canch/s outch/s rcvin/s xmtin/s madmin/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  avg lgth  %avail avg avail
08:32:05    0    0.0    0    0.0    0

```

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.

## **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.



Performance Tools

---

```
$ sadp -h -d brd0-0 3600<CR>
```

CYLINDER ACCESS HISTOGRAM

disk-0:

Total transfers = 33291

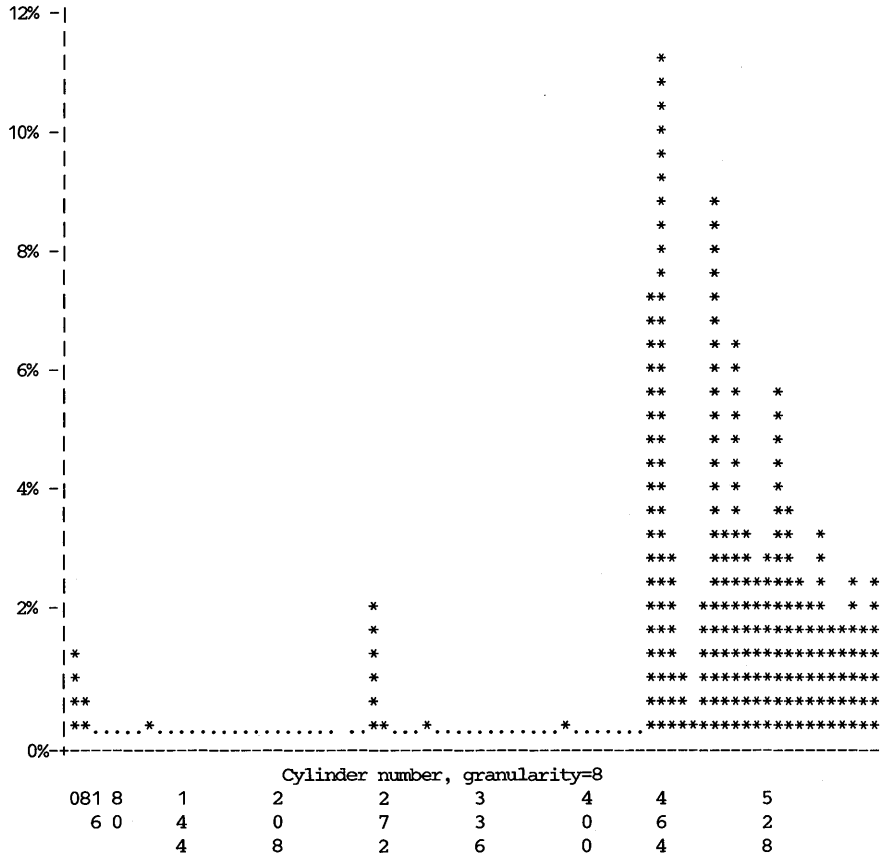


Figure 6-3: Output from **sadp**: Cylinder Access Histogram

---

SEEK DISTANCE HISTOGRAM

disk-0:

Total seeks = 30308

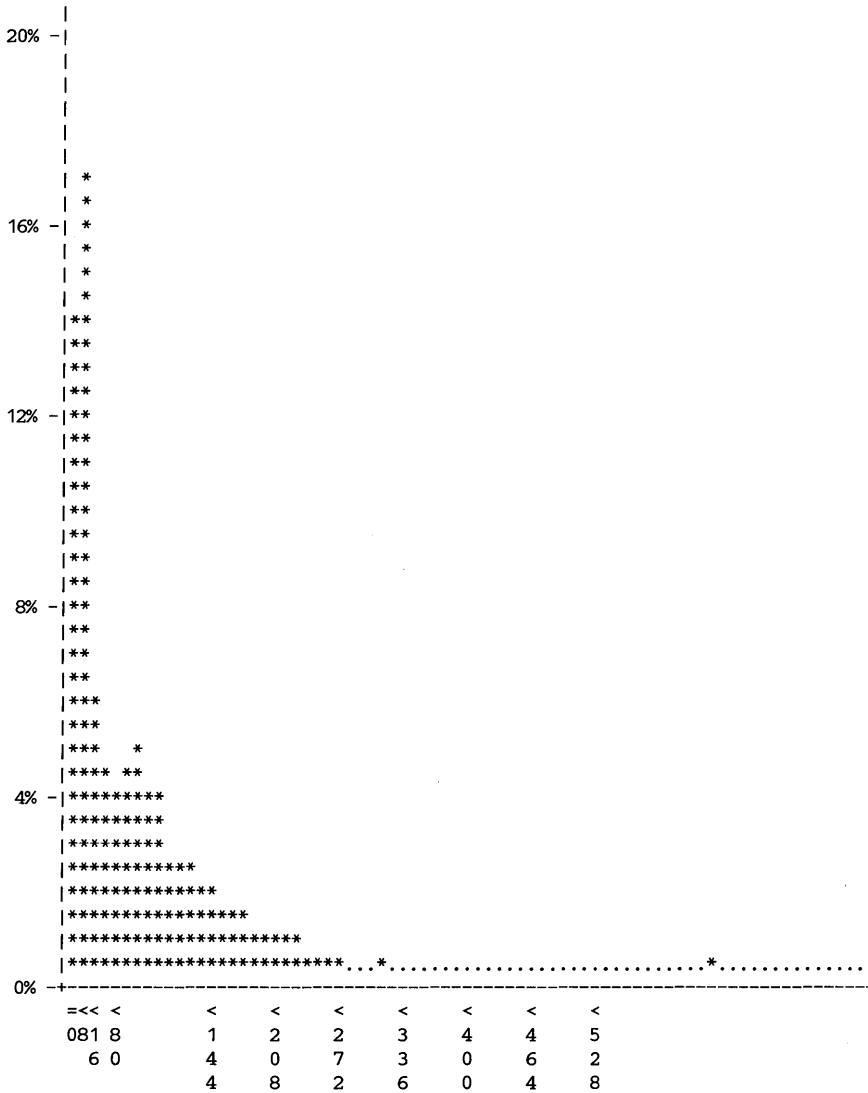


Figure 6-4: Output from **sadp**: Seek Distance Histogram

Using the **sadb** output along with the output of **/etc/mount(1M)** or **/etc/mkpart(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 6-3) 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 6-4) 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 than 16 cylinders to reference a given block, and the more to the left the stars are 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 Chapter 4, "Disk/Tape Management", and Chapter 5, "File System Administration", for more information).

---

## 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 your 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.

Figure 6-5 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 files:

- `/usr/include/sys/kdef.h`
- `/etc/conf/modules/msg/space.c`
- `/etc/conf/modules/sem/space.c`
- `/etc/conf/modules/shm/space.c`

The default parameter settings defined in the `/usr/include/sys/kdef.h` file 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/conf/modules/msg/space.c`, `/etc/conf/modules/sem/space.c`, and `/etc/conf/modules/shm/space.c`, respectively. The following notes apply to Figure 6-5:

- 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 parameters are set to specific values, as defined in the appropriate 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.

## Tunable Parameters

---

- 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-megabyte recommendations. In Figure 6-5, 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 system 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.

## Tunable Parameters

Parameter	Equipped RAM				Default Value	Size per Entry in Bytes
	1 MEG	2 MEGS	3 MEGS	4 MEGS		
NBUF	100	250	400	600	200	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	—

Figure 6-5: Suggested Parameter Values: Release 3.0 Systems (sheet 1 of 3)

## Tunable Parameters

Parameter	Equipped RAM				Default	Size per Entry in Bytes
	1 MEG	2 MEGS	3 MEGS	4 MEGS	Value	
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	—
GPGSMK	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	—

Figure 6-5: Suggested Parameter Values: Release 3.0 Systems (sheet 2 of 3)

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
SEVMX	32767	32767	32767	32767	32767	—
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	—

Figure 6-5: Suggested Parameter Values: Release 3.0 Systems (sheet 3 of 3)

## Kernel Parameters

The following parameters are defined in the `/usr/include/sys/kdef.h` file.

**NBUF** Specifies how many system buffers to allocate. The UNIX 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



## Tunable Parameters

---

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.

**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-super-user 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.

<b>NHBUF</b>	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. <b>This value must be a power of 2.</b> 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.
<b>NPBUF</b>	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.
<b>NAUTOUP</b>	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.
<b>BDFLUSHR</b>	Specifies the rate in seconds for checking the need to write the file system buffers to disk. The default is 1 second.
<b>MAXPMEM</b>	Specifies the maximum amount of physical memory to use in pages. The default value of 0 specifies that all available physical memory be used.
<b>SHLBMAX</b>	Specifies the maximum number of shared libraries that can be attached to a process at one time.
<b>FLCKREC</b>	Specifies the number of records that can be locked by the system. The default value is 100. Each entry contains 28 bytes.
<b>PUTBUFSZ</b>	Specifies the size of a circular buffer, <b>putbuf</b> , that is used to contain a copy of the last PUTBUFSZ characters written to the console by the operating system. The contents of <b>putbuf</b> can be viewed using <b>crash(1M)</b> .

## Tunable Parameters

---

<b>REL</b>	Specifies the UNIX system release.
<b>NODE</b>	Specifies the node name of the system. The default node name is <b>unix</b> (see Procedure 1.3, "Establish or Change System and Node Names").
<b>SYS</b>	Specifies the system name. The default system name is <b>unix</b> (see Procedure 1.3, "Establish or Change System and Node Names").
<b>VER</b>	Specifies the version. This value may be 1 or 2.
<b>MAXSLICE</b>	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 is normally one second (100 clock ticks on the 80386).

## 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.

<b>VHNDFRC</b>	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 <b>vhand</b> 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 <b>vhand</b> 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  $\leq 300$ ). 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  $\geq 0$  and  $< \text{GPGSHI}$ ).
- 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  $> 0$ ,  $> \text{GPGSLO}$ , and  $< 25$  percent of the number of pages of available memory.)
- GPGSMASK** 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.

## Tunable Parameters

---

**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).

## Streams Parameters

The following tunable parameters are associated with Streams processing. These parameters are defined in the `/usr/include/sys/kdef.h` 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 your 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 `L_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 3B2 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 multiplexer link (STREAMS `L_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<sub>n</sub>** 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  $0 \leq \text{STRLOFRAC} \leq \text{STRMEDFRAC}$ .

**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  $\text{STRLOFRAC} \leq \text{STRMEDFRAC} \leq 100$ .

NOTE

There is no cutoff fraction for high priority allocation requests; it is effectively 100.

## Log Driver Parameters

The configurable parameters for the log driver are found in the file `/etc/conf/modules/log/space.c`. 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.

## Message Parameters

The following tunable parameters are associated with interprocess communication messages. These parameters are defined in the `/etc/conf/modules/msg/space.c` 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.

## Tunable Parameters

---

<b>MSGMAX</b>	Specifies the maximum size of a message. The default value is 2048. The maximum size is 64 kilobytes -1.
<b>MSGMNB</b>	Specifies the maximum length of a message queue. The default value is 4096.
<b>MSGMNI</b>	Specifies the maximum number of message queues system-wide (id structure). The default value is 50.
<b>MSGSSZ</b>	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).
<b>MSGTQL</b>	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.
<b>MSGSEG</b>	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).

## Semaphore Parameters

The following tunable parameters are associated with interprocess communication semaphores. These parameters are defined in the `/etc/conf/modules/sem/space.c` file. The order in which they are described follows the order in which they are defined in the output of the `/etc/sysdef` command.

<b>SEMMAP</b>	Specifies the size of the control map used to manage semaphore sets. The default value is 10. Each entry contains 8 bytes.
<b>SEMMNI</b>	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.

<b>SEMMNS</b>	Specifies the number of semaphores in the system. The default value is 60. Each entry contains 8 bytes.
<b>SEMMNU</b>	Specifies the number of undo structures in the system. The default value is 30. The size is equal to $8 \times (\text{SEMUME} + 2)$ bytes.
<b>SEMMSL</b>	Specifies the maximum number of semaphores per semaphore identifier. The default value is 25.
<b>SEMOPM</b>	Specifies the maximum number of semaphore operations that can be executed per <b>semop(2)</b> system call. The default value is 10. Each entry contains 8 bytes.
<b>SEMUME</b>	Specifies the maximum number of undo entries per undo structure. The default value is 10. The size is equal to $8 \times (\text{SEMMNU})$ bytes.
<b>SEMVMX</b>	Specifies the maximum value a semaphore can have. The default value is 32767. The default value is the maximum value for this parameter.

## Tunable Parameters

---

**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.

## Shared Memory Parameters

The following tunable parameters are associated with interprocess communication shared memory. These parameters are defined in the `/etc/conf/modules/shm/space.c` 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.

## Remote File Sharing Parameters

RFS parameters are discussed under "Parameter Tuning" in Chapter 10, "Remote File Sharing". NSRMOUNT is included in Figure 6-5 because it is located in the `/usr/include/sys/kdef.h` file. The values associated with NSRMOUNT are 0 unless the RFS package is installed. The rest of the RFS parameters are located in `/etc/conf/modules/du/space.c`.

---

## Chapter 7: LP Spooler Administration

Introduction	7-1
How the LP Spooling System Works	7-1
Installation Information	7-2
Administrative Commands	7-3
Command Descriptions and Examples	7-4
/usr/lib/lpadmin	7-4
/usr/lib/lpsched	7-7
/usr/lib/lpshut	7-8
/usr/lib/lpmove	7-8
/usr/lib/accept	7-9
/usr/lib/reject	7-10
Printer Interface Programs	7-12
Model Interface Programs	7-12
Writing Interface Programs	7-12
Files and Directories	7-16
/usr/spool/lp/FIFO	7-16
/usr/spool/lp/default	7-16
/usr/spool/lp/log	7-16
/usr/spool/lp/oldlog	7-16
/usr/spool/lp/outputq	7-17
/usr/spool/lp/pstatus	7-17
/usr/spool/lp/qstatus	7-17
/usr/spool/lp/seqfile	7-18
/usr/spool/lp/class	7-18
/usr/spool/lp/interface	7-18
/usr/spool/lp/member	7-18
/usr/spool/lp/model	7-18

## Table of Contents

---

<b>/usr/spool/lp/request</b>	7-19
Lock Files	7-19
Cleaning Out Log Files	7-20

---

## 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.

## 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 while you continue with other work. (LP originally stood for Line Printer but has come to include many other types of printing devices.) 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 Figure 7-1.



Command	Description
<b>enable(1)</b>	Activate the named printer(s).
<b>cancel(1)</b>	Cancel a request for a file to be printed.
<b>disable(1)</b>	Deactivate the named printer(s).
<b>lp(1)</b>	Send a file or files to a printer.
<b>lpstat(1)</b>	Print the status of the LP system.

Figure 7-1: User Commands for the LP Spooling System

---

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.

## Installation Information

Instructions for installing software packages are in the *Owner/Operator Manual* for your computer. The following manual can be of specific help in the area of printer operation:

*Terminal Filters Utilities Guide* explains such functions as reverse line feed, proportional spacing, and underlining

---

## Administrative Commands

A separate set of commands available for the LP administrator is shown in Figure 7-2. 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 1, "System Security", or Procedure 1.4 for a description of how to set up a password for a system login).

Command	Description
<code>/usr/lib/accept(1M)</code>	Permit job requests to be queued for a specified destination.
<code>/usr/lib/reject(1M)</code>	Prevent jobs from being queued for a specified destination. Described on the same manual page as <b>accept(1M)</b> .
<code>/usr/lib/lpadmin(1M)</code>	Set up or change the LP configuration.
<code>/usr/lib/lpmove(1M)</code>	Move output requests from one destination to another. Described on the same manual page as <b>lpsched(1M)</b> .
<code>/usr/lib/lpsched(1M)</code>	Start the LP scheduler.
<code>/usr/lib/lpshut(1M)</code>	Stop the LP scheduler. Described on the same manual page as <b>lpsched(1M)</b> .

Figure 7-2: Administrative Commands for the LP Spooling System

---

In Figure 7-2 the administrative commands are listed in the order in which they occur in the *System Administrator's Reference Manual*. In the section that follows, we describe the commands in the order in which they are usually used.

## Command Descriptions and Examples

### **/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.

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** 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:

<b>-c</b> <i>class</i>	This argument assigns the printer specified in the <b>-p</b> option to the specified <i>class</i> .
<b>-e</b> <i>printer</i>	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 <b>-p</b> option.
<b>-h</b>	When adding a new printer, this argument means that the printer is hard-wired to the computer.
<b>-i</b> <i>interface</i>	Use this argument to specify a new interface program for the printer specified in the <b>-p</b> option. <i>interface</i> is the path name of the new program.
<b>-l</b>	When adding a new printer, this argument means that the device associated with the printer is a login terminal.

- mmodel** 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 your 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.
- rclass** Use this argument to remove a printer from a class.
- vdevice** This argument must be used when you add a new printer to the LP system. It associates the printer with the UNIX 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 **-xdest** 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**.

#### **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 hard-wired 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
#
```

### **/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.

## Administrative Commands

---

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
#
```



If many job requests are queued, there may be a delay before the `lpstat` command reports that the scheduler is running.

### `/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
#
```

### `/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.

### 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
#
```



The two requests are now renamed dqp10\_1-54 and dqp10\_1-55.

### **/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)*



### 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
#
```

### **/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.

### 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
$
```

---

## 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.

## 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.

## 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 7-3. 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 ...*

Arguments for the interface program are:

<i>P</i>	printer name
<i>id</i>	request ID returned by lp
<i>user</i>	logname of user who made the request
<i>title</i>	optional title specified by the user

<i>copies</i>	number of copies requested by user
<i>options</i>	blank-separated list of class or printer-dependent options specified by user
<i>file</i>	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:

<b>Code</b>	<b>Meaning to lpsched</b>
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 <b>lpsched</b> command notifies users by <b>mail(1)</b> that there was an error in printing the request.
greater than 127	These codes are reserved for internal use by <b>lpsched</b> . 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)**].



---

## Files and Directories

This section describes the files and directories in the LP Spooling structure.

### **/usr/spool/lp/FIFO**

**FIFO** is a special file that all the commands use to send messages to **lp sched**. Any of the LP commands may write to **FIFO**, but only **lp sched** may read it.

### **/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.

### **/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 **lp sched** error messages that occur are also recorded. The first line of the log file shows the time that the LP scheduler was started.

### **/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.

## **/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.

## **/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.

## **/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.



## **/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.

## **/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.

## **/usr/spool/lp/interface**

The interface directory contains one executable interface program for each printer that is in the LP system. The file name 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.

## **/usr/spool/lp/member**

The member directory contains one file for each LP printer. The file name is the same as the printer name. Each file contains the path name of the device to which the member is connected.

## **/usr/spool/lp/model**

This is a directory that contains the printer interface programs that are distributed with the LP Spooling Utilities.

## **/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.

## **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.

## 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 "Monitoring Files and Directories that Grow" in Chapter 5, "File System Administration").

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" #
```

Lock File	Data File
OUTQLOCK	outputq
PSTATLOCK	pstatus
QSTATLOCK	qstatus
SEQLOCK	seqfile

---

## Chapter 8: TTY Management

Introduction	8-1
Definition of Terms	8-1
The TTY System	8-3
How the TTY System Works	8-3
How to Tell What Line Settings Are Defined	8-4
How to Create New Line Settings and Hunt Sequences	8-5
How to Modify TTY Line Characteristics	8-6
How to Set Terminal Options	8-7



---

## 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

## 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 UNIX system and peripheral devices, including the system console. It shows up in commands such as <b>getty(1M)</b> and <b>stty(1)</b> , in the names of device special files such as <b>/dev/tty01</b> , and in the names of files such as <b>/etc/gettydefs</b> , which is used by <b>getty</b> .
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 <b>termio(7)</b> .

