System Manager's Manual

for the

Sun Workstation — Models 100U/150U

Sun Microsystems, Inc.,
2550 Garcia Avenue,
Mountain View,
California 94043
(415) 960-1300
This manual supersedes the *User's Guide for the Sun Workstation* dated 15th October 1982. The prior User's Guide was specific to Sun Workstations running the Version 7 UNIX† system.

**Credits and Trademarks**

Multibus is a trademark of Intel Corporation.

Sun Workstation, Sun-1, and Sun-2 are trademarks of Sun Microsystems, Incorporated.

UNIX is a trademark of Bell Laboratories.

**Acknowledgements**

The tutorials in the final section of this manual were originally derived from the work of various people associated with various institutions. Their names and the titles of the original works are:

*Building Sun Workstation Kernels*

was originally written by Samuel J. Leffler, formerly of the Computer Systems Research Group, University of California at Berkeley.

*Fack — The UNIX File System Check Program*

was originally written by T. J. Kowalski, Bell Laboratories, Murray Hill, New Jersey; and

was revised by Marshall Kirk McKusick, Computer Systems Research Group, University of California at Berkeley.

*SENDMAIL — An Internetwork Mail Router*

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

*SENDMAIL — Installation and Operation Guide*

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

*Uucp Implementation Description*

was originally written by D. A. Nowitz, Bell Laboratories, Murray Hill, New Jersey.

*USENET Installation and Maintenance*

was originally written by Matt Glickman, Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, and revised by Mark Horton.

*Line Printer Spooler Manual*

was originally written by Ralph Campbell, Computer Systems Research Group, University of California at Berkeley.

**Copyright © 1984 by Sun Microsystems, Inc.**

This publication is protected by Federal Copyright Law, with all rights reserved. No part of this publication may be reproduced, stored in a retrieval system, translated, transcribed, or transmitted, in any form, or by any means manual, electric, electronic, electro-magnetic, mechanical, chemical, optical, or otherwise, without prior explicit written permission from Sun Microsystems.

† UNIX is a trademark of Bell Laboratories.
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>19th February 1983</td>
<td>First ( \beta ) release of this System Manager's Guide. [First ( \beta ) release of this System Manager's Guide.]</td>
</tr>
<tr>
<td>B</td>
<td>11th March 1983</td>
<td>Corrected errors and misinformation in original release.                           [Corrected errors and misinformation in original release.]</td>
</tr>
<tr>
<td>C</td>
<td>15th April 1983</td>
<td>Corrected errors in setup information. Added material on setting up UUCP, Mail System, and Printers. [Corrected errors in setup information. Added material on setting up UUCP, Mail System, and Printers.]</td>
</tr>
<tr>
<td>D</td>
<td>1st August 1983</td>
<td>Rewrote for 100U/150U. Added information on subsystem set-up, system set-up and operation, and upgrading. Revised hardware documentation, and included Xylogics 450 disk controller. Revised and updated installation procedures. [Rewrote for 100U/150U. Added information on subsystem set-up, system set-up and operation, and upgrading. Revised hardware documentation, and included Xylogics 450 disk controller. Revised and updated installation procedures.]</td>
</tr>
<tr>
<td>E</td>
<td>15th September 1983</td>
<td>Updated for new Sun CPU and Memory Expansion Boards. Added several tutorials on new aspects of the system, and revised operation and maintenance procedures to reflect new software. [Updated for new Sun CPU and Memory Expansion Boards. Added several tutorials on new aspects of the system, and revised operation and maintenance procedures to reflect new software.]</td>
</tr>
<tr>
<td>F</td>
<td>1st November 1983</td>
<td>Revised for 1.0 Release.                                                                                                                  [Revised for 1.0 Release.]</td>
</tr>
<tr>
<td>H</td>
<td>12th March 1984</td>
<td>Bug corrections, minor additions.                                                    [Bug corrections, minor additions.]</td>
</tr>
</tbody>
</table>
System Manager's Manual

Table of Contents

Section I. System Installation and Maintenance Guide

Gives unpacking and set-up directions for the Model 100U and 150U Sun Workstations and their subsystems; walks you through UNIX installation; and presents basic information for hardware and software operation, maintenance, and upgrading. This is the system administrator's basic tool. A detailed table of contents precedes the manual proper.

Section II. Commands and Utilities

Reference manual for system maintenance and operation commands. Contains information related to system bootstrapping, operation, and maintenance; and describes all the server processes and daemons which run on the system. See the first entry of the manual, intro(8), for a more detailed description and table of contents.

Section III. Tutorials

Presents several papers on relevant aspects of the system:

1. Building Sun Workstation Kernels
2. Fack — The UNIX File System Check Program
3. SENDMAIL — An Internetwork Mail Router
4. SENDMAIL — Installation and Operation Guide
5. Uucp Implementation Description
6. USENET Installation and Maintenance
7. Line Printer Spooler Manual
Welcome to the Sun Workstation.
This manual is meant to help you get the Sun Workstation Models 100U (desktop) and 150U (rackmount) up and running. The manual has three basic divisions. The first, *System Installation and Maintenance Guide*, gives unpacking and set-up directions for new desktop and rackmount workstations and their subsystems; walks you through UNIX installation procedures for Sun systems; and covers the basics on hardware and software operation, maintenance, and upgrading. The second section is a reference manual for UNIX commands and utilities relevant to system management. The final section includes several installation tutorials.
Summary of Contents

Contents of the *System Installation and Maintenance Guide* are:

**Chapter 1 — Unpacking and Setting Up the 100U** — is a guide to getting a new Model 100U out of its shipping cartons and setting up the desktop workstation ready to run. If you have disk and/or tape subsystems, work through Chapter 3 next to complete your hardware set up.

**Chapter 2 — Unpacking and Setting Up the 150U** — is a guide to getting a new Model 150U out of its shipping cartons and setting up the rackmount workstation ready to run. If you have disk and/or tape subsystems, work through Chapter 3 next to complete your hardware set up.

**Chapter 3 — Subsystem Set-up** — covers unpacking, mounting, and cabling the disk and tape subsystems which may be supplied with either the 100U or the 150U. Also describes how to add peripherals to existing subsystems.

**Chapter 4 — Installing UNIX for the First Time** — guides you through the procedure for installing the UNIX System on a new Sun Workstation from a half-inch or quarter-inch distribution tape. Network installation procedures are also covered.

**Chapter 5 — Installing UNIX on Systems Without Tape Support** — describes the procedure for installing the UNIX System on a Sun Workstation equipped with only a disk drive (no tape), over the Ethernet from a system equipped with a tape drive.

**Chapter 6 — System Set-up and Operation** — covers basic system set-up and maintenance procedures. Topics from *uucp* and *sendmail* installation to doing dumps are covered.

**Chapter 7 — Upgrading System Software** — gives a brief review of how to upgrade your system to a new software release level.

**Chapter 8 — Hardware Configuration and Expansion** — describes the 7-slot Multibus card cage in the Model 100U and the 15-slot cage in the 150U, and provides configuration details for each board in the card cage. You must use this chapter to determine placement and switchsettings for new boards when you add boards or peripherals. Also useful for troubleshooting, if you need to get inside the card cage and identify the boards.

**Appendix A — Start-up and Bootstrap** — discusses the startup and boot functions of the Sun Workstation monitor, and lists diagnostic messages which it currently displays.

The second section of this manual is a reference guide to UNIX commands and utilities of special interest to system managers. Section 8 covers the procedures and commands needed for day-to-day administration of the UNIX system on the Sun Workstation. The pages are arranged in alphabetical order.

The final section of the manual contains several installation tutorials.

This manual is still under development and will undergo several revisions before it reaches its final form. It will evolve over time to cover the kinds of things you do on a day to day basis; and to address the kinds of problems that you are likely to encounter, and what to do about those problems. To help direct our development — and to help us maintain the currency and accuracy of this material — we have supplied a reader comment sheet at the end of this guide. Please use the comment sheet to list errors, omissions and outright lies. Your responses will help a great deal in our efforts to keep our documentation cogent.
Applicable Documents

We emphasize that this manual outlines rather than exhausts many of its topics. References to applicable documents supplied with your system are given throughout, however, and we urge you to read these documents when you can.