## Introduction

---

- 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)**.

---

## The TTY System

The remaining sections in this chapter describe how the TTY system operates, and how you can administer it.

### How the TTY System Works

A series of four processes [**init(1M)**, **getty(1M)**, **login(1)**, **sh(1)**] connects a user to the UNIX 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 *System 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 your 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 UNIX system has now been made.



## How to Tell What Line Settings Are Defined

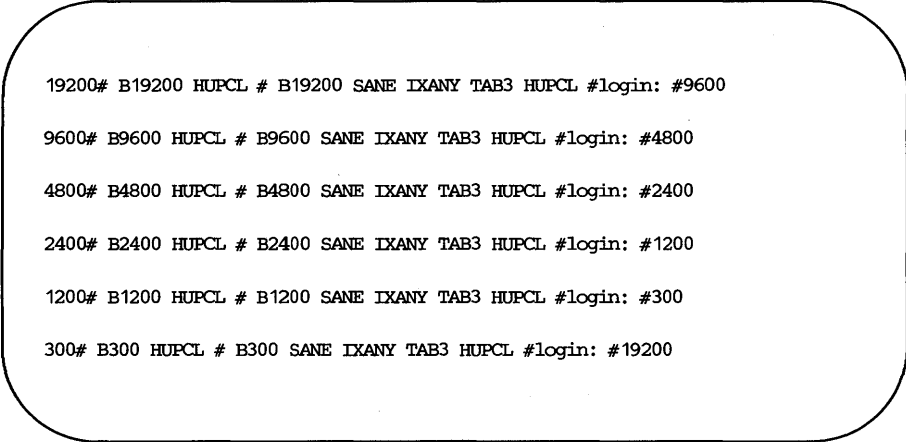
You have two ways to check line settings.

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 8-1 shows a few lines from a **gettydefs** file.



```
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 8-1: **gettydefs** Entries

---

The entries shown in Figure 8-1 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 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)**.

## 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.

## How to Modify TTY Line Characteristics

You have two ways to 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.

<i>identification</i>	A unique one- or two-character identifier for the line entry.
<i>level</i>	The run-level in which the entry is to be performed.
<i>action</i>	How <b>/etc/init</b> treats the process field [refer to the <b>init-tab(4)</b> manual page for complete information].
<i>process</i>	The shell command to be executed.

**/etc/inittab** contains several entries that spawn **getty** processes. Figure 8-2 is a selection of such entries **grep**'ed from an **/etc/inittab**.

```
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 8-2: **getty** Entries from **/etc/inittab**

---

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 **-tnn**. 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.

## How to Set Terminal Options

The TTY system described thus far establishes a basic style of communication between the user's terminal and the UNIX 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 TTY System

---

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:

- |                 |   |
|-----------------|---|
| <b>cr0 nl0</b>  | 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.  |
| <b>echoe</b>    | Erases characters as you backspace.   |
| <b>-tabs</b>    | Expand tabs to spaces when printing.  |
| <b>erase ^H</b> | Change the character-delete character to a ^H. The default character-delete character is the pound sign (#). Most terminals transmit a ^H when the <b>backspace</b> key is pressed. Specifying this option makes the <b>backspace</b> key useful. |

---

## Chapter 9: Basic Networking

Introduction	9-1
Networking Hardware	9-2
Networking Commands	9-3
User Programs	9-3
Administrative Programs	9-4
Daemons	9-5
Internal Programs	9-6
Supporting Data Base	9-7
<b>Devices</b> File	9-8
Protocols	9-13
<b>Dialers</b> File	9-13
<b>Systems</b> File	9-16
<b>Dialcodes</b> File	9-20
<b>Permissions</b> File	9-21
How Entries are Structured	9-21
Considerations	9-22
Options	9-22
<b>Poll</b> File	9-30
<b>Devconfig</b> File	9-30
<b>Sysfiles</b> File	9-31
Other Networking Files	9-32

## Table of Contents

---

Administrative Files	9-33
Direct Links	9-36
General	9-36
How the Direct Link is Connected	9-37
Computer to Computer Direct Link	9-37
BNU Software and Direct Links	9-37
Making <b>Devices</b> File Entries	9-38
Making Changes to the <b>/etc/inittab</b> File	9-39
Making Systems File Entries	9-40

---

## Introduction

The Basic Networking Utilities let computers using the UNIX operating system communicate with each other 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.



---

## 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 computer it is trying to contact 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.

---

## 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.

### 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.

## 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 *System 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 your computer communicates with. The log files contain records of each use of **uucp**, **uuto**, and **uux**.

**uucleanup** Cleans up the spool directory. It is normally executed from a shell script called **uudemon.cleanup**, which is started by **cron**.

**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.

**uuccheck** 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.

---

## 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**.

## Internal Programs

**uugetty** This program is very similar to the **getty** program except it permits a line (port) to be used in both directions. 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 *System Administrator's Reference Manual*.

---

## 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.

<b>Devices</b>	Contains information concerning the location and line speed of the automatic call unit, direct links, and network devices.
<b>Dialers</b>	Contains character strings required to negotiate with network devices (automatic calling devices) in the establishment of connections to remote computers (non 801-type dialers).
<b>Systems</b>	Contains information needed by the <b>uucico</b> daemon and the <b>cu</b> 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.
<b>Dialcodes</b>	This file contains dial-code abbreviations that may be used in the phone number field of <b>Systems</b> file entries.
<b>Permissions</b>	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.
<b>Poll</b>	This file defines computers that are to be polled by your system and when they are polled.
<b>Devconfig</b>	This file is used to configure utilities for the Basic Networking Utilities on a STARLAN NETWORK or some other transport provider that conforms to the AT&T Transport Interface.
<b>Sysfiles</b>	This file is used to assign different or multiple files to be used by <b>uucico</b> and <b>cu</b> as <b>Systems</b> , <b>Devices</b> , and <b>Dialers</b> 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.

## 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. Although provisions are made for several types of devices, only the AT&T Automatic Dial Modem and Direct Links are supported by AT&T.

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 an AT&T

Transport Interface-compatible network, such as STARLAN 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 \  
          sword: Oakgrass

You can designate a protocol to use for a device within this field. See the "Protocols" section at the end of the description of this file.

- Line* This field contains the device name of the line (port) associated with the **Devices** entry. For instance, if the Automatic Dial Modem for a particular entry was attached to the */dev/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 \  
          sword: 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 bps.

### *Dialer-Token-Pairs:*

This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a 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).

## 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 do in the form *Type,Protocol* (for example, **STARLAN,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:

```
STARLAN,e starlan - - TLIS \D
```

This says, for device **Starlan**, use **e** protocol.

## 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 ...*

## Supporting Data Base

---

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\r\c CONNECT
rixon =&-% "" \d\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\N\e \007
micom "" "" \s\c NAME? \D\r\c GO
direct
att2212c =+-, "" \r\c :--: ato12=y,T\r\c red
att4000 =,-, "" \033\r\c DEM: \033s0401\c \006 \033s0901\c \
\006 \033s1001\c \006 \033s1102\c \006 \033dT\r\c \006
att2224 =+-, "" \r\c :--: T\r\c red
nls "" "" NLPS:000:001:\N\c
```

There are also three AT&T modems that have entries in the **Dialers** file. The meaning of some of the escape characters (those beginning with "\") used in the **Dialers** file are listed below:

- \p pause (approximately 1/4 to 1/2 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

---

<code>\E</code>	enable echo checking (for slow devices)
<code>\e</code>	disable echo checking
<code>\r</code>	carriage return
<code>\c</code>	no new-line or carriage return
<code>\n</code>	send new-line
<code>\mmn</code>	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:

<code>""</code>	Wait for nothing. (In other words, proceed to the next thing.)
<code>\d</code>	Delay for 2 seconds.
<code>&gt;</code>	Wait for a >.
<code>s\p9\c</code>	Send an <b>s</b> , pause for 1/2 second, send a <b>9</b> , send no terminating new-line
<code>)-W\p\r\ds\p9\c-</code>	Wait for a <b>)</b> . If it is not received, process the string between the - characters as follows. Send a <b>W</b> , pause, send a carriage-return, delay, send an <b>s</b> , pause, send a <b>9</b> , without a new-line, and then wait for the <b>)</b> .
<code>y\c</code>	Send a <b>y</b> .
<code>:</code>	Wait for a <b>:</b> .
<code>\E\TP</code>	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 <b>\T</b> means take the phone number passed as an argument and apply the <b>Dialcodes</b> translation and the modem function translation specified by field 2 of this entry. Then send a <b>P</b> .

>	Wait for a >.
9\c	Send a 9 without a new-line.
OK	Waiting for the string OK.

## 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 the section "Other Networking Files" in this chapter for a description of the **remote.unknown** file). More than one entry may be present for a particular computer. The additional entries represent alternative communication paths that will be tried in sequential order. 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 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 penri1
```



## Supporting Data Base

---

You can define the protocol used to contact the system by adding it on to the *Type* field. The example above shows how to attach the protocol **g** to the device type **ACU**. See the information under the "Protocols" section in the description of the **Devices** file for details.

*Class* This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (for example, C1200, D1200) to differentiate between classes of dialers (refer to the discussion on the **Devices** file, *Class* field). Some devices can be used at any speed, so the keyword **Any** may be used. This field must match the *Class* field in the associated **Devices** file entry as shown below:

```
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: \  
uucpx word: xyzy
```

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 **xyzy**.

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:

<code>\N</code>	Send or expect a null character (ASCII NUL).
<code>\b</code>	Send or expect a backspace character.
<code>\c</code>	If at the end of a string, suppress the new-line that is normally sent. Ignored otherwise.
<code>\d</code>	Delay two seconds before sending or reading more characters.
<code>\p</code>	Pause for approximately $\frac{1}{4}$ to $\frac{1}{2}$ second.
<code>\E</code>	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.)
<code>\e</code>	Echo check off.
<code>\n</code>	Send a new-line character.
<code>\r</code>	Send or expect a carriage-return.
<code>\s</code>	Send or expect a space character.
<code>\t</code>	Send or expect a tab character.
<code>\\</code>	Send or expect a <code>\</code> character.
<code>EOT</code>	Send or expect EOT new-line twice.
<code>BREAK</code>	Send or expect a break character.
<code>\K</code>	Same as BREAK.
<code>\ddd</code>	Collapse the octal digits (ddd) into a single character.

## 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`.

## 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.

### 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 new-line character. Blank lines are ignored (even within multiline 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.

## Considerations

The following items should be considered when using the **Permissions** file to restrict the level of access granted to remote computers:

- All login IDs used by remote computers to login for UUCP communications must appear in one and only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry, will have the following default permissions/restrictions:
  - Local send and receive requests will be executed.
  - The remote computer can send files to your computer's **/usr/spool/uucppublic** directory.
  - The commands sent by the remote computer for execution on your computer must be one of the default commands; usually **rmail**.

## 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` options 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**.

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=/
```

## 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**.

## Devconfig File

The `/usr/lib/uucp/Devconfig` file is used when your computer communicates over a STARLAN network or some other Streams-based transport provider that conforms to the AT&T 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=STARLAN  push=ntty:tirdwr:ld0
service=uucico  device=STARLAN  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 UNIX system text editors.

## 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
```

## Supporting Data Base

---

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 `uname` command or the `cu` list using the `uname -c` command.

## 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 UNIX system text editor (`ed` or `vi`).

- |                       |  |
|-----------------------|--|
| <b>Maxuuxqts</b>      | This file defines the maximum number of <code>uuxqt</code> programs that can run at once.  |
| <b>Maxuuscheds</b>    | This file defines the maximum number of <code>uusched</code> programs that can run at once.  |
| <b>remote.unknown</b> | This file is a shell script that executes when a machine that is not in any of the <code>Systems</code> 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 ( <code>chmod 000 remote.unknown</code> ), your system will accept any conversation requests. |

---

## 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 4-digit job sequence number assigned by UUCP. Work files contain the following information:

- Full path name of the file to be sent or requested
  - Full path name of the destination or user file name
  - User login name
  - List of options
  - Name of associated data file in the spool directory. If the **uucp -c** or **uuto -p** option was specified, a dummy name (**D.0**) is used
  - Mode bits of the source file
  - Remote user's login name to be notified upon completion of the transfer
- D. (Data file)** Data files are created when it is specified in the command line to copy the source file to the spool directory. The names of data files have the following format:

*D.systmxxxxyyy*

where *systm* is the first five characters in the name of the remote computer, *xxxx* is a 4-digit job sequence number assigned by **uucp**. The 4-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 4-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.

---

## Direct Links

### 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.

If the link is established, the computers can not be separated by more than 100 cable-feet (two 50-foot cables connected together) because the longest cable available is 50 feet.

If the two computers are separated by more than 100 cable-feet, a null-modem cable must be constructed as follows:

- Pin 1 to 1
- Pin 2 to 3
- Pin 3 to 2
- Strap pin 4 to 5 in the same plug
- Pin 6 to 20
- Pin 7 to 7
- Pin 8 to 20
- Pin 20 to 6
- Pin 20 to 8.

---

## How the Direct Link is Connected

### Computer to Computer Direct Link

The following instructions guide you in establishing a direct link between two computers.

NOTE

Do not establish a direct link between the two computers before **uugetty**s are started on both computers (from **inittab** file). If the ports are running **getty**s, both computers will continuously try to log in to each other and will crash.

- Step 1: Connect one end of the first shielded cable to the selected port on your computer. Be sure to attach the ground connector.
- Step 2: Connect the other end of the first shielded cable to the ACU/Modem Adapter. (If you want to connect the two computers over a distance greater than 100 feet, use a Terminal/Printer Adapter attached to a null modem cable of the appropriate length, instead of an ACU/Modem adapter.)
- Step 3: Connect the Terminal/Printer Adapter to the ACU/Modem Adapter.
- Step 4: Connect the second shielded cable to the Terminal/Printer Adapter.
- Step 5: Connect the other end of the second shielded cable to the appropriate port on the remote computer.

## BNU Software and Direct Links

Ideally, systems that have a direct link should be running the same release of the UNIX system to have the full set of capabilities available. (Bidirectional ports, which are supported by the **uugetty** program, were introduced with UNIX System V Release 2.0 Version 1.) However, lack of a common version of the UNIX operating system will not prevent you from using the Basic Networking Utilities.

## Direct Links

---

This section describes the software files that must be modified on your computer in order to accommodate a direct link connection. You may want to consult the documentation provided with your computer if you are linking directly to a remote computer other than your computer.

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 Menu subcommand **uucpmgmt**. (See the *Owner/Operator Manual* for information on using the System Administration Menu.)

### 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 Menu 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 (1200, 9600, or 19200).

To access the System Administration Menu 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.*

## 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 **ugetty** instead of **getty**. This means that if two computers (both using **ugetty**) 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 **ugetty**.

If the direct link is connecting your computer 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 **ugetty** 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 computer 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 **ugetty** 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 computer to poll the remote computer. (You can use the System Administration Menus subcommand **uucpmgmt pollmgmt** to do this.) If the remote

## Direct Links

---

computer is allowing only outgoing traffic on the link, it must be set up to poll your computer.

Entries in the `/etc/inittab` file can be changed using the System Administration Menu 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 (1200, 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.*

## 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 Menu 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 (1200, 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 'uucpnmgt',
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 10: Remote File Sharing

Overview	10-1
Resource Sharing	10-2
Domains	10-4
Administration	10-5
Security	10-6
Connect Security	10-7
Mount Security	10-8
User and Group Security	10-8
Sharing Resources	10-10
Advertise Resources	10-11
Aliases	10-12
Resource Security	10-12
Local Advertise Table	10-13
Domain Advertise Table	10-14
Unadvertise	10-15
Mount Remote Resources	10-16
Automatic Remote Mounts	10-17
Mounting Guidelines	10-17
Mounting Rules	10-18
Host Mount Table	10-19
Domain Mount Table	10-20
Unmounting	10-21
Forced Unmount	10-21
Mapping Remote Users	10-23
How Mapping Works	10-23
Mapping Components	10-24
Rules Files	10-24
<b>idload</b> Command	10-27

## Table of Contents

---

Host <b>passwd</b> and <b>group</b> Files	10-28
Choosing Mapping Strategy	10-28
No Mapping	10-29
Mapping Remote IDs	10-29
Mapping Remote Names	10-30
<b>Maintenance</b>	10-33
Introduction	10-33
Host Administration	10-33
Domain Administration	10-33
Monitoring and Tuning	10-34
Host Maintenance	10-34
Start Up Host	10-37
Passwords and Verification	10-38
<b>rfstart</b> Options	10-39
Shut Down Host	10-41
Change Host Password	10-41
Update User and Group Information	10-41
Domain Maintenance	10-42
Add New Hosts	10-42
Remove a Host	10-43
Change Current Domain Name Server	10-43
Reassign Domain Name Server	10-44
Add New Domains	10-44
Update User and Group Files	10-44
Remote File Sharing Mode ( <b>init 3</b> )	10-44
Entering Run Level 3	10-45
<b>init 3</b> Processing	10-46
Changing <b>init 3</b> Processing	10-47
Adding RFS Mode Scripts	10-48
Recovery	10-49
Primary Goes Down	10-49
Primary and Secondaries Go Down	10-50

Server Goes Down	10-50
Monitoring	10-52
Remote System Calls ( <b>sar -Dc</b> )	10-53
CPU Time ( <b>sar -Du</b> )	10-55
Server Processes ( <b>sar -S</b> )	10-57
Resource Usage ( <b>fusage</b> )	10-58
Remote Disk Space ( <b>df</b> )	10-59
Parameter Tuning	10-60
RFS Parameters	10-61



---

## Overview

Remote File Sharing (RFS) allows computers running UNIX System V Release 3.0 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 that 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 UNIX 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 UNIX 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 UNIX 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.  |

## Resource Sharing

Sharing a resource on a Remote File Sharing system begins with a path name to a UNIX 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.

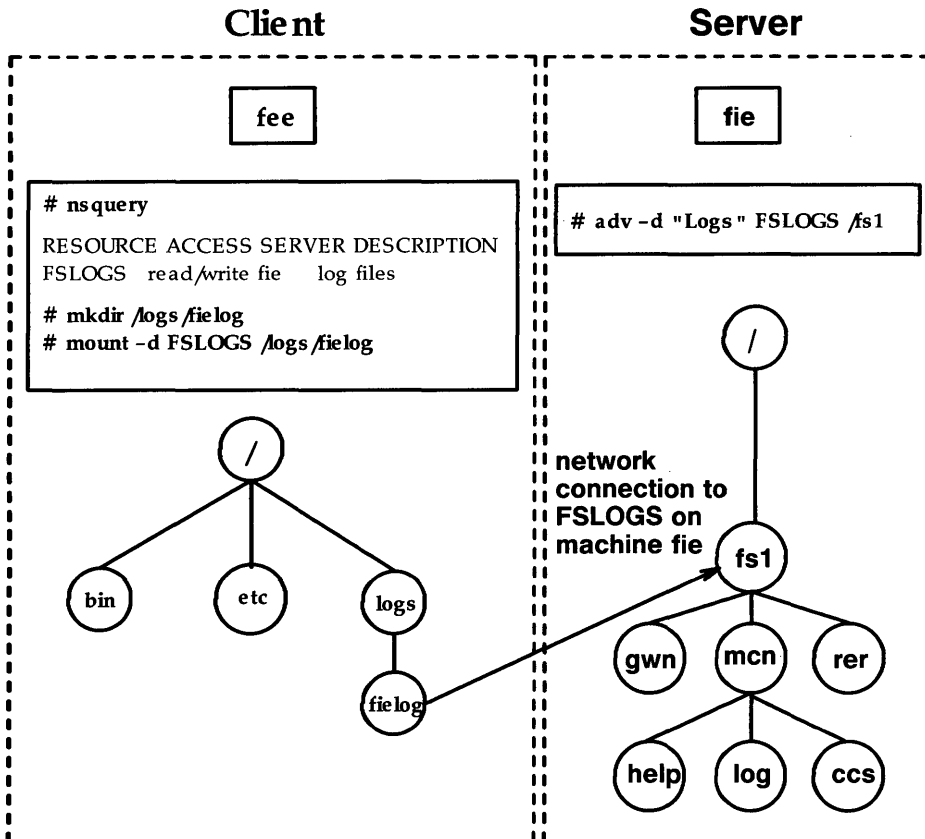


Figure 10-1: Example - Sharing Resources

Figure 10-1 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 **/fs1** on its file system tree. The administrator advertises **/fs1** as a resource called **FSLOGS**.



## Overview

---

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 **/fs1** 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.

## 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.

## 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 UNIX 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 UNIX system file structure looks the same, but it is extended by mounting remote resources (`mount -d` command).

## 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 log in as they always have. As for remote accessibility, you can set up security to allow only certain remote hosts to access your resources.

Figure 10-2 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.

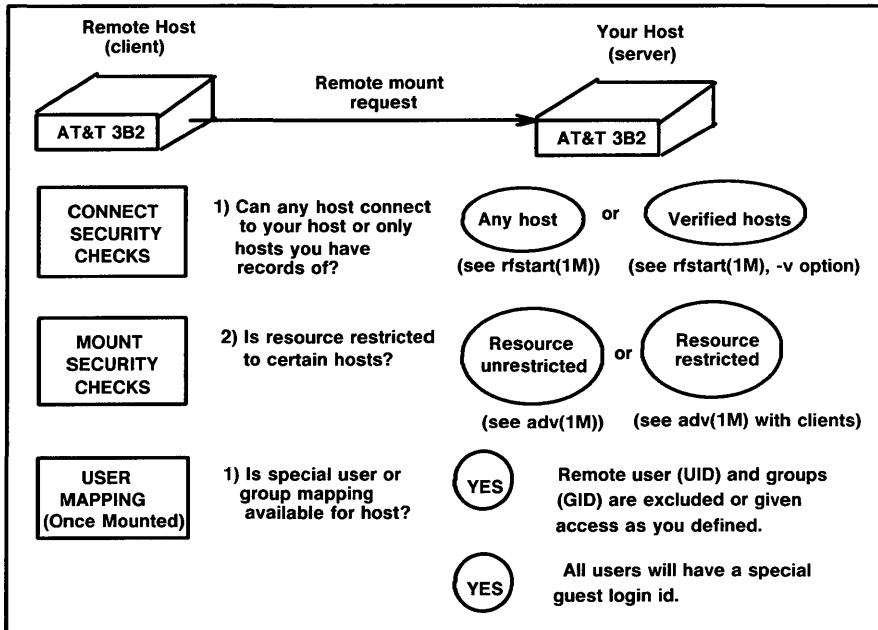


Figure 10-2: Example - Remote File Sharing Security

### 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.

### 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).

### 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 the "Mapping Remote Users" section of this chapter for more details.)

---

## 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 the section titled "Monitoring" in this chapter.)
- df(1M)** Reports disk space availability on remote hosts. (See the section titled "Monitoring" in this chapter.)

To remove remote resources, you will use these commands:

- unadv(1M)** Unadvertises one of your advertised resources.
- umount** 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.

## 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.



## Sharing Resources

---

- pathname* This is the full path name 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 "Aliases" below.)
- 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.

### 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.

### 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.

**UNIX system security**    Normal UNIX system access security, governing read, write, and execute permissions, will apply to any advertised resource.

### **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 "UNIX 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

## Sharing Resources

---

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

### 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:

<i>host</i> name	To list the resources advertised by a particular host in your domain.
<i>domain.</i> 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.)
<i>domain.host</i> 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.

## 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.

## Sharing Resources

---

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 the "Advertise Resources" section of this chapter 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.

## 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*.

### 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.

### 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.

### 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 [**umount(1M)**], remove the program, then readvertise the resource. (See Chapter 6 for a discussion of local sticky bit use.)

**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.



## Sharing Resources

---

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
$
```

## 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 output only 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 path name field, it means you have unadvertised the resource, but it is still mounted on the listed remote machines.

## 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.

## 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.

## Sharing Resources

---

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.

---

## 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).

### 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.

## 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.

### 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 following example 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.

## Mapping Remote Users

---

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.

### **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.



## Mapping Remote Users

---

- 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.

### Host passwd 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 rerunning **idload**.

### 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.

## 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.

## 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.



The **exclude 0** line is strongly recommended to prevent possible security breaches from **root** users on other systems.

## Mapping Remote Users

---

```
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.

## 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.

**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.

**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
```

## Mapping Remote Users

---

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

---

# Maintenance

## 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.

## 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.

## 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.

## Maintenance

---

- **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.

## 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.

## 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/conf/modules/du/space.c`**

This file contains the Remote File Sharing tunable parameters. (See the "Tunable Parameters" section of this chapter.)

**`/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 <b>dnsname -a</b> to see the name of your host's domain and the type of network it is running on.
current name server	Type <b>rfadmin</b> to print the host name of the current name server.
mapping rules	Type <b>idload -n</b> to see the user mapping that is done based on the <b>uid.rules</b> and <b>gid.rules</b> files in <b>/usr/nserve/auth.info</b> . 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.

### 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 (multiuser 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 AT&T Transport Interface-compatible network, such as STAR-LAN NETWORK.

- super-user

You must be logged in as **root**.

### 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 reentering it with the **rfpasswd** command. If that doesn't work, follow the next sequence shown.

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.

### 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
rfdaemon
server
```

There may be multiple processes of some of these names running.

## Shut Down Host

The **rfstop** command disconnects your host from the Remote File Sharing network. Before you can use **rfstop**, you must:

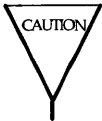
- unadvertise all your resources (**unadv**).
- unmount everything you have mounted from remote hosts (**umount**).
- make sure all your advertised resources are unmounted from remote hosts (can be forced by using **fumount**).

These steps will happen automatically when you leave init state 3.

## 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.



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.

## 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 the "Mapping Remote Users" section or the **idload(1M)** manual page.

## 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 the "Host Maintenance" section.

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 <b>dname -a</b> to see the name of your host's domain and network type.
name servers	Look at the <b>rfmaster</b> file in <b>/usr/nserve</b> . 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 <b>rfadmin</b> 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.

### 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.

## 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.

## 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.



## 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.

## 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.

## 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.)

## 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 the "Operating Levels" section of Chapter 3, "Processor Operations", 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.

### 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 multiuser mode. Therefore, when you type **init 3** from single-user mode, all multiuser processes (**gettys**, **cron**, etc.) will be started, followed by Remote File Sharing mode processes.

- from multiuser mode (run level **2**)

When **init 3** is run from run level 2, **init** checks that all multiuser 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:

### 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 "Automatic Remote Mounts" in this chapter 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**.



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**.

## 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.

### 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:

*S00name*

or

*K00name*

The file names can be split into three parts:

- |               |  |
|---------------|--|
| <b>S or K</b> | The first letter of each file defines whether the process should be started ( <b>S</b> ) or killed ( <b>K</b> ) upon entering the new run level.                                     |
| <i>00</i>     | 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). |
| <i>name</i>   | The rest of the file name is the <b>/etc/init.d</b> file name this file is linked to.  |

For example, the **init.d** file **netdaemon** is linked to **rc2.d** file **S68netdaemon** and **rc0.d** file **K67netdaemon**. When you enter **init 2**, this file is executed with the **start** option: **sh S68netdaemon start**. When you enter **init 0**, this file is executed with the **stop** option: **sh K67netdaemon stop**.

## 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.

### 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.

### 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 File 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. Readvertise 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.

### 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.

**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, <b>rfudaemon</b> sends a disconnect message and the resource name to the <b>rfuadmin</b> shell script.   |
| fumount    | When a resource is unmounted ( <b>fumount</b> ) by the server, the <b>rfudaemon</b> sends a fumount message and the resource name to the <b>rfuadmin</b> shell script.  |
| fuwarn     | When a server sends a message that a resource is about to be unmounted ( <b>fumount</b> ), the <b>rfudaemon</b> sends a fuwarn message, the resource name, and the number of seconds before the resource will be unmounted to the <b>rfuadmin</b> shell script. |

**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.



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)**.]

## 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 the Mounting Guidelines section of this chapter 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.

### **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.

```
$ sar -Dc
fee fee 3.0 2 3B2 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 10-3: 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.)

### **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.

```

$ sar -Du
fee fee 3.0 2 3B2 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 10-4: Output from `sar -Du`

---

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 the section titled "Parameter Tuning.") You may also want to make sure clients are doing I/O in an efficient way (see `sar -Dc`).

## 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**.

```

$ sar -S

fee fee 3.0 2 3B2    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 10-5: Output from **sar -S**

As an administrator you can set the number of server processes available to service remote system calls (see "Parameter Tuning"). 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 in Figure 10-6.

### 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.

### 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%.

## 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 10-6.)

```
# 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 10-6: 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.

### 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
```

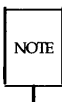
where *resource* is the name of a remote resource mounted on your machine.



```
# df USERsrc USERmail
/usr/src (USERsrc ): 5436 blocks 2202 i-nodes
/usr/mail (USERmail ): 5436 blocks* 2202 i-nodes
#
```

Figure 10-7: Output from **df**

---



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.

## 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.

## 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/conf/modules/du/space.c`, except for `NSRMOUNT`, which is in `/usr/include/sys/kdef.h`. Read these files for default values. If you change any of these values, see Chapter 6 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.

The following table lists the Remote File Sharing parameters and recommended values for different uses of Remote File Sharing.

RFS Tunable Parameter Settings  
(file `/etc/conf/modules/du/space.c`)

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 `/usr/include/sys/kdef.h`.

---

## Diskette Naming Conventions

Diskette names have the following format:

**f**<drive#><density flag><#sectors/track><side flag>[**t**]

The *density flag* can be either **d** (double density) or **q** [quad density (AT386 only)].

The sector numbers supported are 4, 8, 9, and 15 (AT386 only).

The *side flag* can be either **d** (double-sided) or **s** (single-sided).

If the name ends with a **t**, the total disk (including the first cylinder) is being specified.

For example, **/dev/dsk/f0d4d** specifies a floppy drive connected to controller board 0, with double density and 4 sectors/track that is double-sided. Cylinder 0 is NOT specified.

**/dev/dsk/f0d9dt** specifies the floppy drive connected to controller board 0, with double density and 9 sectors/track that is double-sided. In this case, the entire diskette, including Cylinder 0, is specified.



---

## **Introduction**

Appendix B describes directories and files of interest to a system administrator.



---

## Directories

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

<b>bck</b>	Directory used to mount a backup file system for restoring files.
<b>bin</b>	Directory containing public commands.
<b>dev</b>	Directory containing special files that define all of the devices on the system.
<b>dgn</b>	Directory containing diagnostic programs.
<b>etc</b>	Directory containing administrative programs and tables.
<b>install</b>	Directory used by System Administration to mount utilities packages for installation and removal (/install file system).
<b>lib</b>	Directory containing public libraries.
<b>lost+found</b>	Directory used by <b>fsck(1M)</b> to save disconnected files.
<b>mnt</b>	Directory used to temporarily mount file systems during restoration of the operating system from floppy disks.
<b>save</b>	Directory used by Simple Administration for saving data on floppies.
<b>tmp</b>	Directory used for temporary files.
<b>usr</b>	Directory used to mount the /usr file system.

---

## Files

The following files and directories are important in the administration of your computer:

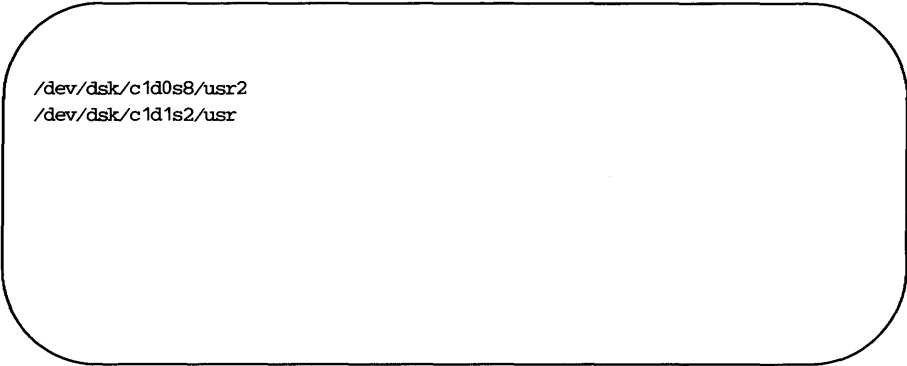
- **/etc/checklist**
- **/etc/fstab**
- **/etc/gettydefs**
- **/etc/group**
- **/etc/init.d Directory**
- **/etc/inittab**
- **/etc/conf Directory**
- **/etc/motd**
- **/etc/passwd**
- **/etc/profile**
- **/etc/rc0**
- **/etc/rc0.d Directory**
- **/etc/rc2**
- **/etc/rc2.d Directory**
- **/etc/rc.d Directory**
- **/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.

### **/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 multiuser mode (run level 2). The **root** file system (**/dev/rdisk/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 **/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

## `/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 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 *System 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

---

## `/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 `/etc/gettydefs` file.

```
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: #console2
console2# B300 HUPCL OPOST ONLCR # B300 SANE IXANY TAB3 #Console Login: #console3
console3# B2400 HUPCL OPOST ONLCR # B2400 SANE IXANY TAB3 #Console Login: #console4
console4# B4800 HUPCL OPOST ONLCR # B4800 SANE IXANY TAB3 #Console Login: #console5
console5# B19200 HUPCL OPOST ONLCR # B19200 SANE IXANY TAB3 #Console Login: #console
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

## **/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:

<i>group name</i>	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.
<i>password</i>	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.
<i>group id</i>	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.
<i>login names</i>	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 <b>/etc/newgrp</b> command to become a member of the group.

Figure B-4 shows a typical **/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

## `/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/rcn.d`, where *n* is the appropriate run level. Files are not executed from this directory. They are only executed from `/etc/rcn.d` directories.

## `/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 multiuser modes. Chapter 3, "Processor Operations," 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 `/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:



## Files

---

<i>identification</i>	The identification field is a one- or two-character identifier for the line entry. The identifier is unique for a line.
<i>run-state</i>	The run-state defines the run level in which the entry is to be processed.
<i>action</i>	The action field defines how <b>/etc/init</b> treats the process field. [Refer to the <b>inittab(4)</b> manual page in the <i>Programmer's Reference Manual</i> for complete information.]
<i>process</i>	The process field defines the shell command that is to be executed.

```
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:
p1:s1234:powerfail:/etc/led -f# start green LED flashing
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

## **/etc/conf Directory**

The **/etc/conf** directory contains files that define the configuration of hardware devices, software drivers, system parameters and aliases. The files are used by **/etc/mkunix** to obtain device information for the generation of device driver and configurable module files. The **/etc/sysdef(1M)** program uses the **conf** 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 system file in the **/etc/conf/systems** directory. [Refer to the **/etc/system(4)** manual page in the *Programmer's Reference Manual* for additional information.]

## **/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.

## **/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:

<i>login name</i>	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.
-------------------	--

---

<i>passwd</i>	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.
<i>user id</i>	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.
<i>group id</i>	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.
<i>account</i>	The fifth field is used by accounting programs. This field typically contains the user name, department number, and bin number.
<i>login directory</i>	The sixth field defines the full path name of the login directory.
<i>program</i>	The seventh field defines the program to be executed after login. If it is null, the shell ( <b>/bin/sh</b> ) 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.]

```
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:ZQJgJg55150HI:3:3:0000-Admin(0000):/usr/src:
adm:iQ00Marb9NXZM:4:4:0000-Admin(0000):/usr/adm:
uucp:rQ5fM/1P8Z6yU:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp:4Rka.6jZpnZeI:10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/uuc
ico
rje:FRStaYwCm0do:18:18:0000-rje(0000):/usr/rje:
trouble:PRS31ul156VCI:70:1:trouble(0000):/usr/lib/trouble:
lp:YR6Oh4EJQfTy2:71:2:0000-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.pymZKs: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:umount 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

## **/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 computer default profile.

```
# 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

---

## **/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, 5, or 6 can be done by adding a file to the **/etc/shutdown.d** directory. Figure B-8 shows a typical **/etc/rc0** 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*  
do  
    {  
        if [ -s ${f} ]  
        then  
            /bin/sh ${f} start  
        fi  
    }  
done  
  
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)

## `/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.

## **/etc/rc2**

The **/etc/rc2** file contains a shell script that is executed by **/etc/init** on transitions to run level 2 (multiuser 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.

- |                     |   |
|---------------------|---|
| <b>MOUNTFILESYS</b> | Sets up and mounts file systems. Builds the mount table and mounts the <b>root (/)</b> and user ( <b>/usr</b> ) file systems. Makes the <b>/usr/tmp</b> directory, cleaning up (deleting) any previous files in that directory.                             |
| <b>autoconfig</b>   | Makes a <b>/unix</b> if self-configuration occurred during the boot sequence. The new in-memory operating system is copied to <b>/unix</b> .  |
| <b>cron</b>         | Starts the <b>cron</b> daemon by executing <b>/etc/cron</b> .   |
| <b>syssetup</b>     | Removes the <b>/etc/ps_data</b> to force the <b>/bin/ps</b> command to read the <b>/unix</b> file. Outputs the system configuration if the <b>/etc/prtconf</b> command exists. Outputs the system trademark information.                                    |
| <b>uucp</b>         | When basic networking is added to the system, the <b>uucp</b> file is added to this directory. The <b>uucp</b> file deletes <b>uucp</b> locks (LCK*), status files (STST*), and temporary files (TM*) under the <b>/usr/spool/uucp</b> directory structure. |
| <b>lp</b>           | When line printer spooling is added to the system, the <b>lp</b> file is added to the <b>rc.d</b> . The <b>lp</b> 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 **/etc/rc2** file.

```
# "Run Commands" executed when the system is changing to init state 2,
# traditionally called "multiuser".
. /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)

---

## **/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.

## **/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 UNIX System release 3.0 were moved to **/etc/rc2.d**. This directory is only maintained for compatibility reasons.)

## **/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).

## **/etc/rc3.d Directory**

The **/etc/rc3.d** directory contains files executed by **/etc/rc3** for transitions to system run level 3 (multiuser 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**

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 10 for details.)