Table 1: Hardware and System Documentation

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>800-0277</td>
<td>Fujitsu M2311K/2312K Microdisk Drives CE Manual</td>
</tr>
<tr>
<td>800-0274</td>
<td>Interphase SMD 2180 Storage Module Controller/Formatter User's Guide</td>
</tr>
<tr>
<td>800-0277</td>
<td>Fujitsu M2311K/M2312K Microdisk Drives CE Manual</td>
</tr>
<tr>
<td>800-0339</td>
<td>Sun 1024 Video Board User Manual</td>
</tr>
<tr>
<td>800-0393</td>
<td>3COM 3C400 Multibus Ethernet Controller Reference Manual</td>
</tr>
<tr>
<td>800-0398</td>
<td>Sun Color Video Board User's Manual</td>
</tr>
<tr>
<td>800-0402</td>
<td>User's Guide for the Sun Workstation Mouse Subsystem</td>
</tr>
<tr>
<td>800-0415</td>
<td>Sun 1/4&quot; Tape Interface User Manual</td>
</tr>
<tr>
<td>800-0485</td>
<td>Power Drain Requirements for Sun Workstation Boards</td>
</tr>
<tr>
<td>800-0620</td>
<td>CPC TAPEMASTER Product Specification</td>
</tr>
<tr>
<td>800-0622</td>
<td>CPC TAPEMASTER Application Note</td>
</tr>
<tr>
<td>800-0623</td>
<td>CDC Streaming Tape Unit 9218X Reference Manual (1/2&quot; Tape Drive)</td>
</tr>
<tr>
<td>800-0628</td>
<td>Archive Product Manual (Sidewinder 1/4&quot; Streaming Cartridge Tape Drive)</td>
</tr>
<tr>
<td>800-1000</td>
<td>Philips Preliminary Service/Operator Manual High Resolution 17&quot; CRT Display</td>
</tr>
<tr>
<td>800-1002</td>
<td>BARCO GD 33 Color Display 120VAC Operation Instruction (13&quot; Color Monitor Manual)</td>
</tr>
<tr>
<td>800-1007</td>
<td>Fujitsu M228X Fixed Disk Unit CE Manual</td>
</tr>
</tbody>
</table>
Hardware and System Documentation

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>800-1011</td>
<td>Fujitsu M2351A/AF Mini-Disk Drive CE Manual</td>
</tr>
<tr>
<td>800-1005</td>
<td>Xylogics Model 450 Peripheral Processor SMD Disk Subsystems Maintenance and Reference Manual</td>
</tr>
<tr>
<td>800-1052</td>
<td>Zilog Serial Communications Controller Technical Manual</td>
</tr>
<tr>
<td>800-1104</td>
<td>Fast Floating-Point Processor System Integration Manual</td>
</tr>
</tbody>
</table>

Table 2: Software Documentation

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>800-1107</td>
<td>User's Manual for the Sun Workstation</td>
</tr>
<tr>
<td>800-1108</td>
<td>System Interface Manual for the Sun Workstation</td>
</tr>
<tr>
<td>800-1115</td>
<td>Programmer's Reference Manual for SunCore</td>
</tr>
<tr>
<td>800-1116</td>
<td>Programmer's Reference Manual for SunWindows</td>
</tr>
<tr>
<td>800-1117</td>
<td>System Internals Manual for the Sun Workstation</td>
</tr>
</tbody>
</table>

Notations Used In This Guide

References to commands and utilities from the User's Manual for the Sun UNIX System and the System Interface Manual for the Sun UNIX System use the notation:

```
passwd(1)
```

to indicate the `passwd` page in section 1 of the User's Manual. (The User's Manual contains sections 1, 6, and 7.)** Similarly, `passwd(5)` means refer to the `passwd` manual page in section 5 of the System Interface Manual. (The System Interface Manual contains sections 2, 3, 4, and 5.) The notation `spline(1G)` means that this is a Graphics command in section 1 of the User's Manual. Finally, `/etc/reboot(8)` implies that this is a command from section 8, which is in this manual. (The System Manager's Manual contains section 8.)

** For an explanation of this weird numbering scheme, see the Introduction to the Sun UNIX System Manuals in the User's Manual.
Throughout the examples, there are numbers in the notation:

`0xnnnn`

A number with `0x` in front of it is a hexadecimal (base 16) number. Those wishing more information should consult any of the literature on the C programming language.

Finally, this manual provides a number of tutorial "walkthrough" examples which use certain formatting conventions. In all these examples, what the user types is shown in boldfaced text, like this. The system's response is shown in roman type, like this. Variable items which you must substitute — which depend on your system's specific configuration — are shown in italic type, like this. Be especially careful with all such substitutions. Lastly, text which appears [ within brackets ] is commentary; usually, we use brackets to abbreviate a long series of system response messages, or to offer explanation of a process.
System Installation
and Maintenance Guide

for the

Sun Workstation — Models 100U/150U

CAUTION: This document does not cover hardware upgrades; it is intended for use with new systems. If you are upgrading your 100/150, follow the hardware upgrading procedures given in Sun's Upgrade Installation Guide BEFORE using this manual.

Sun Microsystems, Inc.,
2550 Garcia Avenue,
Mountain View,
California 94043
(415) 960-1300
Table of Contents

Chapter 1  Unpacking and Setting up the 100U ................................................................. 1-1
Chapter 2  Unpacking and Setting up the 150U ................................................................. 2-1
Chapter 3  Subsystem Set-Up ............................................................................................. 3-1
Chapter 4  Installing UNIX for the First Time ................................................................. 4-1
Chapter 5  Installing UNIX on Systems without Tape Support .................................. 5-1
Chapter 6  System Set-Up and Operation ........................................................................... 6-1
Chapter 7  Upgrading System Software ........................................................................... 7-1
Chapter 8  Hardware Configuration and Expansion ...................................................... 8-1
Appendix A  Power-up and Bootstrap ............................................................................... A-1
# Table of Contents

**Introduction** ........................................................................................................................................... vii  

**Chapter 1** Unpacking and Setting up the 100U ................................................................................... 1-1  
1.1. Safety Precautions ............................................................................................................... 1-1  
1.2. Environmental Considerations ............................................................................................ 1-1  
1.2.1. Physical Environment .................................................................................................. 1-2  
1.2.2. Power Drain Requirements ......................................................................................... 1-2  
1.3. Receipt and Unpacking Instructions .................................................................................. 1-3  
1.4. Set-up .................................................................................................................................... 1-4  
1.4.1. Keyboard and Mouse .................................................................................................... 1-5  
1.4.2. Ethernet ......................................................................................................................... 1-5  
1.4.3. Asynchronous Serial Ports .......................................................................................... 1-6  

**Chapter 2** Unpacking and Setting up the 150U .................................................................................. 2-1  
2.1. Safety Precautions ............................................................................................................... 2-1  
2.2. Environmental Considerations ............................................................................................ 2-1  
2.2.1. Physical Environment .................................................................................................. 2-2  
2.2.2. Power Drain Requirements ......................................................................................... 2-2  
2.3. Receipt and Unpacking Instructions .................................................................................. 2-3  
2.3.1. 150U Monitor — Unpacking ........................................................................................ 2-4  
2.3.2. 150U Card Cage Enclosure — Unpacking and Mounting ........................................ 2-5  
2.4. Set-up .................................................................................................................................... 2-5  
2.4.1. Keyboard and Mouse .................................................................................................... 2-5  
2.4.2. Ethernet ......................................................................................................................... 2-6  
2.4.3. Asynchronous Serial Ports .......................................................................................... 2-6  

**Chapter 3** Subsystem Set-Up ............................................................................................................ 3-1  
3.1. M2312K (Fujitsu 84-MByte) Disk ....................................................................................... 3-1  
3.1.1. M2312K (Fujitsu 84-MByte) Disk Expansion ............................................................... 3-3  
3.2. M2284 (Fujitsu 169-MByte) Disk ..................................................................................... 3-4  
3.2.1. M2284 (Fujitsu 169-MByte) Disk Expansion ............................................................... 3-6  
3.3. M2351 (Fujitsu 474-MByte) Disk ..................................................................................... 3-7  
3.3.1. M2351 (Fujitsu 474-MByte) Disk Expansion ............................................................... 3-8  
3.4. Quarter-Inch Magnetic Tape ............................................................................................... 3-8  
3.5. Nine-Track Magnetic Tape ................................................................................................. 3-10
Chapter 4 Installing UNIX for the First Time .............................................. 4-1
4.1. What is on the Distribution Tape? .......................................................... 4-1
4.2. Overview of the Installation Procedure .................................................. 4-3
4.3. Installation Walkthrough ........................................................................... 4-5
  4.3.1. Determining Network Information ....................................................... 4-5
    4.3.1.1. Procedure for New Systems ............................................................... 4-5
    4.3.1.2. Procedure for Existing Systems ....................................................... 4-6
  4.3.2. Loading the Tape ................................................................................... 4-6
  4.3.3. Loading the Bootstrap Program ............................................................. 4-6
  4.3.4. Using the Diag Utility ............................................................................ 4-7
    4.3.4.1. Formatting the Disk ........................................................................... 4-9
    4.3.4.2. Labelling the Disk ............................................................................ 4-10
  4.3.5. Loading the Mini UNIX System ............................................................. 4-11
  4.3.6. Booting the Mini UNIX System ............................................................. 4-11
  4.3.7. Setting the Date .................................................................................... 4-12
  4.3.8. Loading the Root File System ............................................................... 4-12
  4.3.9. Booting the UNIX System ..................................................................... 4-13
  4.3.10. Using Setup to Configure Your System .............................................. 4-14
    4.3.10.1. Path 1: Standalone System Configuration .................................. 4-16
    4.3.10.2. Path 2: Standard Server Configuration ......................................... 4-18
    4.3.10.3. Path 3: Non-Standard Server Configuration .................................. 4-21
    4.3.10.3. Configuration Worksheet for Path 3 .............................................. 4-21
    4.3.10.3. Setup Walkthrough for Path 3 ....................................................... 4-27
  4.3.11. Booting the Full UNIX System ........................................................... 4-30
  4.3.12. Kernel Configuration .......................................................................... 4-31
    4.3.12.1. Making a New Configuration .......................................................... 4-31
    4.3.12.2. Configuration Directory Layout ....................................................... 4-36
    4.3.12.3. Building a Configuration ................................................................. 4-37
    4.3.12.4. Format of the Configuration File ..................................................... 4-37
    4.3.12.4. General System Description Lines ............................................... 4-38
    4.3.12.4. Device Description Lines ................................................................. 4-39
  4.3.13. Loading the Manuals, Demos, and Games Directories ......................... 4-40
  4.3.14. The Next Step ..................................................................................... 4-41

Chapter 5 Installing UNIX on Systems without Tape Support ......................... 5-1
5.1. Overview of the Installation Procedure ...................................................... 5-1
5.2. Determining Network Information ........................................................... 5-3
5.3. Setting up the Remote Host ....................................................................... 5-4
5.4. Loading the Mini File System to the Server .............................................. 5-4
5.5. Running Diag ............................................................................................. 5-5
    5.5.1. Formatting the Disk ........................................................................... 5-6
    5.5.2. Labelling the Disk ............................................................................ 5-6
5.6. Loading the Mini UNIX System ................................................................. 5-7
5.7. Booting the Mini UNIX System ................................................................. 5-7
5.8. Setting the Date .......................................................................................... 5-8
5.9. Editing the /etc/hosts and .rhosts Files ................................................. 5-8
5.10. Loading the Root File System ................................................................. 5-8
5.11. Booting the UNIX System in Single-User State ........................................ 5-9
5.12. Using Setup to Configure Your System .................................................. 5-9
5.13. Booting the Full UNIX System ................................................................. 5-11
5.14. Loading the Manuals, Demos, and Games Directories ......................... 5-12

Chapter 6 System Set-Up and Operation ....................................................... 6-1
6.1. Network Configuration ............................................................................. 6-1
  6.1.1. Setting up a Gateway Machine .......................................................... 6-1
  6.1.2. Reducing Network Overhead .............................................................. 6-2
  6.1.3. Network Security — /etc/host.equiv and .rhosts ............................... 6-3
  6.1.4. Making 1.1 Networks Compatible with Existing Networks .................. 6-5
6.2. Setting Up the Mail System ..................................................................... 6-6
  6.2.1. Picking a Name for your Domain ....................................................... 6-8
  6.2.2. Picking a “Main Machine” for Mail Forwarding ................................. 6-9
    6.2.2.1. Configuring for Mail Forwarding .................................................. 6-9
    6.2.2.2. Telling Sendmail your Domain Name ........................................... 6-9
    6.2.2.3. Setting up the “Postmaster” Alias ................................................ 6-11
  6.2.3. Testing your Mailer Configuration ...................................................... 6-12
  6.2.4. Diagnosing Troubles with Mail Delivery ............................................ 6-12
6.3. System Log Configuration ....................................................................... 6-13
6.4. Setting Up a UUCP Connection ............................................................... 6-13
  6.4.1. UUCP Installation Overview ............................................................... 6-15
6.5. USENET Installation/Conversion ............................................................ 6-17
  6.5.1. USENET Installation ....................................................................... 6-17
  6.5.2. Conversion from USENET Version 2.9 to 2.10 ................................. 6-19
6.6. Setting Up the Line Printer System ......................................................... 6-20
  6.6.1. Printer Configuration ....................................................................... 6-21
  6.6.2. Line Printer Troubleshooting ............................................................. 6-22
6.7. Adding Users ........................................................................................... 6-23
6.8. Bootstrap and Shutdown Procedures ....................................................... 6-23
6.9. Device Errors and Diagnostics ............................................................... 6-24
6.10. File System Checks, Backups, and Disaster Recovery ......................... 6-24
6.11. Moving Filesystem Data ...................................................................... 6-25
6.12. Monitoring System Performance ......................................................... 6-25
6.13. Making Local Modifications ................................................................. 6-26
6.14. Accounting ........................................................................................... 6-26
6.15. Resource Control .................................................................................. 6-26
6.16. Network Troubleshooting ..................................................................... 6-26
6.17. Files which Need Periodic Attention .................................................... 6-27

Chapter 7 Upgrading System Software ......................................................... 7-1
7.1. Step 1: What To Save ............................................................................. 7-1
7.2. Step 2: Merging .................................................................................... 7-3
Chapter 8 Hardware Configuration and Expansion

8.1. The 100U Card Cage
8.1.1. 100U Multibus Mastering Scheme
8.1.2. 100U Multibus P2 Connector
8.1.3. 100U Card Insertion Order

8.2. The 150U Card Cage
8.2.1. 150U Multibus Mastering Scheme
8.2.2. 150U Multibus P2 Connectors
8.2.3. 150U Card Insertion Order

8.3. Sun-2 CPU
8.3.1. CPU Multibus Priority
8.3.2. Asynchronous Serial Ports

8.4. Sun-2 Memory Board
8.5. Video Board
8.6. Xylogics 450 Disk Controller
8.7. Interphase 2180 Disk Controller
8.8. Nine-Track Tape Controller
8.9. Quarter-Inch Tape Controller
8.9.1. Tape Controller Multibus Memory
8.10. 3COM Ethernet Controller Board
8.11. Sky Floating Point Board

Appendix A Power-up and Bootstrap

A.1. Power-On Self Test Procedures
A.1.1. Critical Errors from Self Test
A.1.2. Non-Critical Errors from Self Test
A.1.3. No Errors from Self Test
A.2. Automatic Boot Procedure
A.3. Booting from Specific Devices
A.3.1. Booting from Disk
A.3.2. Booting from Network Disk
A.3.3. Booting from Tape
A.3.4. Booting Files from the Default Device
A.4. Messages from the Monitor and Boot Program
List of Tables