## **/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:

- |                      |   |
|----------------------|---|
| <b>except</b>        | A list of the directories and files that should not be copied as part of a backup ( <b>savefiles</b> ) is maintained in this file.      |
| <b>timestamp/...</b> | The date and time of the last backup (volume or incremental) is maintained for each file system in the <b>/etc/timestamp</b> directory. |

## **/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 **/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
    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]                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
fi
if [ -z "${TZ}" -a -r /etc/TIMEZONE ]
then
    . /etc/TIMEZONE
fi
echo '
Shutdown started.    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)

## `/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 UNIX System V Release 3.0 were moved to `/etc/rc0.d`. This directory is only maintained for compatibility reasons.)

## **/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.

The format of the **TZ** is as follows:

```
TZ=TTT#SSS
```

Explanations for the fields of the **TZ** variable are as follows:

<i>TTT</i>	The three-character abbreviation for the local time zone.
<i>#</i>	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.
<i>SSS</i>	The three-character abbreviation for the local daylight saving time zone. This field is entered only if daylight saving time is observed.

```
# Set timezone environment to default for the machine
TZ=EST5EDT
export TZ
```

Figure B-11: Typical **/etc/TIMEZONE** File

## **/etc/utmp**

The **/etc/utmp** file contains information on the run-state of the system. This information is accessed with a **who -a** command.

## **/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.

## **/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.

```
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

## `/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.

```
! *** cron started ***  pid = 237 Sun Aug 19 14:06:45 1984
> CMD: /usr/lib/uucp/uu demon.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/uu demon.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/uu demon.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/uu demon.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/uu demon.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/uu demon.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

---

## `/usr/lib/help/HELPLLOG`

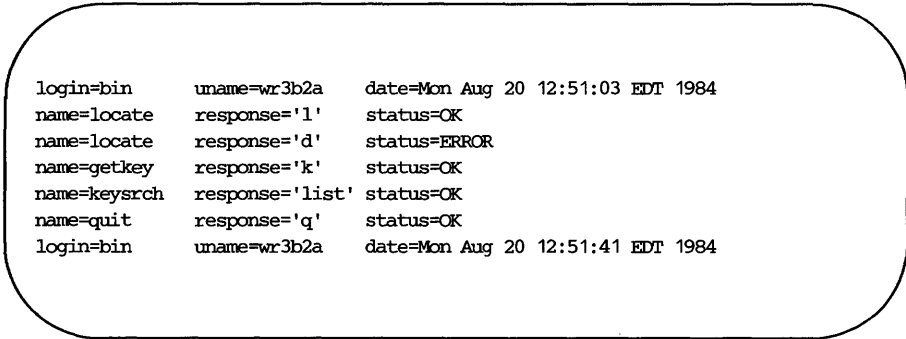
Providing that **help** monitoring has been enabled, a history of all actions taken by the `/usr/bin/help` command is kept in the `/usr/lib/help/HELPLLOG` file. The **HELPLLOG** file is copied to `/usr/lib/help/oHELPLLOG`, and a new `/usr/lib/help/HELPLLOG` file is created by the `/usr/lib/help/helpclean` command. Executing the **helpclean** command twice in succession zeros out the **HELPLLOG** and the **oHELPLLOG** files. Note that `/etc/cron`, `/etc/rc0`, or `/etc/rc2` can be used to clean up the **HELPLLOG** 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/HELPLLOG > /tmp/help
mv /tmp/help /usr/lib/help/HELPLLOG
```

Figure B-14 shows the information typically found in the `/usr/lib/help/HELPLLOG` file.



```
login=bin      uname=wr3b2a   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=wr3b2a   date=Mon Aug 20 12:51:41 EDT 1984
```

Figure B-14: Typical `/usr/lib/help/HELPLLOG` File

---

## `/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. (Refer to the *3B2 computer Spell Utilities Guide* for information on adding words to the dictionary and cleaning up the `spellhist` file.)



## **/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).

## **/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.

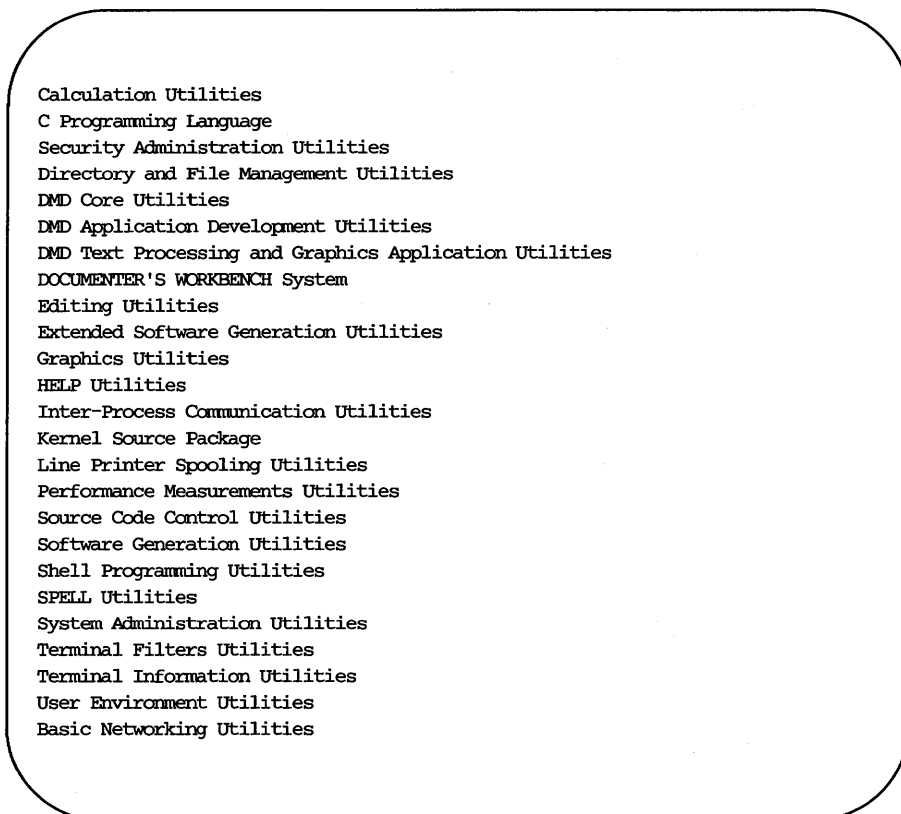
The example shown in Figure B-15 was taken from a system with a 32-megabyte hard disk. All of the utilities identified in Figure B-15 cannot be loaded on a 10-megabyte hard disk at the same time. The full names of the utilities are contained in the **/usr/options** files. The contents of the files identified 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

---

## `/usr/spool/cron/crontabs`

The `/usr/spool/cron/crontabs` directory contains crontab files for `adm`, `root`, and `sys` logins. Providing their log names 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:

<i>minute</i>	The minutes field is a 1- or 2-digit number in the range 0 through 59.
<i>hour</i>	The hour field is a 1- or 2-digit number in the range 0 through 24.
<i>day</i>	The day field is the numerical day of the month in the range 1 through 31.
<i>month</i>	The month field is the numerical month of the year in the range 1 through 12.
<i>day-of-week</i>	The day-of-week field is the numerical day of the week where Sunday is 0, Monday is 1, . . . and Saturday is 6.
<i>command</i>	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.

```
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

---

[Refer to the **crontab**(1) manual page in the *User's Reference Manual* for additional information.]

---

## UNIX System Error Messages

UNIX 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 UNIX system error message tables are divided into severity classes. A description of each severity class is given with each table.

---

## UNIX 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.

<b>Error Message (Prefaced by NOTICE)</b>	<b>Description/Action</b>
bad block on floppy drive, slice #	An out-of-range block number was specified. Check the file system. (Procedure 5.3)
bad block on integral hard disk drive #, partition #	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 floppy drive, slice #	A bad count in the super block. Check the file system (Procedure 5.3).
bad count on integral hard disk drive #, partition #	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 floppy drive, slice #	The free list count is inconsistent. Check the file system (Procedure 5.3).
Bad free count on integral hard disk drive #, partition #	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 interprocess communication messages.

Error Message (Prefaced by NOTICE)	Description/Action
Configured value of NOFILES (#) is greater than max (#) NOFILES set to #.	The value of NOFILES in the system file exceeds the allowed maximum. No immediate action required. Maximum value has been used. To avoid repetitions on future config boots, change the system file [see system(4)] and run <b>mkunix</b> .
Configured value of NOFILES (#) is less than min (#) NOFILES set to #.	The value of NOFILES in the system file is less than the allowed minimum. No immediate action required. Minimum value has been used. To avoid repetitions on future config boots, change the system file [see system(4)] and run <b>mkunix</b> .
/dev/swap doesn't match swap-dev; changing it on fs	The system was booted from a new device for the very first time. This is an advisory message only.
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.
Floppy Access Error: Consult the System Administration Utilities Guide	Make sure the diskette is in the drive and the door is closed. Remove write protect tab or mount file system with -r (read only) option. The diskette requires reformatting, or its file system runs off the end of the disk. Reconfigure the diskette file system to be within the boundaries of the diskette. The diskette may be defective. If reformatting fails, replace the diskette.



## NOTICE Messages

---

Error Message (Prefaced by NOTICE)	Description/Action
!getcpages – waiting for # contiguous pages	
hard disk: Bad sanity word in VTOC on drive #	The volume table of contents (VTOC) is either bad or the wrong version. Restore the hard disk from the core diskettes. Select the full restore option.
hard disk: Bad sanity word on drive #	Hard disk defect table is bad. Call your service representative.
hard disk: cannot access sector: #, track: #, cylinder: #, on drv #	Log that the error message occurred. No further action is required for this message. The <b>hdlogger</b> will identify any problems. Other messages requiring action may be output if the <b>hdlogger</b> logs errors. Refer to Chapter 4, "Disk/Tape Management", for additional information.
hard disk: Cannot read defect map on drive #	Log that the error message occurred. The hard disk defect map is bad. Call your service representative to resolve the problem.
hard disk: Cannot read sector 0 on drive #	Log that the error message occurred. The hard disk defect table is bad. Call your service representative to resolve the problem.
hard disk: Cannot read the VTOC on drive #	Restore the hard disk from the core diskettes. Select the full restore option. Call your service representative if you cannot resolve the problem.
hard disk: Can't recall drive #	Log that the error message occurred. Hardware problem is indicated. Call your service representative to resolve the problem.
hard disk: Drive # is in the 1.0 layout. It cannot be used until the conversion is made to the current layout.	Restore the hard disk from the core diskettes. Select the upgrade option.
hard disk: Drive # not equipped	Log that the error message occurred. Hardware problem is indicated. Reboot the system. If after reboot the problem still exists, call your service representative.

Error Message (Prefaced by NOTICE)	Description/Action
hard disk: drive # out of service	Log that the error message occurred. Hardware problem is indicated. Reboot the system. If after reboot the problem still exists, call your service representative.
hard disk: partition # on drive # is marked read-only	Trying to write to a read only partition. Check your command/program or mount the partition as read/write.
hard disk: too little space allocated in driver for defect table on drive #	The current UNIX Operating System cannot work with the disk configuration. Call your service representative to resolve the problem.
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.
<i>str</i> - Insufficient memory to allocate # pages - system call failed	<i>str</i> may be <i>growreg</i> , <i>exec</i> , <i>fork</i> , or other kernel function name. Try again later.
no space on floppy drive, slice #	The disk has reached its capacity.
no space on integral hard disk drive #, partition #	Clean up file system indicated by the partition number. Repartition if necessary.
Out of inodes on 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 integral hard disk drive #, partition #	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 floppy drive, slice #	Probable bad diskette. Try a different diskette. If the problem persists, call your service representative.

## NOTICE Messages

---

Error Message (Prefaced by NOTICE)	Description/Action
page read error on integral hard disk #, partition #	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 <b>hdfix</b> .
Pdinfo doesn't match parameters for device #	(AT286/386 only) The number of heads or cylinders for a hard disk specified in the VTOC doesn't match the values obtained from CMOS memory. VTOC values are used.
proc on q	No action is necessary. The system tried to put a process on the run queue that was already on the queue. Call your service representative if the problem persists.
shmctl - couldn't lock # pages into memory	Memory overcommitted. Try again later.
Soft read error corrected by ECC algorithm: unit #, sector #	The controller allowed a defective sector to be read successfully.
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.
str on hard disk unit #, partition #	
str on disk unit #, partition #	
str on Ram Disk partition #	
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.

Error Message (Prefaced by NOTICE)	Description/Action
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 <b>maxfc</b> was found to be greater than 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 <b>maxsc</b> was found to be greater than 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.

---

## UNIX System WARNING Messages

WARNING error messages indicate that the UNIX system may stop functioning if corrective action is not taken. These error messages are defined alphabetically in the following table.

<b>Error Message (Prefaced by WARNING)</b>	<b>Description/Action</b>
Alternates table invalid on device #	
Cannot read alternates table on device #	
Cannot read pinfo on device #	
Cannot read vtoc on device #	
CANNOT READ VOLUME LABEL ON DEVICE #	(MB1 only) Get error reading disk sector 0 on hard disk.
ERROR SENDING DISK PARAMETERS FOR DEVICE #	(MB1 only) Probable i214 controller problem
ERROR SENDING NEW DISK PARAMETERS FOR DEVICE #	(MB1 only) Probable i214 controller problem
floppy disk: Bad address returned from VTOP	Log that the error message occurred. A bad address was passed to the driver ioctl, or a Virtual TO Physical (VTOP) address conversion failed.
floppy disk timeout: work list flushed	Make sure diskette is in the drive and the door is closed. Contact your service representative if the problem recurs.
hard disk: Bad sanity word in VTOC on drive #	Restore the hard disk from backup using the full restore option. (Procedure 5.4)

Error Message (Prefaced by WARNING)	Description/Action
hard disk: Bad sanity word on drive #	The defect table on the hard disk needs to be rebuilt. Contact your service representative for assistance.
hard disk: cannot access sector #, head #, cylinder # on drive #	The hard disk error log should report that this disk block is logged. In single-user mode, use the <b>hdefix</b> command.
hard disk: Cannot read defect map on drive #	The defect table on the hard disk needs to be rebuilt. Contact your service representative for assistance.
hard disk: cannot read sector 0 on drive #	The defect table on the hard disk needs to be rebuilt. Contact your service representative for assistance.
hard disk: Cannot read the VTOC on drive #	Restore the hard disk from backup (Procedure 5.4). If the problem persists, it may be necessary to replace the drive.
hard disk: cannot recal drive #	Probably a hardware problem with the disk drive.
hard disk: Drive # is in the 1.0 layout. It can not be used until conversion is made to the current layout.	Restore the hard disk using the "upgrade" option of the restore floppy disks. The conversion can be made by running the <b>fmthard(1M)</b> command either from the floppy restore disk or from the first disk to the second disk of a two-disk configuration. Contact your service representative for assistance.
hard disk: Drive # not equipped	Probably a hardware problem. The drive is either not present or has become disconnected.
hard disk: drive # out of service	Probably a hardware problem. Check cabling. Reboot if necessary.
hard disk: partition # on drive # is marked read only	If you wish to write to this disk partition, the permissions in the VTOC must be changed. Use <b>fmthard(1M)</b> .

## WARNING Messages

---

<b>Error Message (Prefaced by WARNING)</b>	<b>Description/Action</b>
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.
Invalid pinfo on device #	
INVALID VOLUME LABEL ON DEVICE #	(MB1 only) Bad magic number in Intel volume label on hard disk.
inode table overflow	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.
out of swap space: needed # blocks	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.

---

**WARNING Messages**

<b>Error Message (Prefaced by WARNING)</b>	<b>Description/Action</b>
out of text	If recurring problem, reconfigure the system with increased text table limits (NTEXT).
PDINFO DOES NOT MATCH LABEL ON DEVICE #	(MB1, only) Inconsistent VTOC on hard disk.
PORTS: EXPRESS QUEUE OVERLOAD: One entry lost	Log that the error message occurred. The PORTS board may be insane. Reboot the system. Contact your service representative if the problem recurs.
PORTS: FAULT - opcode=#, board #, subdev=#, bytectn=#, buff address=#	An invalid PORTS opcode was encountered. "Pump" the associated PORTS board. If the problem still exists, reboot the system.
PORTS: QFAULT - opcode=#, board #, subdev=#, bytectn=#, buff address=#	The PORTS job queue is invalid. "Pump" the associated PORTS board. If the problem still exists, reboot the system.
PORTS: SYSGEN failure on board #	Log that the error message occurred. Reboot the system. Contact your service representative if the problem recurs.
PORTS: timeout on drain board (#), port (#)	Log that the error message occurred. Reboot the system. Contact your service representative if the problem recurs.
PORTS: unknown completion code: #	Log that the error message occurred. Possible hardware problem. Reboot the system. Contact your service representative if the problem recurs.
PORTS: unknown pump command: #	Log that the error message occurred. Reboot the system. Contact your service representative if the problem recurs.
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).



## WARNING Messages

---

<b>Error Message (Prefaced by WARNING)</b>	<b>Description/Action</b>
too few HDE equipped disk slots	Log that the error message occurred. Reboot the system. Contact your service representative if the problem recurs.
Unable to change hard disk sector size.	(AT386 only)
unreadable CRC hard disk error: maj/min = #/# block = #	Log that the error message occurred. Reboot the system. The <b>hdlogger</b> logged a disk access error. Refer to Chapter 4, "Disk/Tape Management", for additional information.
Vtoc invalid on device #	
... 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.

---

## UNIX System PANIC Messages

PANIC error messages indicate a problem severe enough that the UNIX 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 UNIX 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.



After the panic completes, take a crash dump if desired (see Procedure 3.5 for information about how to do that). Reboot the UNIX system. If panics occur frequently, contact your service representative.

## PANIC Messages

---

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.
cannot expand TEXT with swap	A request for text growth was rejected since text cannot be expanded.
cannot mount root	The root file system was corrupted or nonexistent when trying to boot. Reboot the UNIX system. If reboot fails, then do a partial restore from the core diskettes.
data size error in swapin	The size of the swapped-in process data section is not the same size that was swapped out. The system description file may be incorrect, or a modified driver may have caused the error.
devtab	The list header for the chain of buffers attached to the block-type device cannot be found.
floppy disk: Bad address returned from VTOP	A Virtual TO Physical (VTOP) address conversion failed.
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.
hard disk: Bad address returned from VTOP	A Virtual TO Physical (VTOP) address conversion failed.
HD controller: write fault	(AT386 only) Manual says panic—only way to clear write fault from drive is power cycle.
HD controller: command aborted	(AT386 only) Controller problem
HD controller: stays busy	(AT386 only) Can't get controller to accept command.
iget — mounted on inode not in mount table.	Inode has mount flag set but not in mount table. Probable software bug.

<b>Error Message (Prefaced by PANIC)</b>	<b>Description/Action</b>
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.
input — 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).

## PANIC Messages

---

Error Message (Prefaced by PANIC)	Description/Action
KERNEL MODE <i>str</i> FAULT	The processor unexpectedly registered the error given by <i>str</i> . These errors are detected by the module and are listed in fault type class 3.
kernel mode trap. Type #	Processor trap detected.
kernel process stack exception	A stack reference caused a memory fault.
Krnlfilt returned to k_trap	Take a sysdump and reboot.
kseg — ptmemall failed	No physical memory available for kernel or driver when needed.
loadstbl — bad section id	Invalid section number was passed to loadstbl. Probable kernel bug.
loadstbl — segment table too short	Segment table too short to map entire region. Probable kernel bug.
main — copyout of icode failed	The kernel was not able to copy the assembly code which is used to start up the system.
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 hard disk drive 1	(AT386 only) No hard disk found.
no imt	No indirect mount point was found in the mount table.