Table 0-1 Hardware and System Documentation ........................................... ix
Table 0-2 Software Documentation ........................................................................ x
Table 1-1 Physical Environment ........................................................................ 1-2
Table 1-2 Power Supply Ratings ........................................................................ 1-2
Table 1-3 Power Consumption of Boards ............................................................ 1-2
Table 1-4 Ethernet Cabling Limitations .............................................................. 1-5
Table 2-1 150U Physical Environment ................................................................ 2-2
Table 2-2 Model 150U Power Consumption ......................................................... 2-2
Table 2-3 Power Consumption of Boards ............................................................ 2-2
Table 2-4 Ethernet Cabling Limitations .............................................................. 2-6
Table 4-1 Contents of Distribution Tape(s) .......................................................... 4-1
Table 4-2 UNIX Disk Device Abbreviations ......................................................... 4-4
Table 4-3 UNIX Tape Device Abbreviations ......................................................... 4-4
Table 4-4 Default Addresses for Disk Controllers ............................................. 4-8
Table 4-5 Layout of a Generic Abstract Disk ...................................................... 4-15
Table 4-6 Increase in /pub Size to Accommodate Online Manual Pages, Demos, or Games ........................................................................ 4-15
Table 5-1 Tape Device Abbreviations .................................................................. 5-2
Table 5-2 Disk Device Abbreviations .................................................................... 5-2
Table 5-3 Default Addresses for Disk Controllers ............................................. 5-5
Table 7-1 Standard List of Files to Save when Upgrading ................................. 7-1
Table 7-2 UNIX Tape Device Abbreviations ......................................................... 7-2
Table 8-1 Pin Assignments for J1 on the CPU Board ............................................ 8-9
Table 8-2 Video Board Address Selection .......................................................... 8-10
Table 8-3 Selecting Disk Controller Base Address ............................................. 8-13
Table 8-4 Selecting Quarter-Inch Tape Controller Interrupt Level ................. 8-14
Table 8-5 Selecting Quarter-Inch Tape Controller Base Address ..................... 8-15
Table 8-6 Selecting Quarter-Inch Tape Controller Memory Base Address .......... 8-16
Table 8-7 Selecting SKY Board Interrupt Level .................................................. 8-17
Table 8-8 Selecting SKY Board Base Address .................................................... 8-18
Chapter 1

Unpacking and Setting up the 100U

This chapter outlines how to unpack and set up your new Sun Model 100U and some of its peripherals (subsystems are covered in the chapter Subsystem Set-up).

If you are upgrading your Sun Model 100 rather than starting with a system 'fresh from the box,' please heed the warning on the front page of this manual. Sun-1 systems which are upgrading must not attempt to install this release of the system until all Multibus memory has been removed or disabled, and — for an upgrade from a Xylogics 440 to a Xylogics 450 — cabling has been re-routed. Disk switch settings may also need to be reset. Make sure you have followed the procedures given in Sun's Upgrade Installation Guide before proceeding.

If you are adding boards and/or peripherals to an existing system, the chapter on Hardware Configuration and Expansion will give you the information you need.

1.1. Safety Precautions

DO NOT attempt procedures which are not described in this manual; if you do, you may void your warranty. Please refer all servicing not described in this document to qualified service personnel. If in doubt, contact your authorized Sun service representative.

Observe common-sense safety precautions as you would for any electrical or electronic equipment. Always disconnect the power cord before opening any system enclosure.

Even after power is disconnected from the workstation, very high voltages remain in the capacitors behind the picture tube. It takes about ten minutes for the capacitors to lose their charge.

The following sections contain set-up and configuration specifications for the Sun Workstation. Consult the additional hardware manuals supplied with your workstation (and with user-supplied Multibus accessory boards, if any) for further information and precautions.

1.2. Environmental Considerations

This section describes the environmental requirements for the Sun Model 100U.

Attention: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio
communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

1.2.1. Physical Environment

The Sun Model 100U is manufactured for the following physical environment:

Table 1-1: Physical Environment

<table>
<thead>
<tr>
<th></th>
<th>Operating</th>
<th>Non-Operating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambient Temperature</td>
<td>10 °C - 40 °C</td>
<td>-40 °C - 65 °C</td>
</tr>
<tr>
<td>Humidity</td>
<td>5% - 90%</td>
<td>5% - 90%</td>
</tr>
<tr>
<td>(Non-condensing)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Altitude</td>
<td>10,000 feet</td>
<td>40,000 feet</td>
</tr>
</tbody>
</table>

1.2.2. Power Drain Requirements

Model 100U desktop workstations are equipped with a 200 watt four-output power supply, which delivers +5, -5, +12, and -12 volts. The table which follows shows the power consumption of each Sun-supplied board. Boards documented are those current for this release.

Table 1-2: Power Supply Ratings

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Maximum Current in Amps</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5</td>
<td>35.0</td>
</tr>
<tr>
<td>-5</td>
<td>1.5</td>
</tr>
<tr>
<td>+12</td>
<td>4.0</td>
</tr>
<tr>
<td>-12</td>
<td>1.5</td>
</tr>
</tbody>
</table>

It is possible that a full card cage will exceed the ratings of this supply. The table which follows shows the power consumption of each Sun-supplied board. Boards documented are those current for this release.

1 NOTE that this rating is for the Model 100U current with this release; older systems (pre-0.4 release of August 1983) have 175 watt, 30 amp power supplies. The peak amperage for these systems is: +5: 30 amps; -5: 2.0 amps; +12: 4.0 amps; -12: 2.0 amps.
Table 1-3: Power Consumption of Boards

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>BOARD</th>
<th>AMPS</th>
<th>TOTAL WATTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>370-1012</td>
<td>Xylogics SMD Disk Controller Board</td>
<td>8.0</td>
<td>45</td>
</tr>
<tr>
<td>370-1021</td>
<td>Sky Floating Point Processor</td>
<td>4.0</td>
<td>20</td>
</tr>
<tr>
<td>501-0289</td>
<td>Color Display Controller</td>
<td>6.0</td>
<td>36</td>
</tr>
<tr>
<td>590-0059</td>
<td>Black and White Display Controller</td>
<td>5.0</td>
<td>25</td>
</tr>
<tr>
<td>590-0217</td>
<td>Interphase SMD Disk Controller Board</td>
<td>4.0</td>
<td>23</td>
</tr>
<tr>
<td>590-0288</td>
<td>10Mbit Ethernet Board</td>
<td>5.0</td>
<td>31</td>
</tr>
<tr>
<td>590-0502</td>
<td>1/2-inch Tape Controller Board</td>
<td>4.0</td>
<td>20</td>
</tr>
<tr>
<td>590-0526</td>
<td>1/4-inch Tape Controller Board</td>
<td>3.0</td>
<td>15</td>
</tr>
<tr>
<td>590-1007</td>
<td>Sun-2 CPU Board, Keyboard, and Mouse</td>
<td>6.0</td>
<td>30</td>
</tr>
<tr>
<td>590-1013</td>
<td>1M Memory Board</td>
<td>3.0</td>
<td>15</td>
</tr>
<tr>
<td>590-1013</td>
<td>Sun-2 (low power) Memory Board</td>
<td>2.5</td>
<td>13</td>
</tr>
</tbody>
</table>

Note that the keyboard and mouse draw their power from the CPU board.

Review the above list carefully to determine exactly what configuration of boards you can have in the card cage. If you get boards from other suppliers, make sure you obtain the current power consumption information from them, so that you can factor that information in with the list above. Overloading the rated capacity of the power supply can cause system malfunction (which often shows up as "glitches" on the video screen), and may damage the power supply.

1.3. Receipt and Unpacking Instructions

Note that the shipping weight of the Sun Workstation and its shipping carton is approximately 105 pounds. The weight of the workstation itself is about 90 pounds.

1. When you receive your shipment, inspect all shipping cartons immediately for evidence of damage. If any shipping carton is severely damaged, request that the carrier's agent be present when the carton is opened. If the carrier's agent is not present when a carton is opened and the contents are found to be damaged, keep all contents and packing materials for the agent's inspection.

2. To open the shipping carton, cut the binding straps on it with a knife or scissors. The shipping carton is in the form of a square 'tube' with two lids on it, one on the bottom, and one on the top. The workstation is encased in styrofoam packing materials inside the lids.

3. Remove the top lid from the 'tube', and you will find the keyboard, the distribution tape(s), the power cord, and an envelope containing manuals in the recesses in the top of the packing.
material. Remove these items.

4. Lift out the rubber and styrofoam packing material (the fit is fairly tight; try rocking it out). At this point you may find a mouse and mouse pad tucked along the side of the monitor; if you ordered an 84-MByte disk subsystem, these will be packed with it instead.

5. Remove the square ‘tube’ from its bottom lid. Don’t try to lift the Sun Workstation out of the shipping carton: it’s too heavy. When you remove the ‘tube’ from the bottom lid, the workstation is left standing in the bottom piece of packing material. You can then lift the workstation out of the bottom shipping material, remove the plastic wrapping, and place the workstation on its working surface.