Error Message (Prefaced by PANIC)	Description/Action
no procs	A process table entry was not found during a fork when an entry is available.
no swap for page table	Swap/paging device full in the middle of a swap operation.
no swap for u-area	Swap/paging device full in the middle of a swap operation.
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.
procdup() problem	An inconsistency has occurred between the parent process and the child process text size.
process exception, user = 0x#	Accessing a process control area caused a memory fault.
process exception, proc = 0x#, pcbp = 0x#	The system took a process exception while in the kernel or an interrupt handler. "proc" is the pointer to the process table entry for the currently running process. "pcbp" points to the current pcb. Dump and reboot. Suspect a hardware problem if the error persists.
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.
shmslp: swap <i>n</i> count <i>n</i> valid <i>n</i>	Log that message occurred.
srmount — cannot mount root	The kernel was not able to mount the root file system when booting. Check the VTOC on the disk.

## PANIC Messages

---

<b>Error Message (Prefaced by PANIC)</b>	<b>Description/Action</b>
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". Check VTOC. Boot off another disk and run <b>fsck</b> on root.
svirtophys — movtrw failed	The "movtrw" instruction failed to convert a virtual to a physical address. Check to see if the address was within range.
svirtophys — not present.	A system virtual address points to a non-resident page. Probably a bad pointer.
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.
sys3b — DELMEM premove failed	An attempt to remove memory from a system's available memory failed. Kernel bug. Do not remove so much memory.
SYSTEM ALIGNMENT ERROR INTERRUPT	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.
SYSTEM BUS TIME OUT INTER- RUPT	A bus request by the system was not fulfilled within the allotted time.
SYSTEM PARITY ERROR INTER- RUPT	A memory parity error occurred. If this occurs repeatedly, a hardware problem exists.
text size error in swapin	The size of the swapped-in process text section is not the same size of the swapped-out text.
Timeout table overflow	The queue for timeout requests has overflowed while attempting to add another entry.

Error Message (Prefaced by PANIC)	Description/Action
total size error in swapin	The computed size of the entire process swapped-in is not the same as that swapped-out.
trap recursion	A trap occurred from within the system trap handler.
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.
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.
User mode trap. Type #	Processor trap detected.
usrxmemflt. impossible condition	An invalid page fault occurred.
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.
xsap() current process Ox#	The swap-out process has been called with a process table address pointing to its own entry.
xswap error	An illegal operation was passed to the xswap function.



---

## UNIX System Call Error Messages

A system call that is unsuccessful returns an impossible value to the calling process. This impossible value is almost always a -1. When a system call is successful, a value of 0 is returned to the calling process. Any time a system call is unsuccessful, an external variable called *errno* is set to one of the numbers in the following table.

When a -1 value is returned, the *errno* variable contains the number corresponding to the reason of the failure. The *errno* variable is only valid immediately after a system call failure. It is not cleared on successful system calls. These error numbers are defined in the `/usr/include/sys/errno.h` header file.

Error		Description
Number	Name	
1	EPERM	<b>Not Owner</b> Typically this error indicates an attempt to modify a file in some way forbidden except to its owner or super-user. It is also returned for attempts by ordinary users to do things allowed only to the super-user.
2	ENOENT	<b>No such file or directory</b> This error occurs when a file name is specified and the file should exist but does not, or when one of the directories in a path name does not exist.
3	ESRCH	<b>No such process</b> No process can be found corresponding to that specified by <code>pid</code> in <code>kill</code> or <code>ptrace</code> .
4	EINTR	<b>Interrupted system call</b> An asynchronous signal (such as interrupt or quit), which the user has elected to catch, occurred during a system call. If execution is resumed after processing the signal, it will appear as if the interrupted system call returned this error condition.
5	EIO	<b>I/O error</b> Some physical I/O error. This error may in some cases occur on a call following the one to which it actually applies.

Error		Description
Number	Name	
6	ENXIO	<b>No such device or address</b> I/O on a special file refers to a subdevice which does not exist, or beyond the limits of the device. It may also occur when, for example, a tape drive is not on-line or no disk pack is loaded on a drive.
7	E2BIG	<b>Arg list too long</b> An argument list longer than 5,120 bytes is presented to a member of the <i>exec</i> family.
8	ENOEXEC	<b>Exec format error</b> A request is made to execute a file which, although it has the appropriate permissions, does not start with a valid magic number [see, <i>a.out</i> (4)].
9	EBADF	<b>Bad file number</b> Either a file descriptor refers to no open file, or a read (respectively write) request is made to a file which is open only for writing (respectively reading).
10	ECHILD	<b>No child processes</b> A <i>wait</i> , was executed by a process that had no existing or unwaited-for child processes.
11	EAGAIN	<b>No more processes</b> A <i>fork</i> , failed because the system's process table is full or the user is not allowed to create any more processes.
12	ENOMEM	<b>Not enough space</b> During an <i>exec</i> , <i>brk</i> , or <i>sbrk</i> , a program asks for more space than the system is able to supply. This is not a temporary condition; the maximum space size is a system parameter. The error may also occur if the arrangement of text, data, and stack segments requires too many segmentation registers, or if there is not enough swap space during a <i>fork</i> .

## System Call Errors

---

Error		Description
Number	Name	
13	EACCES	<b>Permission denied</b> An attempt was made to access a file in a way forbidden by the protection system.
14	EFAULT	<b>Bad address</b> The system encountered a hardware fault in attempting to use an argument of a system call.
15	ENOTBLK	<b>Block device required</b> A non-block file was mentioned where a block device was required, e.g., in <b>mount</b> .
16	EBUSY	<b>Mount device busy</b> An attempt to mount a device that was already mounted or an attempt was made to dismount a device on which there is an active file (open file, current directory, mounted-on file, active text segment). It will also occur if an attempt is made to enable accounting when it is already enabled.
17	EEXIST	<b>File exists</b> An existing file was mentioned in an inappropriate context, e.g., <b>link</b> .
18	EXDEV	<b>Cross-device link</b> A link to a file on another device was attempted.
19	ENODEV	<b>No such device</b> An attempt was made to apply an inappropriate system call to a device; e.g., read a write-only device.
20	ENOTDIR	<b>Not a directory</b> A non-directory was specified where a directory is required, for example in a path prefix or as an argument to <b>chdir(2)</b> .

Error		Description
Number	Name	
21	EISDIR	<b>Is a directory</b> An attempt to write on a directory.
22	EINVAL	<b>Invalid argument</b> Some invalid argument (e.g., dismounting a non-mounted device; mentioning an undefined signal in <b>signal</b> , or <b>kill</b> , reading or writing a file for which <b>lseek</b> has generated a negative pointer). Also set by the math functions described in the (3M) entries of this manual.
23	ENFILE	<b>File table overflow</b> The system's table of open files is full, and temporarily no more <b>opens</b> can be accepted.
24	EMFILE	<b>Too many open files</b> No process may have more than 20 file descriptors open at a time.
25	ENOTTY	<b>Not a typewriter</b>
26	ETXTBSY	<b>Text file busy</b> An attempt to execute a pure-procedure program which is currently open for writing (or reading). Also an attempt to open for writing a pure-procedure program that is being executed.
27	EFBIG	<b>File too large</b> The size of a file exceeded the maximum file size (1,082,201,088 bytes) or ULIMIT; see <b>ulimit(2)</b> .
28	ENOSPC	<b>No space left on device</b> During a write to an ordinary file, there is no free space left on the device.
29	ESPIPE	<b>Illegal seek</b> An <b>lseek</b> was issued to a pipe.
30	EROFS	<b>Read-only file system</b> An attempt to modify a file or directory was made on a device mounted read-only.

## System Call Errors

---

Error		Description
Number	Name	
31	EMLINK	<b>Too many links</b> An attempt to make more than the maximum number of links (1000) to a file.
32	EPIPE	<b>Broken pipe</b> A write on a pipe for which there is no process to read the data. This condition normally generates a signal; the error is returned if the signal is ignored.
33	EDOM	<b>Math argument</b> The argument of a function in the math library (3M) is out of the domain of the function.
34	ERANGE	<b>Result too large</b> The value of a function in the math package (3M) is not representable within machine precision.
35	ENOMSG	<b>No message of desired type</b> An attempt was made to receive a message of a type that does not exist on the specified message queue; See <code>msgop(2)</code> .
36	EIDRM	<b>Identifier removed</b> This error is returned to processes that resume execution due to the removal of an identifier from the file system's name space [see <code>msgct(2)</code> , <code>semct(2)</code> , and <code>shmct(2)</code> ].
45	EDEDLK	<b>Deadlock</b> A deadlock situation was detected and avoided.

---

## Pump Error Messages

**pump** is a user-level command that downloads firmware to feature cards mounted in your computer backplane slots during the power-up sequence. **pump** error messages appear on the console terminal when a phase in the autopump sequence fails. Although these errors are not fatal to the entire system, the affected card is not operational. Therefore, normal services provided by the device are not accessible.

Error Message (See Notes at End)	Description/Action
Pump: <i>/dev/devname</i> returned a CIO FAULT during <i>phase</i>	Turn power off using power switch. After powerdown sequence has completed, turn power on again. If error occurs again, contact your service representative. The UNIX System may panic soon.
Pump: <i>/dev/devname</i> returned a CIO invalid Queue Entry during <i>phase</i>	Same action as above.
Pump: <i>/dev/devname</i> did not respond during <i>phase</i>	Same action as above.
Pump: A timeout has occurred on <i>/dev/devname</i> during <i>phase</i>	Same action as above.
Pump: There was no return for <i>/dev/devname</i> during <i>phase</i>	Same action as above.
Pump Error: 208-ioctl call	

## Pump Errors

---

NOTE

The term */dev/devname* refers to */dev/ttyAB*, */dev/NI*, and so on, where:

A = Feature Card slot on backplane  
B = Port on Expansion Port Feature Card  
NI = Network Interface Feature Card

NOTE

The term *phase* refers to one of the following phases:

**Reset** = Reset of the Feature Card so that pumping can occur  
**Download** = Pumping to firmware of Feature Card  
**Sysgen** = Initialization of Feature Card to known state  
**Force call to function** = Calling the starting address of firmware that was downloaded.

---

## LP Spooler Error Messages

This section provides a description of the error messages that are associated with LP commands. The following variables are used in the error messages:

<i>file(s)</i>	Indicates the file or files that are to be printed.
<i>dest</i>	Indicates the name of the destination printer.
<i>printer-id</i>	Indicates the request identification number of the printout. For example, <i>dqp10_2-46</i> is the printer name followed by the request identification number.
<i>printer-name</i>	Indicates the name of the printer.
<i>program-name</i>	Indicates the program name that was executed.
<i>user</i>	Indicates the user who requested the printout.

Following each message is an explanation of the probable cause of the error and the corrective action to take. If you are not able to correct all the error conditions you encounter, call your service representative for assistance.

Some lengthy error messages that appear all on one line on the display are too long to be shown as one line in the documentation. When more than one line is required in the documentation to show a one-line error message, a "\\" is used to split the message.

Error Message	Description/Action
dest is an illegal destination name	The <i>dest</i> you used is not a valid destination name. Use the <b>lpstat -p</b> command to list valid destination names.
file is a directory	The file name you typed is a directory and cannot be printed.
xx is not a request id or a printer	The argument you used with the <b>cancel</b> command is not a valid request identification number or a printer name. Use the <b>lpstat -t</b> command to give you all the printers and requests waiting to get printed.



## LP Error Messages

---

Error Message	Description/Action
xx is not a request id	The request identification number you used with the <b>lpmove</b> command is not a valid request identification number. To find out what requests are valid, use the <b>lpstat -u</b> command.
xx not a request id or a destination	You used an invalid request identification number or destination with the <b>lpstat</b> command. To find out what is valid, use the <b>lpstat -t</b> command.
dest not accepting requests since <i>date</i>	Requests to the printer which you are trying to use have been stopped by the <b>reject</b> command.
Can't access FIFO	The named pipe file <b>/usr/spool/lp/FIFO</b> is incorrect. The mode should be 600.
LP Administrator not in password file	You must have an entry in the <b>/etc/passwd</b> file for "lp," and you must belong to the group "bin."
acceptance status of destination "printer-name" unknown	Use the <b>accept</b> command to enable the printer so that it will accept requests.
can't access file "xx"	The mode could be wrong on your directory or the file that you are trying to access.
can't create class "xx"-it is an existing printer name	The class name you are trying to use has already been given to a printer. You will have to use another name or remove the printer to use the class name.
can't create new acceptance status file	The mode may be wrong on the <b>/usr/spool/lp</b> directory. It should be 755 with the owner "lp" and the group "bin."
can't create new class file	The mode may be wrong on the <b>/usr/spool/lp</b> directory. It should be 755 with the owner "lp" and the group "bin."

Error Message	Description/Action
can't create new interface program	The mode may be wrong on the <code>/usr/spool/lp/interface</code> directory. It should be 755 with the owner "lp" and the group "bin."
can't create new member file	The mode may be wrong on the <code>/usr/spool/lp/member</code> directory. It should be 755 with the owner "lp" and the group "bin."
can't create new printer status file	The mode may be wrong on the <code>/usr/spool/lp/pstatus</code> . It should be 644 with the owner "lp" and the group "bin."
can't create new request directory	The mode may be wrong on the <code>/usr/spool/lp/request</code> directory. It should be 755 with the owner "lp" and the group "bin."
can't create printer "printer-name" -- it is an existing class name	The printer-name you are trying to use has already been used as a class name. You will have to assign another name for the printer.
can't create new output queue	The mode on the file <code>/usr/spool/lp/seqfile</code> is incorrect. It should be 644, and the mode on the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.
can't create new sequence number file	The mode on the file <code>/usr/spool/lp/seqfile</code> is incorrect. The mode of the file should be 644, and the mode of the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.

## LP Error Messages

---

Error Message	Description/Action
can't create request file "xx"	The mode on the file <code>/usr/spool/lp/request/printer-name/r-id</code> is incorrect. <i>Printer-name</i> is the name of the printer such as <code>dqp10</code> , and <i>r-id</i> is the request identification number. The mode of the file should be <code>444</code> , and the mode of the directory should be <code>755</code> . The owner should be <code>"lp"</code> and the group should be <code>"bin."</code> This may be corrected by typing the command at a later time.
can't fork	You either have several processes running and are not allowed to run anymore, or the system has all the processes running that it can handle. You will have to rerun this command later.
can't lock acceptance status	This is a temporary file in <code>/usr/spool/lp</code> that prevents more than one <code>"lp"</code> request from being taken at any given instant. You will have to rerun this command later.
can't lock output queue	The file <code>/usr/spool/lp/QSTATLOCK</code> prevents more than one <code>"lp"</code> request from being printed on a printer at a time. You will have to rerun this command later.
can't lock printer status	The temporary file <code>/usr/spool/lp/PSTATLOCK</code> prevents more than one <code>"lp"</code> request from being printed on a printer at a time. You will have to rerun this command later.
can't lock sequence number file	The file <code>/usr/spool/lp/SEQLOCK</code> prevents more than one <code>"lp"</code> request from getting the next printer-id (request identification) number at a time. You will have to rerun this command later.
can't move request printer-id	<i>Printer-id</i> is the request identification number that cannot be moved. You will probably have to change the modes on the files and directories in <code>/usr/spool/lp/request</code> . Also, you will have to manually move the request from the disabled printer directory to the new destination after you shut down the LP scheduler.

Error Message	Description/Action
can't open class file	The <b>lp</b> program is trying to access the list of classes for printers. One reason it may not be able to open the class file is that the system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later time.
can't open member file	The <b>lp</b> program is trying to access the list of members in the directory <b>/usr/spool/lp/member</b> . The system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later time.
can't open xx file in MEMBER directory	There are a couple of reasons why file <b>xx</b> in the <b>/usr/spool/lp/member</b> directory cannot be opened. The mode on the file could be incorrect. It should be 644. Another possibility is that the system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later time.
can't open xx file in class directory	One possibility why file <b>xx</b> cannot be opened is that the mode on the file or directory is incorrect. The file mode should be 644, and the directory mode should be 755. Another possibility is that the system has the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open xx	You cannot print on printer <b>xx</b> because the mode is incorrect on <b>/dev/tty</b> . The mode should be 622.
can't open FIFO	The mode on the named pipe file <b>/usr/spool/lp/FIFO</b> may be incorrect. It should be 600. Or, the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open MEMBER directory	The mode on the directory <b>/usr/spool/lp/member</b> could be incorrect. It should be 755. Another possibility is that the system could have the maximum number of files open that are allowed at any time. If the maximum number of files are open, try typing the command at a later time.

## LP Error Messages

---

Error Message	Description/Action
can't open acceptance status file	The mode on the file <code>/usr/spool/lp/qstatus</code> may not be correct. It should be 644. Another possibility is that the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open default destination file	Check the mode on the file <code>/usr/spool/lp/default</code> . The mode should be 644. If the mode is okay, it could be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't open file filename	The <i>filename</i> was incorrectly typed or you don't have the correct modes set. The mode should be at least 400 if you are the owner.
can't open output queue file	Check the mode on the file <code>/usr/spool/lp/outputq</code> . It should be 644. This error message could also be generated if the system has the maximum number of files open that are allowed at any one time. Try entering the command at a later time.
can't open printer status file	The mode on the file <code>/usr/spool/lp/pstatus</code> is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't open request directory directory name	The mode on the directory <code>/usr/spool/lp/request</code> is incorrect. The mode should be 655. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.

<b>Error Message</b>	<b>Description/Action</b>
can't open request file xx	The mode on the file <code>/usr/spool/lp/member/request/xx</code> is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the <b>lpmove</b> command at a later time.
can't open system default destination file	The mode on the file <code>/usr/spool/lp/default</code> is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command again at a later time.
can't open temporary output queue	The mode on the file <code>/usr/spool/lp/outputq</code> is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't proceed -- scheduler running	Many of the <b>lpadmin</b> command options cannot be executed while the scheduler is running. Stop the scheduler using the <b>lpshut</b> command and then try invoking the command again.
can't read current directory	The <b>lp</b> and <b>lpadmin</b> commands cannot read the directory containing the file to be printed. The directory name may be incorrect or you do not have read permission on that directory.
can't remove class file	The mode may be wrong on the <code>/usr/spool/lp/class</code> . It should be 755. The owner should be "lp," and the group should be "bin." Another possibility is the file in that directory may have the wrong mode. It should be 644.

## LP Error Messages

---

Error Message	Description/Action
can't remove printer	The mode may be wrong on the <b>/usr/spool/lp/member</b> directory. It should be 755, and the files in that directory should be 644. Both the directory and the files should be owned by "lp," and the group should be "bin."
can't remove request directory	The mode may be wrong on the <b>/usr/spool/lp/request</b> directory. It should be 755 and should be owned by "lp," and the group should be "bin." The directory may still have pending requests to be printed which will have to be removed before the directory can be removed.
can't set user id to LP Administrator's user id	The <b>lpsched</b> and <b>lpadmin</b> commands can only be used when you are logged in as "lp" or "root."
can't unlink old output queue	The <b>lpsched</b> program cannot remove the old output queue. You will have to remove it manually by using the command <b>rm/usr/spool/lp/outputq</b> .
can't write to xx	The <b>lpadmin</b> command cannot write to device <b>xx</b> . The mode is probably wrong on the <b>/dev/ttyxx</b> file. It should be 622 and owned by "lp."
cannot create temp file filename	The system may be out of free space on the <b>/usr</b> file system. Use the command <b>df /usr</b> to determine the number of free blocks. Several hundred blocks are required to insure that the system will perform correctly.
class "xx" has disappeared!	Class <b>xx</b> was probably removed since the scheduler was started. The system may be out of free space on the <b>/usr</b> file system. Use the command <b>df /usr</b> to find out. Use the <b>lpshut</b> command to stop the scheduler and restore the class from a backup.
class "xx" non-existent	The class <b>xx</b> may have been removed because the system is out of free space on the <b>/usr</b> file system. Use the command <b>df /usr</b> to find out how much free space is available. The class will probably have to be restored from a backup.

Error Message	Description/Action
class directory has disappeared!	The <code>/usr/spool/lp/class</code> directory has been removed. The system may be out of free space on <code>/usr</code> ; use the <code>df /usr</code> command to find out. The class directory contains all the data for each printer class. To restore this directory, get these files and directory from a backup.
corrupted member file	The <code>/usr/spool/lp/member</code> directory has a corrupted file in it. You should restore the directory from backup.
default destination "dest" non-existent	Either the default destination is not assigned or the printer <code>dest</code> has been removed. Use the <code>lpadmin</code> to set up a default destination or set <code>LPDEST</code> to the value of the destination.
destination "dest" has disappeared!	A destination printer, <code>dest</code> has been removed since <code>lpsched</code> was started. Use the <code>lpadmin</code> command to remove the printer.
destination "printer-name" is no longer accepting requests	The printer has been disabled using the <code>reject</code> command. The printer can be reenabled using the <code>accept</code> command.
destination dest non-existent	The destination printer you specified as an argument to the <code>accept</code> or <code>lpadmin</code> command is not a valid destination name, or it has been removed since the scheduler was started.
destination "printer-name" was already accepting requests	The destination printer was previously "enabled." Once a printer is accepting requests, issuing any more <code>accept</code> commands to it are ignored.
destination "printer-name" was already not accepting requests	A <code>reject</code> command was already sent to the printer. Use the <code>accept</code> command to allow the printer to start accepting requests again.



## LP Error Messages

---

Error Message	Description/Action
destination printer-name is not accepting requests move in progress ...	The printer has been disabled by the <b>reject</b> command, and requests are being moved from the disabled printer to another printer. The printer can be enabled again by the <b>accept</b> command.
destinations are identical	When using the <b>lpmove</b> command, you need to specify a printer to move the print requests from and a different printer to move the requests to.
disabled by scheduler: login terminal	The login terminal has been disabled by the LP scheduler. The printer can be reenabled by using the <b>enable</b> command.
error in printer request printer-id	<i>Printer-id</i> is the actual request identification number. The error was most likely due to an error in the printer. Check the printer and reset it if needed.
illegal keyletter "xx"	An invalid option, <i>xx</i> , was used. See the manual page for the correct options.
keyletters "-xx" and "-yy" are contradictory	This combination of options to the <b>lpadmin</b> program cannot be used together.
keyletter "xx" requires a value	The option <i>xx</i> requires an argument. For example, in the command line <b>lpadmin -m model</b> the argument to the <b>-m</b> option is the name of a model interface program.
keyletters -e, -i, and -m are mutually exclusive	These options to the <b>lpadmin</b> command cannot be used together. Refer to the manual page for more information on usage.
lp: xx	In this message the variable <i>xx</i> could be one of several arguments. Typically, it is a message telling you the default destination is not assigned.

<b>Error Message</b>	<b>Description/Action</b>
member directory has disappeared!	The <code>/usr/spool/lp/member</code> directory has been removed. The system is probably out of free disk space in the <code>/usr</code> file system. You need to clean up the <code>/usr</code> file system, and then install the LP commands or retrieve them from a backup.
model "xx" non-existent	The name that you are using for a model interface program is not a valid one. A list of valid models is in the <code>/usr/spool/lp/model</code> directory.
new printers require -v and either -e, -i, or -m	A printer must have an interface program, and this is specified by <code>-e</code> , <code>-i</code> , or <code>-m</code> options. The <code>-v</code> option specifies the device file for the printer. For more information on these options, refer to the <code>lpadmin</code> manual page.
no destinations specified	There are no destination printers specified. Use the <code>lpadmin</code> command to set one up.
no printers specified	There are no printers specified. Use the <code>lpadmin</code> command to set one up.
non-existent printer xx in PSTATUS	A printer with the name <code>xx</code> is in the <code>/usr/spool/lp/pstatus</code> file but no longer exists. The printer should be removed using the <code>lpadmin</code> command.
non-existent printer printer-name in class xx	The printer that you are trying to address in class <code>xx</code> has been removed from that class.
out of memory	Implies the system is in trouble. The message implies that there is not enough memory to contain the text to be printed.
printer "printer-name" already in class "xxP"	The printer you are trying to move to class <code>xx</code> is already in that class. You cannot move a printer to a class that it is already in.

## LP Error Messages

---

Error Message	Description/Action
printer "printer-name" has disappeared! or printer "printer-name" has disappeared	The printer has been removed, and the <b>enable</b> command cannot find it. The printer was most likely removed since the machine was rebooted or since the scheduler was started.
printer "printer-name" non-existent	<i>Printer-name</i> is the name of a printer that has been removed since the scheduler has been started. You must use the <b>lpadmin -xprinter-name</b> .
printer status entry for "printer-name" has disappeared	The <b>/usr/spool/lp/pstatus</b> file must have been corrupted. You will have to resubmit the printer request.
printer "printer-name" was not busy	The printer is not printing a request at this time. Either the request you wanted to cancel is finished printing or you have specified the wrong printer.
request "printer-id" non-existent	You are attempting to cancel a request that does not exist. You may have given the wrong printer name or wrong request id number, or the request may have finished printing.
request not accepted	The request was not accepted by <b>lp</b> . The scheduler may not be running. Use the <b>lpstat -t</b> command to find out more information.
requests still queued for "printer-name" -- use <b>lpmove</b>	<i>Printer-name</i> is the printer that still has requests waiting to get printed. You need to use the <b>lpmove</b> command to get those requests moved to another printer.
scheduler is still running -- can't proceed	You cannot perform this command while the scheduler is running. You will have to use the <b>lpshut</b> command first.
spool directory non-existent	The directory <b>/usr/spool</b> has been removed. You will have to use the <b>mkdir</b> command to restore the directory. This has probably removed some of the necessary LP files. You may have to reinstall the LP commands.

Error Message	Description/Action
standard input is empty	You specified an invalid file name either by incorrectly typing a name or by specifying a nonexistent file. Nothing will be printed on the printers from this request.
this command for use only by LP Administrators	This command is restricted to someone logged in as root or lp.
too many options for interface program	The <b>lp</b> command called the appropriate interface program with too many arguments. For more information on the options and arguments that can be used with the <b>lp</b> command, refer to the <b>lp</b> manual page.
unknown keyletter "xx" or unknown keyletter "-xx"	An invalid option was supplied to the <b>lp</b> or <b>lpadmin</b> commands. Refer to the manual pages in Appendix B for all the correct usages.
unknown option "xx"	This message is displayed in response to an invalid option supplied to the <b>disable</b> , <b>lpstat</b> , or <b>reject</b> commands. Refer to the manual pages in Appendix B for all the correct usages.
usage: disable [-c] [-r[reason]] printer	The syntax for the <b>disable</b> command is not correct. The valid options are: <b>-c</b> to cancel the currently printing request, and <b>-r</b> followed by the reason that you are disabling the printer.
usage: reject [-r[reason]] dest ...	The syntax for the <b>reject</b> command is not correct. The proper format is to specify the reason that the printer is not taking any more print requests and to identify the destination printer.
usage: accept dest	The syntax for the <b>accept</b> command is to specify a destination printer. You are setting up a printer to accept requests, and you did not specify what printer should accept requests.

## LP Error Messages

---

Error Message	Description/Action
usage: enable printer	The syntax for the <b>enable</b> program is to specify a destination printer.
usage: cancel id .... printer ...	The syntax for the <b>cancel</b> command is not correct. The proper format is to specify the request identification number or the printer name.
usages: lpadmin -pprinter [-vdevice] [-cclass] [-rclass] [-eprinter ! -iinterface ! -mmodel] [-h ! -l] -or- lpadmin -d[destination] -or- lpadmin -xdestination	The correct syntax for the <b>lpadmin</b> command is to specify at least one of the options mentioned above.
your printer request printer-id was canceled by user	The printer request did not finish printing because another <i>user</i> cancelled it. Typically, you will get this message in your mail. One reason a person may cancel a request other than their own is because the request is not printing correctly.

---

## Basic Networking Utilities Error Messages

This section lists the error messages associated with Basic Networking Utilities. There are two types of error messages. ASSERT errors are recorded in the `/usr/spool/uucp/.Admin/errors` file. STATUS errors are recorded in individual machine files found in the `/usr/spool/uucp/.Status` directory.

---

## BNU 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 the text listed below. In most cases, these errors are the result of file system problems. The "errno" (when present) should be used to investigate the problem. If "errno" is present in a message, it is shown as () in the following list.

<b>Error Message</b>	<b>Description/Action</b>
CAN'T OPEN	An open() or fopen() failed.
CAN'T WRITE	A write(), fwrite(), fprintf(), etc., failed.
CAN'T READ	A read(), fgets(), etc., failed.
CAN'T CREATE	A create() call failed.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T LOCK	An attempt to make a LCK (lock) file failed. In some cases, this is a fatal error.
CAN'T STAT	A stat() call failed.
CAN'T CHMOD	A chmod() call failed.
CAN'T LINK	A link() call failed.
CAN'T CHDIR	A chdir() call failed.
CAN'T UNLINK	An unlink() call failed.
WRONG ROLE	This is an internal logic problem.

<b>Error Message</b>	<b>Description/Action</b>
CAN'T MOVE TO CORRUPTDIR	An attempt to move some bad C. or X. files to the <code>/usr/spool/uucp/.Corrupt</code> directory failed. The directory is probably missing or has wrong modes or owner.
CAN'T CLOSE	A <code>close()</code> or <code>fclose()</code> call failed.
FILE EXISTS	The creation of a C. or D. file is attempted, but the file exists. This occurs when there is a problem with the sequence file access. Usually indicates a software error.
No uucp server	A <code>tcp/ip</code> call is attempted, but there is no server for UUCP.
BAD UID	The uid cannot be found in the <code>/etc/passwd</code> file. The file system is in trouble, or the <code>/etc/passwd</code> file is inconsistent.
BAD LOGIN_UID	Same as previous.
ULIMIT TOO SMALL	The ulimit for the current user process is too small. File transfers may fail, so transfer is not attempted.
BAD LINE	There is a bad line in the <b>Devices</b> file; there are not enough arguments on one or more lines.
FSTAT FAILED IN EWRDATA	There is something wrong with the ethernet media.
SYSLST OVERFLOW	An internal table in <code>gename.c</code> overflowed. A <code>big/strange</code> request was attempted. Contact your AT&T Account Representative or Authorized Dealer.
TOO MANY SAVED C FILES	Same as previous.



## BNU ASSERT Error Messages

---

<b>Error Message</b>	<b>Description/Action</b>
RETURN FROM fixline ioctl	An ioctl, which should never fail, failed. There is a system driver problem.
BAD SPEED	A bad line speed appears in the <b>Devices/Systems</b> files (Class field).
PERMISSIONS file: BAD OPTION	There is a bad line or option in the <b>Permissions</b> file. Fix it immediately!
PKCGET READ	The remote machine probably hung up. No action need be taken.
PKXSTART	The remote machine aborted in a non-recoverable way. This can generally be ignored.
SYSTAT OPEN FAIL	There is a problem with the modes of <b>/usr/lib/uucp/.Status</b> , or there is a file with bad modes in the directory.
TOO MANY LOCKS	There is an internal problem! Contact your AT&T Account Representative or Authorized Dealer.
XMV ERROR	There is a problem with some file or directory. It is likely the spool directory, since the modes of the destinations were to have been checked before this process was attempted.
CAN'T FORK	An attempt to fork and exec failed. The current job should not be lost, but will be attempted later ( <b>uuxqt</b> ). No action need be taken.

---

## BNU STATUS Error Messages

Status error messages are messages that are stored in the `/usr/spool/uucp/.Status` directory. This directory contains a separate file for each remote machine that your computer attempts to communicate with. These individual machine files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common error messages that may appear in these files.

Error Message	Description/Action
OK	Things are OK.
NO DEVICES AVAILABLE	There is currently no device available for the call. Check to see that there is a valid device in the <b>Devices</b> file for the particular system. Check the <b>Systems</b> file for the device to be used to call the system.
WRONG TIME TO CALL	A call was placed to the system at a time other than what is specified in the <b>Systems</b> file.
TALKING	Self-explanatory.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the Dialer-Token-Pairs script.
CONVERSATION FAILED	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.
DIAL FAILED	The remote machine never answered. It could be a bad dialer or the wrong phone number.

## BNU STATUS Error Messages

---

<b>Error Message</b>	<b>Description/Action</b>
BAD LOGIN/MACHINE COMBINATION	The machine called us with a login/machine name that does not agree with the <b>Permissions</b> file. This could be an attempt to masquerade!
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.
ASSERT ERROR	An ASSERT error occurred. Check the <b>/usr/spool/uucp/.Admin/errors</b> file for the error message and refer to the section <i>ASSERT Error Messages</i> .
SYSTEM NOT IN Systems	The system is not in the <b>Systems</b> file.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the <b>Systems</b> and <b>Devices</b> files.
DEVICE FAILED	The open of the device failed.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.
CALLBACK REQUIRED	The called machine requires that it calls your computer.
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your computer. They could be trying to call your machine. If they have an older version of Basic Networking, the process that was talking to your machine may have failed, leaving the LCK file. If they have the new version of Basic Networking, and they are not communicating with your computer, then the process that has a LCK file is hung.

---

## BNU STATUS Error Messages

<b>Error Message</b>	<b>Description/Action</b>
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your computer in its <b>Systems</b> file.
REMOTE REJECT AFTER LOGIN	The login used by your computer to log in does not agree with what the remote machine was expecting.
REMOTE REJECT, UNKNOWN MESSAGE	The remote machine rejected the communication with your computer for an unknown reason. The remote machine may not be running a standard version of Basic Networking.
STARTUP FAILED	Login succeeded, but initial handshake failed.
CALLER SCRIPT FAILED	This is usually the same as "DIAL FAILED." However, if it occurs often, suspect the caller script in the <b>dialers</b> file. Use <b>uutry</b> to check.



---

## Glossary

<b>address</b>	a number, label, or name that indicates the location of information in the computer's <i>memory</i> .
<b>advertise</b>	a means of making <i>resources</i> available from a local <i>host</i> to other <i>hosts</i> in a <i>Remote File Sharing</i> environment.
<b>a.out</b>	the default name of a freshly compiled <i>object file</i> , pronounced 'A-dot-out'; historically a.out signified assembler output.
<b>archive</b>	<ol style="list-style-type: none"><li>1. a collection of data gathered from several <i>files</i> into one file.</li><li>2. especially, such a collection gathered by <b>ar(1)</b> for use as a library.</li></ol>
<b>automatic calling unit</b>	a hardware <i>device</i> used to dial stored telephone numbers; allows the system to contact another system over phone lines without manual intervention.
<b>bad block</b>	a section of a storage medium which cannot store data reliably.
<b>block</b>	the basic unit of <i>buffering</i> in the <i>kernel</i> , 1024 bytes; see <i>indirect</i> , <i>logical</i> , and <i>physical blocks</i> .
<b>block device</b>	a <i>device</i> upon which a <i>file system</i> [1] can be <i>mounted</i> , typically a permanent storage device such as a tape or disk drive, so called because data transfers to the device occur by <i>blocks</i> ; cf. <i>character device</i> .
<b>boot</b>	to start the operating system, so called because the <i>kernel</i> must bootstrap itself from secondary storage into an empty machine. No <i>login</i> [3] or <i>process</i> persists across a boot. <b>boot block</b> the first block of a <i>file system</i> [1], which is reserved for a booting program.

## Glossary

---

<b>boot program</b>	loads the <i>operating system</i> into <i>core</i> .
<b>buffer</b>	1. a staging area for input output where arbitrary-length transactions are collected into convenient units for system operations; the <i>file system</i> [3] uses buffers, as does <i>stdio</i> . 2. to use buffers.
<b>buffer pool</b>	a region of store available to the <i>file system</i> [3] for holding <i>blocks</i> ; all but <i>raw</i> [2] input-output for <i>block devices</i> goes through the buffer pool so read and write operations may be independent of device blocks.
<b>cartridge tape</b>	a storage medium that consists of a magnetic tape wound on spools housed in a plastic container.
<b>character device</b>	a <i>device</i> upon which a <i>file system</i> [1] cannot be <i>mounted</i> such as a terminal or the <i>null device</i> .
<b>child process</b>	see <i>fork</i> .
<b>client</b>	a <i>host</i> that has <i>mounted</i> an <i>advertised resource</i> from another <i>host</i> in a <i>Remote File Sharing</i> environment.
<b>command</b>	1. an instruction to the <i>shell</i> , usually to run a <i>program</i> [1] as a <i>child process</i> . 2. by extension, any <i>executable file</i> , especially a <i>utility program</i> .
<b>command file</b>	same as <i>shell script</i> .
<b>configuration</b>	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.
<b>controller</b>	a <i>device</i> that directs the transmission of data over the data links of a <i>network</i> .
<b>core file</b>	a <i>core image</i> of a terminated <i>process</i> saved for debugging; a core file is created under the name 'core' in the <i>current directory</i> of the process.

<b>core image</b>	a copy of all the <i>segments</i> of a running or terminated program; the copy may exist in main store, in the <i>swap area</i> , or in a <i>core file</i> .
<b>crash</b>	If a hardware or software <i>error</i> 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 <i>processes</i> , respond to requests for system functions, or when the electrical power is unstable.
<b>cron</b>	a command which creates a daemon that invokes commands at specified dates and times.
<b>cylinder</b>	the set of all <i>tracks</i> on a <i>disk</i> which are the same distance from the axis about which the disk rotates.
<b>daemon</b>	a <i>background</i> process, often perpetual, that performs a system-wide public function, e.g., <b>calen-</b> <b>dar</b> (1) and <b>cron</b> (8); the affected spelling is an ancient legacy.
<b>destination</b>	the remote system that will ultimately receive a <i>file</i> transferred over a <i>network</i> .
<b>device</b>	1. a <i>file</i> [2] that is not a <i>plain file</i> or a <i>directory</i> , such as a tape drive, or the <i>null device</i> ; a <i>special file</i> . 2. a physical input output unit.
<b>diagnostic</b>	a message printed at your terminal that identifies and isolates <i>program</i> errors.
<b>directory</b>	a <i>file</i> that comprises a catalog of <i>filenames</i> [2]; the organizing principle of the <i>file system</i> [2], a <i>directory</i> consists of <i>entries</i> which specify further <i>files</i> (sense 2, including <i>directories</i> ), and constitutes a node of the <i>directory tree</i> .
<b>directory entry, entry</b>	1. an association of a name with an <i>inode number</i> appearing as an element of a <i>directory</i> . 2. the name part of such an association.