We recommend that you save the salvageable shipping cartons and packing material for future use in case the product must be reshipped.

1.4. Set-up

The following subsections describe how to cable up your Sun Model 100U and its basic peripherals.

**CAUTION:** Before plugging in the power cord of any component of your Sun system, be sure that the line power supply voltage and frequency are as specified on the back panel of your workstation. Use only three-prong (grounded) outlets. Always disconnect power before opening any system enclosure or servicing any system component. All servicing should be performed by qualified personnel.
1.4.1. Keyboard and Mouse

The Sun keyboard should be plugged into the connector labelled "KEYBOARD" on the workstation back panel. If you wish to use the keyboard as your console input device (as is normally done) you must power-on the workstation AFTER plugging in the keyboard. The Sun resident PROM monitor will try to use serial port A for input if it does not find an attached keyboard. Plug the mouse into the "MOUSE" connector on the backpanel.

1.4.2. Ethernet

If you purchased a system with Ethernet, your system is shipped with the Ethernet Controller Board installed in your workstation's Multibus card cage. You will receive a package containing the transceiver and transceiver cable necessary to complete implementation of the Ethernet for a single workstation, and an Ethernet Controller Reference Manual. The coaxial cable necessary to implement a network with multiple machines may be purchased separately from Sun.

Setting up an Ethernet with all Sun-supplied components is fairly straightforward:

1. Terminate the 50-ohm coaxial cable by placing a transceiver at both ends. Screw the cable into the transceiver N-connectors; cover the open N-connector on each transceiver with a terminator. In lieu of a transceiver, you may terminate the cable with two barrel connectors and terminators. Handle the coaxial cable with some care, as it is fragile; don't run it in an area where it may be run over.

2. For each workstation, plug one end of the transceiver cable into the 15-pin D-connector on the transceiver, and the other end into the "ETHERNET 10 MBIT" connector on the workstation backplane.

Please note that there are certain cabling limitations which must be observed for proper Ethernet implementation:
Table 1-4: Ethernet Cabling Limitations

<table>
<thead>
<tr>
<th>Ethernet Cabling Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIMUM contiguous length of coaxial cable segments</td>
</tr>
<tr>
<td>Distance between transceivers*</td>
</tr>
<tr>
<td>MAXIMUM length of transceiver cable</td>
</tr>
</tbody>
</table>

* Transceivers must be placed at 2.5 meter intervals along the coaxial cable. These intervals are marked with a black band on the cable. If you want to spread transceivers along a greater length of cable, make sure the distance between any two transceivers is a multiple of 2.5 meters. For more information, see the Ethernet manual shipped with your system. If you have questions about repeaters or non-standard Ethernet configurations, please contact Sun Microsystems.

1.4.3. Asynchronous Serial Ports

You may attach a terminal, modem, printer, plotter, or other device which uses the RS-232 interfaces, to one of the serial port connectors labelled “RS232 A” and “RS232 B” on the back panel.

The serial ports on the Sun Workstation were designed primarily for connecting output peripherals, such as printers and plotters, and can drive these output lines at speeds up to 19.2 Kbaud. Both ports provide the CTS, RTS, DTR, DSR, and DCD control lines required by some devices (such as modems) in addition to the transmit and receive lines; the ports generate DTR and RTS, and pay attention to DSR, CTS, and DCD. Both ports are also wired as DTE (Data Terminal Equipment) ports, and thus permit direct connection of modems. Computers and other DTE devices can be connected using the null modem cable supplied by Sun. See the Asynchronous Serial Ports section of the Hardware Configuration and Expansion chapter for wiring specifications.

NOTE that older workstations (shipped prior to the 1.0 Release) have serial port A configured as a DCE (Data Communications Equipment) and serial port B configured as a DTE. In addition, only serial port A on the older workstations has full modem control. This means that external cables built for the Sun-1 or Sun-1.5 CPU may have to be modified to work correctly with the Sun-2 CPU, or a null modem cable (such as that supplied by Sun) may be required. See the Asynchronous Serial Ports section of the Hardware Configuration and Expansion chapter for wiring specifications.
Chapter 2

Unpacking and Setting up the 150U

This chapter outlines how to unpack and set up your new Sun Model 150U and some of its peripherals (subsystems are covered in the chapter Subsystem Set-up).

If you are upgrading your Sun Model 150 rather than starting with a system 'fresh from the box,' please heed the warning on the front page of this manual. Sun-1 systems which are upgrading must not attempt to install this release of the system until all Multibus memory has been removed or disabled, and — for an upgrade from a Xylogics 440 to a Xylogics 450 — cabling has been re-routed. Disk switch settings may also need to be reset. Make sure you have followed the procedures given in Sun's Upgrade Installation Guide before proceeding.

If you are adding boards and/or peripherals to an existing system, the chapter on Hardware Configuration and Expansion will give you the information you need.

2.1. Safety Precautions

DO NOT attempt procedures which are not described in this manual; if you do, you may void your warranty. Please refer all servicing not described in this document to qualified service personnel. If in doubt, contact your authorized Sun service representative.

Observe common-sense safety precautions as you would for any electrical or electronic equipment. Always disconnect the power cord before opening any system enclosure.

Even after power is disconnected from the workstation, very high voltages remain in the capacitors behind the picture tube. It takes about ten minutes for the capacitors to lose their charge.

The following sections contain set-up and configuration specifications for the Sun Rackmount Workstation. Consult the additional hardware manuals supplied with your workstation (and with user-supplied Multibus accessory boards, if any) for further information and precautions.

2.2. Environmental Considerations

This section describes the environmental requirements for the Sun Model 150U.

Attention: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio
communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

2.2.1. Physical Environment

The Sun Model 150U is manufactured for the following physical environment:

Table 2-1: 150U Physical Environment

<table>
<thead>
<tr>
<th></th>
<th>Operating</th>
<th>Non-Operating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambient Temperature</td>
<td>10 °C - 40 °C</td>
<td>-40 °C - 65 °C</td>
</tr>
<tr>
<td>Humidity (Non-condensing)</td>
<td>5% - 90%</td>
<td>5% - 90%</td>
</tr>
<tr>
<td>Altitude</td>
<td>10,000 feet</td>
<td>40,000 feet</td>
</tr>
</tbody>
</table>

2.2.2. Power Drain Requirements

Model 150U rackmount systems are supplied with a 1000 watt, four-output power supply, which delivers +5, -5, +12, and -12 volts. The power supply is rated as follows:

Table 2-2: Model 150U Power Consumption

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Maximum Current in Amps</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5</td>
<td>150.0</td>
</tr>
<tr>
<td>-5</td>
<td>5.0</td>
</tr>
<tr>
<td>+12</td>
<td>10.0</td>
</tr>
<tr>
<td>-12</td>
<td>10.0</td>
</tr>
</tbody>
</table>

It is unlikely that a full card cage will exceed the ratings of this supply. The table which follows shows the power consumption for Sun-supplied boards. Boards documented are current with this release.
Table 2-3: Power Consumption of Boards

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>BOARD</th>
<th>AMPS 5V</th>
<th>TOTAL WATTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>370-1012</td>
<td>Xylogics SMD Disk Controller Board</td>
<td>8.0</td>
<td>45</td>
</tr>
<tr>
<td>370-1021</td>
<td>Sky Floating Point Processor</td>
<td>4.0</td>
<td>20</td>
</tr>
<tr>
<td>501-0289</td>
<td>Color Display Controller</td>
<td>6.0</td>
<td>36</td>
</tr>
<tr>
<td>590-0059</td>
<td>Black and White Display Controller</td>
<td>5.0</td>
<td>25</td>
</tr>
<tr>
<td>590-0217</td>
<td>Interphase SMD Disk Controller Board</td>
<td>4.0</td>
<td>23</td>
</tr>
<tr>
<td>590-0288</td>
<td>10Mbit Ethernet Board</td>
<td>5.0</td>
<td>31</td>
</tr>
<tr>
<td>590-0502</td>
<td>1/2-inch Tape Controller Board</td>
<td>4.0</td>
<td>20</td>
</tr>
<tr>
<td>590-0526</td>
<td>1/4-inch Tape Controller Board</td>
<td>3.0</td>
<td>15</td>
</tr>
<tr>
<td>590-1007</td>
<td>Sun-2 CPU Board, Keyboard, and Mouse</td>
<td>6.0</td>
<td>30</td>
</tr>
<tr>
<td>590-1013</td>
<td>1M Memory Board</td>
<td>3.0</td>
<td>15</td>
</tr>
<tr>
<td>590-1013</td>
<td>Sun-2 (low power) Memory Board</td>
<td>2.5</td>
<td>13</td>
</tr>
</tbody>
</table>

Note that the keyboard and mouse draw their power from the CPU board.
Review the above list carefully to determine exactly what configuration of boards you can have in the card cage. If you get boards from other suppliers, make sure you obtain the current power consumption information from them, so that you can factor that information in with the list above. Overloading the rated capacity of the power supply can cause system malfunction (which often shows up as "glitches" on the video screen), and may damage the power supply.

2.3. Receipt and Unpacking Instructions

The basic Model 150U is shipped in two separate cartons: one contains the Sun Workstation video monitor on its pedestal, and the second contains a rack-mountable metal enclosure with the workstation card cage and power supply. Each subsystem ordered comes in its own carton — see the chapter Subsystem Set-up for unpacking instructions. You may also find yourself with a few small boxes: the Sun System documentation is shipped in its own package, as are the brackets and power supply for the 169-MByte disk.

When you receive your shipment, inspect all shipping cartons immediately for evidence of damage. If any shipping carton is severely damaged, request that the carrier's agent be present when the carton is opened. If the carrier's agent is not present when a carton is opened and the contents are found to be damaged, keep all contents and packing materials for the agent's inspection.

The following two subsections describe how to unpack the Model 150U monitor and card cage enclosure. Mounting instructions for the card cage enclosure are also given.
2.3.1. 150U Monitor — Unpacking

Note that the shipping weight of the Model 150U monitor and its shipping carton is approximately 80 pounds. The weight of the monitor itself is about 70 pounds.

1. To open the monitor's shipping carton, cut the binding straps on it with a knife or scissors. The shipping carton is in the form of a square 'tube' with two lids on it, one on the bottom, and one on the top. The monitor is cased in styrofoam packing materials inside the lids.

2. Remove the top lid from the 'tube', and you will find the keyboard, the power cord, the distribution tape(s), and an envelope containing manuals in the recesses in the top of the packing material. Remove these items.

3. Lift out the rubber and styrofoam packing material (the fit is fairly tight; try rocking it out). At this point you may find a mouse and mouse pad boxed and tucked along the side of the monitor; if you ordered an 84-MByte disk subsystem, these will be packed with it instead.

4. Remove the square 'tube' from its bottom lid. Don't try to lift the monitor out of the shipping carton: it's too heavy. When you remove the 'tube' from the bottom lid, the monitor is left standing in the bottom piece of packing material. Two people can then lift the monitor out of the bottom shipping material, remove the plastic wrapping, and place the monitor on its working surface.

We recommend that you save the salvageable shipping carton and packing material for future use in case the product must be reshipped.
2.3.2. 150U Card Cage Enclosure — Unpacking and Mounting

Note that the shipping weight of the Model 150U enclosure and its shipping carton is approximately 60 pounds. The weight of the enclosure itself is about 50 pounds.

1. Open the enclosure's shipping carton as you did the monitor's.
2. Remove the top lid from the shipping 'tube', and you will find a cardboard piece with the door for the enclosure; power cord; and extension keyboard, mouse, and video cables on it. Take these items off and set them aside.
3. Pull up the cardboard and styrofoam packing material, and you will expose the unit, covered in plastic.
4. Remove the square 'tube' from its bottom lid. Don't try to lift the enclosure out of the shipping carton: it's too heavy. When you remove the 'tube' from the bottom lid, the unit is left standing in the bottom piece of packing material. Two people can then lift the enclosure out of the bottom shipping material, using the handles on its sides.
   We recommend that you save the salvageable shipping carton and packing material for future use in case the product must be reshipped.
5. The M150U enclosure comes supplied with a set of clips containing captive nuts, and a set of bolts for bolting the unit into a standard 19-inch rack. To mount the enclosure, clip the clips into the holes in the vertical metal rail in the rack. Then commandeer two cohorts to position the unit in the rack while you secure it by the bolts provided.
6. Lastly, slip the enclosure front door on. If you mount the unit with the door on, you run the risk of shearing off the front light and switch.

2.4. Set-up

The following subsections describe how to cable up your Sun Model 150U and its basic peripherals.

CAUTION: Before plugging in the power cord of any component of your Sun system, be sure that the line power supply voltage and frequency are as specified on the back panel of your workstation. Use only three-prong (grounded) outlets. Always disconnect power before opening any system enclosure or servicing any system component. All servicing should be performed by qualified personnel.

2.4.1. Keyboard and Mouse

The Sun keyboard should be plugged into the connector labelled "KEYBOARD" on the card cage back panel. If you wish to use the keyboard as your console input device (as is normally done) you must power-on the workstation AFTER plugging in the keyboard. The Sun resident PROM monitor will try to use serial port A for input if it does not find an attached keyboard.
Plug the mouse into the "MOUSE" connector on the back panel.
2.4.2. Ethernet

If you purchased a system with Ethernet, your system is shipped with the Ethernet Controller Board installed in your workstation’s Multibus card cage. You will receive a package containing the transceiver and transceiver cable necessary to complete implementation of the Ethernet for a single workstation, and an Ethernet Controller Reference Manual. The coaxial cable necessary to implement a network with multiple machines may be purchased separately from Sun.

Setting up an Ethernet with all Sun-supplied components is fairly straightforward:

1. Terminate the 50-ohm coaxial cable by placing a transceiver at both ends. Screw the cable into the transceiver N-connectors; cover the open N-connector on each transceiver with a terminator. In lieu of a transceiver, you may terminate the cable with two barrel connectors and terminators. Handle the coaxial cable with some care, as it is fragile; don’t run it in an area where it may be run over.

2. For each workstation, plug one end of the transceiver cable into the 15-pin D-connector on the transceiver, and the other end into the “ETHERNET 10 MBIT” connector on the card cage enclosure backplane.

Please note that there are certain cabling limitations which must be observed for proper Ethernet implementation:

<table>
<thead>
<tr>
<th>Ethernet Cabling Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIMUM contiguous length of coaxial cable segments</td>
</tr>
<tr>
<td>Distance between transceivers*</td>
</tr>
<tr>
<td>MAXIMUM length of transceiver cable</td>
</tr>
</tbody>
</table>

* Transceivers must be placed at 2.5 meter intervals along the coaxial cable. These intervals are marked on the cable with a black band. If you want to spread transceivers along a greater length of cable, make sure the distance between any two transceivers is a multiple of 2.5 meters.

For more information, see the Ethernet manual shipped with your system. If you have questions about repeaters or non-standard Ethernet configurations, please contact Sun Microsystems.

2.4.3. Asynchronous Serial Ports

You may attach a terminal, modem, printer, plotter, or other device which uses the RS-232 interfaces, to one of the serial port connectors labelled “RS232 A” and “RS232 B” on the back panel.
NOTE that older workstations (shipped prior to the 1.0 Release) have serial port A configured as a DCE (Data Communications Equipment) and serial port B configured as a DTE. In addition, only serial port A on the older workstations has full modem control. This means that external cables built for the Sun-1 or Sun-1.5 CPU may have to be modified to work correctly with the Sun-2 CPU, or a null modem cable (such as that supplied by Sun) may be required. See the *Asynchronous Serial Ports* section of the *Hardware Configuration and Expansion* chapter for wiring specifications.

The serial ports on the Sun Workstation were designed primarily for connecting output peripherals, such as printers and plotters, and can drive these output lines at speeds up to 19.2 Kbaud. Both ports provide the CTS, RTS, DTR, DSR, and DCD control lines required by some devices (such as modems) in addition to the transmit and receive lines; the ports generate DTR and RTS, and pay attention to DSR, CTS, and DCD. Both ports are also wired as DTE (Data Terminal direct connection of modems. Computers and other DTE devices can be connected using the null modem cable supplied by Sun. See the *Asynchronous Serial Ports* section of the *Hardware Configuration and Expansion* chapter for wiring specifications.
Chapter 3

Subsystem Set-Up

This chapter gives step-by-step instructions on how to unpack, mount, and cable up the various disk and tape subsystems which may be supplied with either the Model 100U or Model 150U. This chapter also gives directions on how to add a second Fujitsu M2312K (84-MByte unformatted), M2284 (169-MByte unformatted), or M2351 (474-MByte unformatted) drive to an existing Sun system. For details on configuring disk/tape controller boards, see the Hardware Configuration and Expansion chapter.