## Glossary

---

<b>directory hierarchy</b>	the tree of all <i>directories</i> , in which each is reachable from the <i>root</i> via a chain of <i>subdirectories</i> .
<b>directory tree</b>	same as <i>directory hierarchy</i> .
<b>disk</b>	a platter coated with magnetic material on which data can be stored.
<b>diskette</b>	a magnetic storage medium which is smaller and more flexible than a hard <i>disk</i> .
<b>domain</b>	a logical grouping of <i>hosts</i> in a <i>Remote File Sharing</i> environment. Each <i>host</i> in a <i>domain</i> relies on the same <i>domain name server</i> (s) for certain <i>resource</i> sharing and security services. Each <i>domain</i> has one <i>primary</i> and zero or more <i>secondary domain name servers</i> .
<b>domain name server</b>	a computer that creates and maintains the following information for <i>hosts</i> in a <i>Remote File Sharing</i> domain: <i>advertised resources</i> , <i>host</i> names and passwords, names and addresses for name servers of other domains (optional), <i>host</i> user and group information used for <i>ID mapping</i> (optional).
<b>drive</b>	the hardware device that holds magnetic disks, diskettes, and tapes while they are in use.
<b>dump</b>	a copy of the <i>core image</i> of the operating system.
<b>environment</b>	1. a set of strings, distinct from the <i>arguments</i> , made available to a <i>process</i> when it <i>executes</i> [2] a <i>file</i> ; the environment is usually inherited across <i>exec(2)</i> operations. 2. a specific environment [2] maintained by the <i>shell</i> . 3. a nebulously identified way of doing things, as in 'interactive environment': a deprecated usage, not always expunged from these manuals.
<b>error</b>	occurs when a hardware or software condition prevents the successful <i>execution</i> of a system or a user <i>process</i> .

<b>error message</b>	a message sent from the system to the <i>system console</i> when an <i>error</i> occurs.
<b>exec</b>	a system call which allows the user to request the execution of another program.
<b>executable file</b>	1. an <i>object file</i> that is ready to be copied into the <i>address space</i> of a <i>process</i> to run as the code of that process. 2. a file that has <i>execute permission</i> , either an <i>executable file</i> [1] or a <i>shell script</i> .
<b>execute</b>	1. informally, to run a <i>program</i> . 2. to replace the <i>text segment</i> and <i>data segments</i> of a <i>process</i> with a given <i>program</i> [1].
<b>FIFO</b>	a named permanent <i>pipe</i> which allows two unrelated <i>processes</i> to exchange information using a pipe connection.
<b>file</b>	1. in general, a potential source of input or destination for output. 2. most specifically, an <i>inode</i> and/or associated contents, i.e., a <i>plain file</i> , a <i>special file</i> , or a <i>directory</i> . 3. a <i>directory entry</i> ; several directory entries may name the same file [2]. 4. most loosely, a <i>plain file</i> .
<b>file descriptor</b>	a conventional integer quantity that designates an <i>open file</i> .
<b>filename</b>	1. a <i>pathname</i> . 2. the last component name in a path name.
<b>file system</b>	1. a collection of <i>files</i> that can be <i>mounted</i> on a block <i>special file</i> ; each file of a file system appears exactly once in the <i>i-list</i> of the file system and is accessible via some <i>path</i> from the <i>root</i> directory of the file system. 2. the collection of all <i>files</i> on a computer. 3. the part of the kernel that deals with file systems [1].
<b>filter</b>	a <i>program</i> [1] that reads from the <i>standard input</i> and writes on the <i>standard output</i> , so called because it can be used as a data-transformer in a <i>pipeline</i> .

## Glossary

---

<b>floppy key</b>	a copy of the default <i>firmware</i> password for your computer on a <i>diskette</i> . It may be used to reset the password to its original value.
<b>flush</b>	to empty a <i>buffer</i> , for example to throw away unwanted input output upon <i>interrupt</i> or to release output from the clutches of <i>stdio</i> .
<b>fork</b>	to split one <i>process</i> into two, the <b>parent process</b> and <b>child process</b> , with separate, but initially identical, <i>text</i> , <i>data</i> , and <i>stack segments</i> .
<b>formatting</b>	the process of imposing an addressing scheme on a <i>disk</i> . This includes the establishment of a <i>VTOC</i> , and the mapping of both sides of the disk into <i>tracks</i> and <i>sectors</i> .
<b>free list</b>	in a <i>file system</i> [1], the list of <i>blocks</i> that are not occupied by data.
<b>getty</b>	one of a series of <i>processes</i> which connect the user to the UNIX system. <i>getty</i> is invoked by <i>init</i> , and in turn invokes <i>login</i> .
<b>group</b>	1. a set of <i>permissions</i> alternative to <i>owner</i> permissions for access to a <i>file</i> . 2. a set of <i>userids</i> that may assume the privileges of a group [1]. 3. the <i>groupid</i> of a file.
<b>groupid</b>	an integer value, usually associated with one or more <i>login names</i> ; as the <i>userid</i> of a process becomes the <i>owner</i> of files <i>created</i> by the process, so the <i>groupid</i> of a process becomes the <i>group</i> [3] of such files.
<b>hole</b>	a gap in a <i>plain file</i> caused by <i>seeking</i> while writing; <i>read(2)</i> takes data in holes to be zero; a <i>block</i> in a hole occupies no space in its <i>file system</i> .
<b>host</b>	a computer that is configured to share <i>resources</i> in a <i>Remote File Sharing</i> environment.
<b>ID mapping</b>	a means of setting the permissions that each remote user and group will have for a <i>host's advertised resources</i> in a <i>Remote File Sharing</i> environment.

---

<b>i-list</b>	the index to a <i>file system</i> [1] listing all the <i>inodes</i> of the file system; cf. <i>inode number</i> .
<b>indirect blocks</b>	data blocks that are not directly referenced by a <i>inode</i> (because the file is larger than ten 1024-byte blocks); the <i>inode</i> has 3 <i>addresses</i> that indirectly reference (by a cascade of pointers) some 2,114,114 data blocks (an extremely large potential <i>file size</i> ). the <i>inode</i> 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!
<b>init</b>	a general <i>process</i> spawner which is invoked as the last step in the <i>boot</i> procedure; it regularly checks a table that defines what processes should run at what <i>run level</i> .
<b>inode</b>	an element of a <i>file system</i> [1]; an <i>inode</i> specifies all properties of a particular <i>file</i> [2] and locates the file's contents, if any.
<b>inode number, i-number</b>	the position of an <i>inode</i> in the <i>i-list</i> of a <i>file system</i> [1].
<b>instruction</b>	see <i>address</i> .
<b>integrity</b>	in a <i>file system</i> , the quality of being without errors due to <i>bad blocks</i> .
<b>interface programs</b>	<i>shell scripts</i> furnished with the LP <i>spooling</i> software which interface between the user and the printer.
<b>interrupt</b>	1. a <i>signal</i> that normally terminates a <i>process</i> , caused by a break or an interrupt character. 2. a signal generated by a hardware condition or a peripheral <i>device</i> . 3. loosely, any <i>signal</i> .
<b>IPC</b>	an acronym for interprocess communication.

## Glossary

---

<b>kernel</b>	the UNIX system proper; resident code that implements the <i>system calls</i> .
<b>kernel address space</b>	a portion of memory used for data and code addressable only by the <i>kernel</i> .
<b>line discipline</b>	a module to handle protocol or data conversion for a <i>stream</i> [2]. A line discipline, unlike a <i>filter</i> , is part of the <i>kernel</i> .
<b>link</b>	1. to add an entry for an existing <i>file</i> to a directory; converse of <i>unlink</i> . 2. by extension, a <i>directory entry</i> . 3. loosely, any but one putatively primary directory entry for a given <i>inode</i> ; either linked [1] or a <i>symbolic link</i> .
<b>link count</b>	the number of <i>directory entries</i> that pertain to an <i>inode</i> ; a <i>file</i> ceases to exist when its link count becomes zero and it is not <i>open</i> .
<b>load device</b>	designates the physical <i>device</i> from which a program will be loaded into main <i>memory</i> .
<b>log files</b>	contain records of transactions that occur on the system; software that <i>spools</i> , for example, generates various log files.
<b>logical block</b>	a unit of data as it is handled by the software; the UNIX system handles data in 1024-byte logical blocks.
<b>login</b>	1. the <i>program</i> that controls logging in. 2. the act of <i>logging in</i> . 3. by extension, the computing session that follows a login [2].
<b>memory</b>	1. same as <i>memory image</i> . 2. physical memory represents the available space in main memory; <i>programs</i> are either <i>swapped</i> or <i>paged</i> into physical memory for <i>execution</i> . 3. virtual memory management techniques permit <i>programs</i> to treat <i>disk</i> storage as an extension of main memory.
<b>memory image</b>	same as <i>core image</i> .

---

<b>mode, file mode</b>	the <i>permissions</i> of a <i>file</i> ; colloquially referred to by a 3-digit octal number, e.g., 'a 755 file'; see <i>chmod(1)</i> .
<b>mount</b>	to extend the <i>directory hierarchy</i> by associating the <i>root</i> of a <i>file system</i> [1] with a <i>directory entry</i> in an already mounted file system; converse is <i>unmount</i> , spelled 'umount'.
<b>namelist</b>	same as <i>symbol table</i> .
<b>network</b>	the hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals.
<b>networking</b>	for computer systems, means sending data from one system to another over some communications medium (coaxial cable, phone lines, etc.). Common networking services include <i>file transfer</i> , <i>remote login</i> , <i>remote execution</i> .
<b>node name</b>	an up-to-six character name for the system; used as the official name of the machine in a <i>network</i> . The node name resides in the <i>NODE</i> parameter.
<b>null device</b>	a <i>device</i> [1] that always yields <i>end of file</i> on reading and discards all data on writing.
<b>object file</b>	a <i>file</i> 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 <i>executable file</i> [1].
<b>operating system</b>	the <i>program</i> for managing the resources of the computer. It takes care of such things as input/output procedures, process scheduling, and the file system, removing this burden from user programs.
<b>open file</b>	1. the destination for input or output obtained by <i>opening a file</i> or creating a <i>pipe</i> ; a <i>file descriptor</i> ; open files are shared across <i>forks</i> and persist across <i>executes</i> [2]. 2. loosely, a file that has been

## Glossary

---

	opened, however an open file [1] need not exist in a <i>file system</i> [1], and a file [2] may be the destination of several <i>open files</i> simultaneously.
<b>other</b>	<ol style="list-style-type: none"><li>1. a set of <i>permissions</i> regulating access to a <i>file</i> by processes with <i>userid</i> different from the <i>owner</i> and <i>groupid</i> different from the <i>group</i> of the file.</li><li>2. the customary name of the default <i>group</i> [2] assigned upon <i>login</i>.</li></ol>
<b>owner</b>	the <i>userid</i> of the <i>process</i> that created a <i>file</i> ; the owner has distinctive <i>permissions</i> for a file.
<b>page</b>	a fixed length, 1024-byte block that has a virtual <i>address</i> , and that can be transferred between main and secondary storage.
<b>paging</b>	the process by which <i>programs</i> are truncated into <i>pages</i> and transferred between main and secondary storage by the virtual handler (or paging <i>daemon</i> ).
<b>parent process</b>	see <i>fork</i> .
<b>partitions</b>	units of storage space on disk.
<b>path, pathname</b>	a chain of names designating a <i>file</i> ; a <b>relative pathname</b> leads from the current directory, for example, a path to <i>directory A</i> , thence to <i>directory B</i> , thence to <i>file C</i> is denoted A/B/C; a <b>full pathname</b> begins at the <i>root</i> , indicated by an initial '/', as in /A/B/C.
<b>permission</b>	a right to access a <i>file</i> in a particular way; read, write, execute (or look up in, if a directory); permissions are granted separately to <i>owner</i> , <i>group</i> , and <i>others</i> . <b>permission bit</b> a permission, so called because each permission is encoded into one bit in an <i>inode</i> .
<b>physical block</b>	a unit of data as it is actually stored and manipulated; the UNIX system handles data in 1024-byte physical blocks.

---

<b>physical</b>	<i>see memory.</i>
<b>pipe</b>	a direct stream connection between <i>processes</i> , whereby data written on an <i>open file</i> in one process becomes available for reading in another.
<b>pipeline</b>	a sequence of <i>programs</i> [1] connected by <i>pipes</i> .
<b>polling</b>	the interrogation of <i>devices</i> by the <i>operating system</i> to avoid contention, determine operation status, or ascertain readiness to send or receive data.
<b>ports</b>	the point of physical connection between a peripheral <i>device</i> (such as a terminal or a printer) and the device <i>controller</i> (ports board), which is part of the computer hardware.
<b>primary name server</b>	the computer on which administration for a <i>Remote File Sharing domain</i> is performed.
<b>process</b>	a connected sequence of computation; a process is characterized by a <i>core image</i> with instruction location counter, <i>current directory</i> , a set of <i>open files</i> , <i>control terminal</i> , <i>userid</i> , and <i>groupid</i> .
<b>process id</b>	an integer that identifies a <i>process</i> .
<b>process number</b>	same as <i>process id</i> .
<b>profile</b>	1. an optional <i>shell script</i> , '.profile', conventionally used by the <i>shell</i> upon <i>logging in</i> to establish the <i>environment</i> [3] and other working conditions customary to a particular user. 2. to collect a histogram of values of the instruction location counter of a <i>process</i> .
<b>program</b>	1. an <i>executable file</i> . 2. a <i>process</i> . 3. all the usual meanings.
<b>queue</b>	a line or list formed by items in a system waiting for service.
<b>raw device</b>	a <i>block device</i> , read and write operations to which are not <i>buffered</i> , and are synchronized to natural records of the physical <i>device</i> .



## Glossary

---

<b>reboot</b>	same as <i>boot</i> .
<b>region</b>	a group of machine <i>addresses</i> that refer to a base address.
<b>release</b>	a distribution of fixes or new functions for an existing software product.
<b>Remote File Sharing</b>	a software utilities package that enables computers to share <i>resources</i> across a network.
<b>resource</b>	a directory that is <i>advertised</i> in a <i>Remote File Sharing</i> environment. When a <i>resource</i> is <i>mounted</i> on a <i>client</i> , the contents of the directory (files, devices, and named pipes) and any of its subdirectories are potentially available to users on the <i>client</i> .
<b>re-tension</b>	the process of rewinding the tape in a <i>cartridge tape device</i> to make sure it is at the correct tautness for accurate recording of data.
<b>root</b>	<ol style="list-style-type: none"><li>1. a distinguished directory that constitutes the origin of the <i>directory hierarchy</i> in a <i>file system</i> [1].</li><li>2. specifically, the origin for the <i>file system</i> [2], with the conventional <i>pathname</i> <i>'/'</i>.</li><li>3. the origin of the <i>directory hierarchy</i> in a <i>file system</i> [1].</li></ol>
<b>rotational gap</b>	the gap between the actual <i>disk</i> locations of blocks of data belonging to the same <i>file</i> ; 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.
<b>run level</b>	a software <i>configuration</i> of the system which allows a particular group of <i>processes</i> to exist.
<b>schedule</b>	to assign resources— main store and CPU time—to <i>processes</i> .
<b>scheduler</b>	a permanent <i>process</i> , with <i>process number</i> 1, and associated <i>kernel</i> facilities that does scheduling.

---

<b>search path</b>	in the <i>shell</i> , a list of <i>pathnames of directories</i> that determines the meaning of a <i>command</i> ; the command name is prefixed with members of the search path in turn until a path name of an <i>executable file</i> [2] results; the search path is given by the shell variable PATH.
<b>secondary name server</b>	a <i>host</i> that is configured to take over <i>domain name server</i> responsibilities temporarily in case the <i>primary name server</i> goes down.
<b>section, sector</b>	A 512-byte portion of a <i>track</i> which can be accessed by the magnetic disk heads in the course of a predetermined <i>rotational</i> displacement of the storage device.
<b>segment</b>	a contiguous range of the address space of a <i>process</i> with consistent store access capabilities; the four segments are (i) the <b>text segment</b> , occupied by executable code, (ii) the <b>data segment</b> , occupied by <i>static</i> data that is specifically initialized, (iii) the <b>bss segment</b> , occupied by static data that is initialed by default to zero values, and (iv) the <b>stack segment</b> , occupied by <i>automatic</i> data; see <i>stack</i> ; sometimes (ii), (iii), and (iv) are collectively called data segments.
<b>semaphore</b>	an IPC facility which allows two or more processes to be synchronized.
<b>server</b>	a <i>host</i> that is actively sharing one of its <i>advertised resources</i> with another <i>host</i> in a <i>Remote File Sharing</i> environment.
<b>set userid</b>	a special <i>permission</i> for an <i>executable file</i> [1] that causes a <i>process</i> executing it to have the access rights of the <i>owner</i> of the file; the owner's <i>userid</i> becomes the <b>effective userid</b> of the process, distinguished from the <b>real userid</b> under which the process began.
<b>set userid bit</b>	the associated <i>permission bit</i> .

## Glossary

---

<b>shared memory</b>	an IPC facility which allows two or more processes to share the same data space.
<b>shell</b>	1. the program <i>sh(1)</i> , which causes other programs to be executed on <i>command</i> ; the shell is usually started on a user's behalf when the user <i>logs in</i> . 2. by analogy, any program started upon logging in.
<b>shell script</b>	an executable <i>file of commands</i> taken as input to the <i>shell</i> .
<b>signal</b>	an exceptional occurrence that causes a <i>process</i> to terminate or divert from the normal flow of control; see <i>interrupt</i> , <i>trap</i> .
<b>single-user</b>	a state of the operating system in which only one user is supported.
<b>source file</b>	1. the uncompiled version of a <i>program</i> . 2. generally, the unprocessed version of a <i>file</i> .
<b>special file</b>	an <i>inode</i> that designates a <i>device</i> , further categorized as either (i) a <b>block special file</b> describing a <i>block device</i> , or (ii) a <b>character special file</b> describing a <i>character device</i> .
<b>spool</b>	to collect and serialize output from multiple <i>processes</i> competing for a single output service.
<b>spool area</b>	a <i>directory</i> in which a spooler collects work.
<b>spooler</b>	a <i>daemon</i> that spools.
<b>stack</b>	a <i>segment</i> of the <i>address space</i> into which <i>automatic</i> data and subroutine linkage information is allocated in last-in-first-out fashion; the stack occupies the largest data addresses and grows downward towards <i>static</i> data.
<b>standard error</b>	one of three files described below under <i>standard output</i> .
<b>standard input</b>	the second of three files described below under <i>standard output</i> .