3.1. M2312K (Fujitsu 84-MByte) Disk

Your Sun Workstation may be supplied with a disk subsystem which is the Fujitsu Model M2312K disk drive. The Fujitsu disk drive uses the industry-standard SMD interface which is supported by either the Interphase SMD 2180 disk controller or the Xylogics 450 SMD disk controller supplied with your disk subsystem.

Note that if you are upgrading from an Interphase controller to a Xylogics controller, the DIP switches on the top surface of the disk drive must be re-set. For an Interphase controller, the proper settings are: on DIP Switch 2, switches 2, 4, 5, and 7 ON; switches 1, 3, and 6 OFF; on DIP Switch 3, switch 3 ON; all others OFF. For a Xylogics 440 or 450 controller, the proper settings are: on DIP Switch 2, all switches ON; on DIP Switch 3, switch 3 ON; all others OFF.

Two disk drives may be attached to a single SMD controller. Fujitsu disk drives are shipped in a box nested inside another box. The package containing the disk drive is wrapped in a plastic bag. To unpack, mount, and cable up the subsystem, follow the procedures below.

Note that the shipping weight of the 84-MByte Disk Subsystem and its shipping carton is approximately 85 pounds. The weight of the unit itself is about 75 pounds.

1. Open the disk's shipping carton by cutting its binding straps with a knife or scissors.
2. Open the outer shipping carton and remove the top styrofoam piece. This exposes packing foam containing several components (these vary depending on your system configuration): manuals, a power cord, Ethernet cables and transceiver, mouse and pad, and — for a 150U only — the mounting hardware for the disk subsystem. Remove these items and set them aside.
3. Remove the foam piece and the cardboard below it; you'll see four styrofoam corner pieces and another box. Remove the inner box from the shipping carton.

4. Peel off the tape which seals this box. DON'T CUT OPEN the box, or you may damage the finish on the unit. Remove the metal unit — this holds the 84-MByte disk subsystem, and may also contain a 1/4-inch tape drive.

   We recommend that you save the salvageable shipping carton and packing material for future use in case the product must be reshipped.

5. If you are mounting the subsystem in a 19-inch rack, proceed with the next three steps; otherwise, unlock the heads on the disk drive by following the procedure in step 7, and then proceed with cabling in step 9.

6. To mount the subsystem, first mount the slides at the chosen position in the rack.

7. Before carefully sliding the subsystem into the rack, you must unlock the heads on the disk drive. The head lock for the 84-MByte drive is on the bottom of the subsystem enclosure, so you must either tilt or elevate the enclosure to get at it. To unlock the heads, use a large screwdriver to rotate the white plastic knob on the bottom of the unit to the OFF position. ALWAYS lock the heads by rotating this knob to the ON position before moving the drive — even from table to table. More detailed instructions are in the Fujitsu Microdisk Drives CE Manual.

   ![Diagram of subsystem](image)

8. Now, slide the subsystem into the rack.

9. The disk subsystem is attached to either the Model 100U workstation backpanel or the Model 150U card cage enclosure backpanel via two flat ribbon cables. The (wider) control cable comes out of the back of the Fujitsu disk drive enclosure, and is labelled "DISK". This cable plugs into the connector labelled 'DISK COMMAND' on the Sun backpanel. The (narrower) data or "B" cable for each of up to four drives plugs into one of the four SMD-B connectors labelled 'DISK DATA #x' on the back panel, where x is 0 through 3. The first disk drive must be cabled to connector 'DISK DATA #0'; the second should be cabled to 'DISK DATA #1'.

   The disk controller installed in the card cage of both the Model 100U and the 150U is a Xylogics 450 controller which can control up to four drives; however, the system is configured for a maximum of two drives per controller. As shipped, each drive is configured as the first drive, that is, unit number 0, by setting section 4 of SW1 (accessible at the top of the drive) to ON and all other sections to OFF. SW2 and SW3 are set correctly as they come from Sun, and should be
left alone: SW3 should have section 3 OFF and all others ON; and SW2 should be properly set for your SMD controller type. If you have a Xylogics 450 SMD Disk Controller, SW2 should have all sections ON; if you have an Interphase 2180 SMD Disk Controller, SW2 should have sections 2, 4, 5, and 7 ON and all others OFF.

The Fujitsu subsystem is very simple to operate; it has only a power cord and an on-off switch. However, we suggest that you take some time to look through the installation and operation sections of the Fujitsu manual before using the drive, to understand general operating requirements and precautions.

3.1.1. M2312K (Fujitsu 84-MByte) Disk Expansion

If you are adding a second drive to your system, a bit of configuration work is necessary. Basically, you have to do the following:

**CAUTION:** Turn off the power and disconnect the power cord before opening up the Sun Workstation’s enclosure, doing cabling of any sort, or altering peripheral/board configurations.

1. Remove the four terminator packs from your expansion drive. The terminator packs are required only on the last disk drive in a daisy-chain configuration. If you ordered your expansion drive from Sun, the terminator packs should already have been removed when you receive the drive.

   The terminator packs are located on the drive circuit board inside the chrome chassis. To get to them, unscrew the two Phillips screws on the chrome cover (at the rear of the drive), and lift off the cover. The four terminator packs are the chips just behind the 60-pin command cable connector.

2. Set the new drive's logical address to unit number 1: set sections 1 and 4 on switch SW1 to ON, and leave the other sections OFF. To do this, you need to remove the outer (white metal) and inner covers of the subsystem housing: SW1 is accessible at the top of the drive itself. The other switches, SW2 and SW3, should be left as set originally; see the previous section for proper settings.

3. Daisy chain the drives together using the special cable which comes with the expansion drive. The cable has several female connectors along it (which connect to the disk command cables), and ends in a single male connector (which connects to the workstation backpanel). Connect each disk's command cable to one of the female connectors along the expansion cable, making sure that the drive with the terminator packs is the last drive in the chain. Then connect the male end of the expansion cable to the 'DISK COMMAND' connector on the Sun backpanel.

4. Plug the 25-pin data cable for drive unit #0 into the 'DISK DATA #0' connector on the Sun backpanel, and the data cable for drive unit #1 (your expansion drive, if you followed these procedures) into 'DISK DATA #1'.

No changes to the Xylogics 450 Disk Controller Board are necessary.

Before you mount and/or power up the drive, make sure you have followed the directions for unlocking the heads given in the previous section.
3.2. M2284 (Fujitsu 169-MByte) Disk

The 169-MByte disk subsystem is currently shipped in several boxes: the disk drive itself, the power supply for the subsystem, and the mounting hardware for a 19-inch rack each come packaged separately.

When you follow the unpacking and mounting procedures below, we recommend that you have the Fujitsu M228X Fixed Disk Unit, Customer Engineering Manual close at hand. This manual contains information essential to the process.

Note that the shipping weight of the 169-MByte Disk Subsystem and its shipping carton is approximately 90 pounds. The weight of the unit itself is about 80 pounds.

1. Open the 169-MByte disk's outer shipping carton by removing the four white plastic pins on the lid (squeeze together wings and pull), and then lifting off the lid.
2. When the lid is removed, the internal carton containing the disk subsystem is exposed. Lift out the internal carton, and take off the corner pads.
3. Cut open the internal carton, and remove the styrofoam packing material. This exposes the power and control cables, on the sides of the box; the front panel for the unit; and the subsystem itself, wrapped in a polyethylene bag.
4. With another person, lift the unit out of its bottom pad. Grasp the unit at the bottom to avoid damage to the sub-assemblies.

We recommend that you save the salvageable shipping carton and packing material for future use in case the product must be reshipped.
5. Before applying power to the subsystem, you must unlock the spindle, the actuator, and the motor on the disk drive. These are locked during shipment to protect the subsystem, and must be locked any time the unit is moved. For procedures, see the Fujitsu M228X Fixed Disk Unit, Customer Engineering Manual, sections 3.4.3, 3.4.4, and 3.4.5, respectively.