---

<b>standard output</b>	<i>open files</i> , customarily available when a <i>process</i> begins, with <i>file descriptors</i> 0, 1, 2, and <i>stdio</i> 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.
<b>startup</b>	same as <i>boot</i>
<b>sticky bit</b>	a <i>permission</i> flag that identifies a file as a <i>sticky file</i> .
<b>sticky file</b>	a special <i>permission</i> for a <i>shared text</i> file that causes a copy of the <i>text segment</i> to be retained in the <i>swap area</i> to improve system response.
<b>super block</b>	the second <i>block</i> in a <i>file system</i> [1], which describes the allocation of space in the file system; cf. <i>boot block</i> .  <i>userid</i> 0, which can access any <i>file</i> regardless of <i>permissions</i> and can perform certain privileged <i>system calls</i> , e.g., setting the clock.
<b>swap</b>	to move the <i>core image</i> of an executing program between main and secondary storage to make room for other <i>processes</i> .
<b>swap area</b>	the part of secondary store to which <i>core images</i> are <i>swapped</i> ; the swap area is disjointed from the <i>file system</i> .
<b>symbolic link</b>	an <i>inode</i> that contains the <i>pathname</i> of another. References to the symbolic link become references to the named <i>inode</i> .
<b>symbol table</b>	information in an <i>object file</i> about the names of data and functions in that file; the symbol table and <i>address</i> relocation information are used by the link editor to compile <i>object files</i> and by debuggers.

<b>System Administration</b>	when capitalized, refers to the package of screens and interactive prompts, invoked through the <b>sysadm(1)</b> command, that helps you accomplish most system administration tasks.
<b>system calls</b>	<ol style="list-style-type: none"><li>1. the set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated.</li><li>2. the system primitives invoked by user <i>processes</i> for system-dependent functions, such as I/O, process creation, etc.</li></ol>
<b>system console</b>	the directly connected terminal used for communication between the operator and the computer.
<b>system name</b>	an up-to-six character name for the system; resides in the SYS parameter.
<b>table</b>	an array of data each item of which may be uniquely identified by means of one or more arguments.
<b>text file,</b>	a <i>file</i> , the bytes of which are understood to be in ASCII code.
<b>track</b>	an addressable ring of <i>sections</i> on a <i>disk</i> or <i>diskette</i> ; each disk or diskette has a predefined number of concentric tracks, which allows the disk head to properly access <i>sections</i> of data.
<b>trap</b>	a method of detecting and interpreting certain hardware and software conditions via software; a trap is set to catch a <i>signal</i> (or <i>interrupt</i> ), and determine what course of action to take.
<b>tunable parameters</b>	variables used to set the sizes and thresholds of the various control structures of the <i>operating system</i> .
<b>tuning</b>	<ol style="list-style-type: none"><li>1. modifying the <i>tunable parameters</i> so as to improve system performance.</li><li>2. the reconfiguration of the <i>operating system</i> to incorporate the modifications into <i>executable</i> version of the system.</li></ol>

<b>userid</b>	an integer value, usually associated with a <i>login name</i> ; the <i>userid</i> of a <i>process</i> becomes the <i>owner</i> of files <i>created</i> by the process and descendent ( <i>forked</i> ) processes.
<b>utility, utility program</b>	a standard, generally useful, permanently available <i>program</i> .
<b>version</b>	a separate <i>program</i> product, based on an existing one, but containing significant new code or new functions.
<b>virtual memory</b>	see <i>memory</i> .
<b>VTOC</b>	Volume Table Of Contents is the section of a disk which shows how the <i>partitions</i> on the <i>disk</i> are allocated.



---

# Index

- /etc/getty ... 3:11, 3:15, 8:5-6, B:10
- /etc/init ... B:27
- /etc/inittab ... P9:8
- /etc/passwd ... 7:14
- /etc/profile ... 2:6, B:15
- /etc/rc0 ... 3:11, B:10
- /etc/shutdown ... B:3, B:10
- /unix ... 6:13
- /usr ... P3:13, P5:18, P5:21, P5:32, 5:29, 5:31, 10:20, C:34
- /usr/lib/accept ... P7:5, 7:10
- /usr/lib/lpadmin ... P7:7
- /usr/news ... P10:22, B:4
- /usr/spool/cron/crontabs ... B:4
- /usr2 ... P5:18, P5:20-21, 10:20
- active processes ... 10:40
- ACU ... P9:8, 9:2, 9:9-12, 9:18-19
- administrative commands ... 1:1, 1:7, 7:3, 10:33
- administrative directories ... 3:1
- administrative logins... P1:9, P9:16, 10:27
- adv ... P10:9, P10:22, 10:8, 10:11, 10:50, 10:63
- automatic call units ... P9:22
- backup ... P3:8, P3:13, P5:18, P5:20-21, P5:26, P5:28, P5:30-32, 3:3, 3:5, 4:2, 4:9-10, 4:13, 5:9, 5:31-35, 5:38, B:2, B:24, C:8-9
- bad block ... 4:1, 4:13-18, 5:44, 5:46, 5:56, C:2
- block count ... 5:31, 5:61
- boot program ... P1:12, P3:10, P6:2
- cartridge tape ... P3:13, P4:1, P5:1, P5:18, P5:21, 4:1-4, 4:6, 5:35, 6:8
- cd ... P1:13, P3:6, P3:15, P5:9, P5:32, P10:17, 5:24, 6:13, B:26
- character codes ... 1:4
- character device ... 4:6-7
- chmod ... P7:6, 9:32, B:24
- cleanup phase ... 5:63
- configurable parameters ... 6:55
- configuration ... P1:2, P3:10-11, P4:1, P5:1, P6:1, P9:3, P10:12, 3:2, 3:4, 3:8, 3:11, 4:4, 4:10, 6:2, 6:4, 6:42, 6:52, 6:54, B:12
- console ... P1:2-3, P1:12, P3:2, P3:5, P3:8, P5:29, P5:33, P8:10, 3:8, 3:12, 3:15, 4:10, 4:16, 5:27, 5:35, 6:2, 8:6, 10:18
- console printer ... P1:2
- cpio ... P1:14-15, P10:17, 5:24, 5:36, 6:7
- cpu ... P3:7
- crash ... 6:55, C:13
- cron ... 3:19, B:32
- crontab ... P9:19-20, 3:7, 6:14, 7:20, B:36-37
- cu ... P9:10, P9:14-15, P9:22, P9:30, P9:34, 9:31
- daemons ... P3:4, 3:20, 6:55, 9:5
- default configuration ... 6:2
- default destination ... 7:5
- default parameters ... 6:4
- default shell ... P2:9
- device file ... 5:41
- device filter ... 7:14
- device type ... P9:8, P9:35, 9:17-18, 9:38
- disk bottleneck ... 6:10
- disk driver ... 4:14
- disk space ... P5:1, P5:6, P5:11, 5:2, 5:26-29, 10:10, 10:34, C:37
- disk usage ... 10:10
- DNEWS ... P10:22



## Index

---

- domain ... **P10:2-3, P10:5-6, P10:9-8, P10:10, P10:12-13, P10:15-17, P10:19-20, P10:22, P10:24, C:24, G:4**
- du ... **P1:14, 5:31**
- duplicating disks ... **4:11**
- encryption ... **1:2, 1:7**
- environment strings ... **2:8**
- environment variables ... **2:7, 2:9**
- error messages ... **P3:12, P5:13, P9:24, 3:2, 4:16, 5:50, 7:1, 7:16, C:2, C:41-42, C:45**
- exec ... **1:10, 3:15, C:44**
- file descriptor ... **5:16-18**
- file system administration ... **5:27**
- file system check ... **P3:11, 1:8, 5:39, 5:41, 5:49**
- file table ... **5:13-14, 5:16-17, 6:25, 6:47-48**
- firmware ... **P1:2, P3:9-10, C:26**
- fork ... **C:15, C:17, C:44**
- formatting ... **P4:4, 4:8, 5:10, C:3**
- free-block ... **5:4, 5:43-45, 5:61-63**
- freemem ... **6:29, 6:31, 6:35**
- fsck ... **5:40, 5:42**
- getty ... **8:5-6**
- GPGSHI ... **6:10**
- group IDs ... **P10:18, 2:1, 10:5**
- hardware upgrade ... **6:13**
- hdelogger ... **3:15, B:10**
- host ... **P10:1-2, P10:5-9, P10:10, P10:12-13, P10:15-21, P10:23**
- hunt ... **P8:1, P8:3, P8:6, 8:1, 8:5**
- i-node ... **P5:9, 6:7, 6:15, 6:25, 6:46, C:5**
- incremental backup ... **P5:18, P5:21, 5:33, 5:35**
- indirect address ... **5:8**
- indirect block ... **5:6, 5:46-47**
- inittab ... **P9:8**
- interface ... **P1:2, P7:7, 7:1, 7:4-7, 7:12-14, 7:17-18, C:36-37, C:39**
- interface, printer ... **G:7**
- interrupts ... **6:30**
- kernel ... **4:10, 6:1, 6:3, 6:8, 6:10, 6:15, 6:41-42, 6:52-54, C:5-6, C:9, C:14-15, C:17-18, G:5**
- line settings ... **P8:1-3, P8:6, P8:8, 8:1-2**
- link count ... **5:45, 5:48, 5:52, 5:57, 5:59-60, G:8**
- lock files ... **7:19, 9:33**
- log files ... **P9:20, 9:4**
- login services ... **P9:31**
- logins ... **P1:9-11, P2:12, P2:14, P9:16, 1:1-3, 2:6, 3:7, B:15**
- LOGNAME ... **9:21-23, 9:25-29**
- lpadmin ... **P7:7, 7:6-7, C:36, C:38, C:40**
- LPDEST ... **7:5**
- lpmove ... **7:9**
- maintenance ... **P5:14, P7:1, 2:11, 3:1-2, 4:7, 5:37, 10:33-34, 10:42-43**
- mapping ... **P10:17-19, 4:14-15, 10:6, 10:9, 10:23-24, 10:27-29, 10:34, 10:36-37, 10:41, 10:63, G:6**
- mask ... **2:9**
- MAXSEPGCNT ... **6:53**
- MAXUP ... **6:48**
- MDEV ... **P10:22**
- memory ... **10:19, 10:63-64, C:17**
- mknod ... **4:7**
- modem interrupts ... **6:30**
- monitoring files ... **5:28**
- MSGSEG ... **6:56**
- MSGSSZ ... **6:56**

- multiuser ... **P1:4, P1:6, P2:17, P3:1, P3:4, P3:9-11, P3:13, P4:2, P4:5, P8:2, P8:6, P8:8, P9:21, P10:5, P10:15, 3:1-2, 3:8-9, 3:11-13, 3:15-16, 3:21, 4:11, 5:23, 10:37, 10:45, B:20**  
 multiuser mode ... **B:4, B:9, B:23**  
 NAUTOUP ... **6:49**  
 NBUF ... **6:4-5, 6:45, 6:49**  
 ncheck ... **1:11, 1:13**  
 network ... **P9:29-31, P9:36, P10:2-7, P10:10, P10:12-16, 9:7-9, 9:11-12, 9:30, 10:1, 10:4-5, 10:7, 10:19, 10:35-38, 10:40, 10:42, 10:44, 10:49, 10:53, 10:55, 10:60, 10:62-63, G:2**  
 news ... **P10:22, 2:6, 2:8-9, 2:11-13, 3:1, 3:6, 9:28, B:15, B:34**  
 NFILE ... **6:46-47**  
 NHBUF ... **6:4-5, 6:49**  
 NINODE ... **6:46-47**  
 NINODE/NS5INODE ... **6:11**  
 nlsadmin ... **P9:31-32, P9:36, P10:4-5, P10:14-15**  
 node ... **10:43, 10:49, G:3, G:9**  
 NODE ... **P1:6, 6:50, G:9**  
 NOFILES ... **6:47-48, C:3**  
 NPROC ... **6:47-48**  
 NREGION ... **6:47**  
 NS5INODE ... **6:46**  
 NSTREVENT ... **6:53**  
 NSU ... **6:42, 6:52, 6:54**  
 nuucp ... **P2:16, 9:9-10, 9:17-18**  
 open file table ... **5:14, 5:17-18, 6:46**  
 paging ... **P5:7, 6:5, 6:11, 6:23, 6:28, 6:50-51, C:17, G:10**  
 panic ... **C:13, C:25**  
 partition ... **P3:13, P5:7-8, 4:10, 5:10, B:4, C:5**  
 passwd ... **P1:14, P9:16, 2:4, 7:14**  
 password aging ... **1:3-6, 2:3, B:8, B:13**  
 path ... **P3:15, P5:15, P5:24, P5:26, P9:3, P9:27, P10:20-21, 7:4-5, 7:13, 7:18, 9:24, 9:26-27, 9:31, 9:33-34**  
 performance management ... **6:2**  
 performance tools ... **6:14**  
 permissions ... **P9:17, P9:24, P10:7, P10:10, P10:16, P10:18, P10:22, 1:2, 1:10, 2:4, 2:9, 9:21, 9:26, 9:32, 10:8-9, 10:12-13, 10:15, 10:23, 10:26-33, 10:41, B:30, C:9, G:6, G:10**  
 phase checks ... **5:51, 5:57, 5:61**  
 poll ... **P9:6, P9:13, P9:18-19, 9:39-40**  
 power-up ... **5:39, C:25**  
 print job ... **7:13**  
 print requests ... **7:14, C:39**  
 processor utilization ... **6:23**  
 ps ... **10:40**  
 queue lengths ... **6:21**  
 reboot ... **P1:14, P3:6, P3:12, 5:64, 6:13, C:3, C:5, C:11, C:14**  
 reconfiguration ... **6:13, G:16**  
 recovery ... **P6:2, 10:49**  
 remote file sharing ... **10:45, 10:50, 10:61**  
 remote machine ... **P9:10, P9:16, P9:24, P9:29, 9:22, 10:9, 10:19, 10:26, C:44-45, C:47**  
 restore ... **P1:12, P3:1, P3:13, P3:16, P5:26, 4:10, 4:15, 4:18, 5:35-36, C:8, C:35**  
 restricted shell ... **P2:10, 2:10**  
 RFS ... **P10:20, P10:23, 3:9, 6:31, 6:41-42, 6:54, 6:58, 10:1, 10:46, 10:48, 10:61, 10:63-64**  
 root file system ... **P1:13, P5:7, P9:27, 1:11, 3:8, 5:22, 5:64**  
 run-state ... **3:6, B:10, B:30**  
 runocc ... **6:23, 6:31, 6:35**

## Index

---

- sadp ... 6:37
- sar ... 6:5, 6:10-11, 6:14-16, 6:18-20, 6:22-25, 6:27-31, 10:6, 10:34, 10:53-58, 10:62
- SCHEDLOCK ... 7:7-8, 7:20
- security ... P9:14, P9:28, P9:34, P10:16, 1:1-2, 1:7, 1:10, 5:9, 9:25, 9:27, 10:1-2, 10:6, 10:8, 10:12, B:30, G:4
- self-configuration ... B:20
- semaphore ... 6:22, 6:56-58
- SEMUME ... 6:57
- setup ... P2:16, P3:12, P9:5, P10:2, P10:4, P10:8, P10:12, P10:14, 3:8, 10:42, 10:44
- shared memory ... 6:25, 6:41, 6:47, 6:58
- shell ... P2:8-10, P9:2, P9:17-20, 1:10, 7:12, 7:18, 8:3, 8:6, 9:4-5, 9:32, B:13, B:17, B:20, B:23, B:24, G:14
- shell procedure ... 2:6, 3:13, 5:23, 5:40
- shell programming ... 2:8
- shell script ... 2:11, 3:4, 3:5, 3:19, 5:27, 6:18, 10:22, 10:35, 10:45-46, 10:48, 10:50-52, G:2, G:5, G:7, G:11
- shutdown ... P3:6-8, P9:26, 3:20, 7:7, B:24
- single-user ... P1:4-5, P1:9, P3:1, P3:4, P3:7-9, P4:2, P4:5, P5:2, P5:11, P5:17, P7:3-4, P7:6, P9:2, P9:5, P9:26-27, P10:2, P10:12, 3:6, 3:8-9, 3:12, 3:20, 10:45, B:9, B:17, B:28, C:2, C:5, C:9
- single-user mode ... B:4
- STARLAN ... P9:1, P9:14-15, P9:28-31, P9:33-35, P10:3-6, P10:13-14, 9:7, 9:9, 9:30-31, 10:38
- STIME ... 6:8
- Streams ... P9:36, 6:52-53
- STREAMS ... P9:15, 6:52-54, 9:30
- STRLOFRAC ... 6:54
- STRMEDFRAC ... 6:54
- su ... P3:15, P9:20, 2:6, 7:20, B:15, B:26, B:38
- super-user ... P9:20, 2:4, 6:48, B:8, B:13
- swap ... P5:8, 4:10, 6:10, 6:23, 6:27, 6:29, C:5-6, C:9, C:15, C:17, C:20, G:15
- swap area ... G:3
- swpocc ... 6:23
- SYS ... P1:6, 6:50, G:16
- sysdump ... C:15
- system logins ... P1:9-10
- system name ... P5:4, P5:31, 5:22-23, 6:50
- table overflows ... C:11
- TERM ... B:15
- terminal options ... 8:3, 8:8
- terminal settings ... 8:4, B:6
- TERMINFO ... 2:9
- text-bit ... 6:5-6, 6:11
- timex ... 6:35, 6:37
- TLI ... P9:29-31, P9:35-36, 9:10
- Transport Interface ... P9:4, P9:14-15, P10:3, P10:13, 9:30
- TTY ... P7:7, P8:1-3, P8:6-8, P8:9, 8:1, 8:3, 8:6-7
- tty line ... P8:3, P8:7
- tunable parameters ... P1:6, 6:4, 6:50, 6:52, 6:55-56, 6:58, 10:53, 10:60, B:12
- TZ ... B:29
- umask ... 2:6, 2:9, B:15
- umount ... P1:14, P9:27, P10:17, 5:24, 10:21
- UNREF ... 5:56-57, 5:59-60

uucico ... **P9:14-15, P9:30, P9:34,**  
    **P9:36, 9:31**  
uucp ... **P9:10, P9:20, P9:23, 9:9-10,**  
    **9:17-18, B:38**  
UUCP/CU ... **6:54**  
verification check ... **4:11**  
VHANDL ... **6:50**  
VHANDR ... **6:50**  
VHNDFRC ... **6:50-51**  
volume table of contents ... **4:8, 5:9**  
VTOC ... **C:3, C:5, C:8-9, C:11,**  
    **C:17, C:43**  
wall ... **P3:5, B:26, B:27**  
wcache ... **6:16, 6:31, 6:35**













Other books in the Prentice Hall C and UNIX® Systems Library

- The C Programmer's Handbook **Bell Labs/M. I. Borsky**
- The UNIX System User's Handbook **Bell Labs/M. I. Borsky**
- The Vi User's Handbook **Bell Labs/M. I. Borsky**
- UNIX System Software Readings **AT&T UNIX PACIFIC**
- UNIX System Readings and Applications, Volume I **Bell Labs**
- UNIX System Readings and Applications, Volume II **Bell Labs**
- UNIX System V/386 Utilities Release Notes **AT&T**
- UNIX System V/386 Streams Primer **AT&T**
- UNIX System V/386 User's Guide, Second Edition **AT&T**
- UNIX System V/386 User's Reference Manual **AT&T**
- UNIX System V/386 Programmer's Reference Manual **AT&T**
- UNIX System V/386 Streams Programmer's Guide **AT&T**
- UNIX System V/386 Network Programmer's Guide **AT&T**
- UNIX System V/386 Programmer's Guide **AT&T**
- UNIX System V/386 System Administrator's Guide **AT&T**
- UNIX System V/386 System Administrator's Reference Manual **AT&T**

**PRENTICE HALL, Englewood Cliffs, N.J. 07632**

ISBN 0-13-940891-6