If you are mounting the subsystem, and you are mounting it such that you can get to its underside when it is mounted (necessary to unlock the spindle), you should mount the unit before unlocking the spindle, actuator, and motor for maximum protection. Go on with step 6 for mounting instructions.
6. To mount the subsystem, first mount the slides at the chosen position in the rack. Have one or two brawny folks around to help you lift the unit gently into position. When you are completely through with sliding the disk around, you can screw the pair of retaining bars onto the front of the subsystem, and snap on the front panel.
7. The M2284 disk drive is connected to its power supply unit via three cables.

**CAUTION:** Turn off the power and disconnect the power cord before opening up the Sun Workstation's enclosure, doing cabling of any sort, or altering peripheral/board configurations.

There are two power cables known as the 'A' (AC) and 'B' (DC) power cables, and there is a power control cable for power-up sequencing when multiple disk drives are installed. The 'A' cable (12-pin) connects to PW1 near the center rear of the power supply unit. The 'B' cable (15-pin) connects to PW2 near the top right rear of the unit. The power control cable (9-pin) connects to CN18, just to the left of PW2.
8. There are two switches of interest on the power supply unit. The POWER ON/OFF switch must be left ON all the time after installation. The LOCAL/REMOTE switch must be set to REMOTE at all times.

9. The power supply unit itself should be connected to a standard AC line. The brown wire in the line goes to the first Phillips screw of the series starting at the lower left rear of the power supply unit. The screw is 'under' the label "INPUT". The blue wire in the line goes to the second screw, under the label "AC". The green wire goes to "FG", frame ground.

10. After connecting the M2284 to its power supply, hook up its control and data cables as follows.

The M2284 is connected to the card cage backpanel via two flat ribbon cables. The wider of the cables is a 60-pin control cable. Plug the control cable into the socket labelled 'DISK COMMAND' on the backpanel. The narrower cable (25-pin) is the disk data cable; plug it into the socket labelled 'DISK DATA #x' on the backpanel, where x is 0 through 3. The first disk drive must be cabled to connector 'DISK DATA #0'; the second must be cabled to the 'DISK DATA #1' connector.

After connecting the flat ribbon cables to the backpanel of the Sun Workstation, connect the other ends of the cables to the disk drive. The 60-pin cable (connected to 'DISK COMMAND' on the Sun Workstation backpanel) must be inserted into location 'OM2' on the disk drive. The narrower cable (connected to 'DISK DATA #x' on the Sun Workstation backpanel) must be inserted into location 'OM3' in the disk drive. The red stripe on both cables should be to your left as you look at the disk drive from the back. Both cables should be passed through the cable-routing bracket located directly above their respective termination points on the disk drive. At the same time, the grounding straps from the ribbon cables' shield must be secured using the hardware on the routing bracket.

11. Make sure that the line-termination board is securely seated into 'OM1' on the disk drive (if you have more than one disk, the line-termination board is required only on the last disk drive in a daisy-chain configuration). The grounding strap must be fastened to 'TRM1-2' on the terminal block (marked "OV(2)").

For more information on disk interface and cables, refer to sections 3.5.3 and 3.5.5 of the Fujitsu 228X Customer Engineering Manual.

The disk controller installed in the card cage of both the Model 100U and the 150U is a Xylogics 450 controller which can control up to four drives; however, the system is configured for a maximum of two drives per controller. As shipped, each drive is configured as the first drive, that is, unit number 0, by setting all sections on switch SW1 at location J2/J1 on the CQFM PCB assembly to OFF.
3.2.1. M2284 (Fujitsu 169-MByte) Disk Expansion

If you are adding a second drive to your system, a bit of configuration work is necessary. Proceed as follows:

**CAUTION:** Turn off the power and disconnect the power cord before opening up the Sun Workstation's enclosure, doing cabling of any sort, or altering peripheral/board configurations.

1. Remove the terminator board from your first drive, and place it in your new drive, making sure it is centered and seated securely into ‘OM1’ on the second drive. ‘OM1’ is the 60-pin connector closest to you as you stand at the rear of the drive. The grounding strap must be fastened to ‘TRM1-2’ on the terminal block (marked “OV(2)”). The line-termination board is required only on the last disk drive in a daisy-chain configuration.

2. Set the new drive's logical address to unit number 1, by setting the first section on switch SW1 (at location J2/J1 on the CQFM PCB assembly) to ON, and leaving the other sections OFF. To do this, you need to get to SW1, which is on the circuit board in the PCB chassis (the metal housing at the left rear of the drive) which is closest to you if you stand at the rear of the drive. The board is labelled CQFM. SW1 is the 8-position DIP switch nearest the center of the board, just under the 60-pin edge connector. Proceed as follows:
   1. Shut everything down gracefully, remove the drive from the rack (if mounted), and lock the spindle. If you can get to the top of the PCB chassis — the metal housing at the rear of the drive, next to the motor — without removing the drive from its moorings, so much the better.
   2. Remove the metal plate from the top of the PCB chassis to expose the four circuit boards inside.
   3. Remove the CQFM assembly — the board closest to you as you stand at the rear of the drive.
   4. Make sure the first key on SW1 is set to ON, and all other keys are OFF. This sets the drive's logical address to #1. SW1 is the switch nearest the center of the board.
   5. Put everything back together again.

3. Daisy chain the drives together using the command cable which comes with the expansion drive. Run this cable from the 60-pin command cable socket (‘OM2’) on the drive with the line terminator (your expansion drive, if you followed the procedures above), to ‘OM1’ on the intermediate drive, and run this disk's command cable from ‘OM1’ to the ‘DISK COMMAND’ connector on the Sun backpanel.

4. Plug the 25-pin data cables in. The first disk drive (unit #0) must be cabled to connector ‘DISK DATA #0’; the second (unit #1, the expansion drive if you followed the procedures above) must be cabled to the ‘DISK DATA #1’ connector.

No changes to the Xylogics 450 Disk Controller Board are necessary.

Before you mount and/or power up the drive, make sure you have followed the directions for unlocking the spindle, actuator, and motor given in the Fujitsu M228X Fixed Disk Unit, Customer Engineering Manual, sections 3.4.3, 3.4.4, and 3.4.5, respectively.
3.3. **M2351 (Fujitsu 474-MByte) Disk**

The 474-MByte disk subsystem is shipped in a single carton, along with its cables. The drive itself is shipped in nested boxes, and wrapped in a polyethylene bag. To unpack, mount, and cable up the M2351, follow the procedures below.

We recommend that you have the Fujitsu **M2351A/AF Mini-Disk Drive CE Manual** close at hand as you follow the instructions; it contains essential information.

1. Open the 474-MByte disk's outer shipping carton by cutting the binding straps with a knife or scissors, and opening the flaps.
2. When the box flaps are opened, the shock absorbers surrounding the internal carton are exposed. Remove the shock absorbers, and have AT LEAST two people lift out the internal carton.
3. Cut open the internal carton to expose the subsystem itself, wrapped in a polyethylene bag. Remove the two nylon straps.
4. With another person, lift the unit out of its bottom pad. Grasp the unit at the bottom to avoid damage to the sub-assemblies. We recommend that you save the salvageable shipping carton and packing material for future use in case the product must be reshipped.
5. Remove the cover on the drive. Loosen the two Phillips screws that hold the card cage upright, and gently pivot and lift the card cage to a horizontal position to expose the 'Vibration Preventative Block' (a piece of styrofoam) under the disk enclosure housing. Remove the 'VPB'. Return the card cage to its upright position, and tighten the screws.
6. Locate the rotary actuator ('VCM') at the rear of the drive, near the cables — there is a small label which says 'LOCK ---- FREE'; to the right of this label as you look at the back of the drive is a Phillips screw. Loosen the screw and rotate the locking lever to the unlocked position; tighten the screw in this second hole. ALWAYS lock the rotary actuator before moving the drive. For an illustration, see the Fujitsu manual, section 2.3.

**CAUTION:** Turn off the power and disconnect the power cord before opening up the Sun Workstation’s enclosure, doing cabling of any sort, or altering peripheral/board configurations.

7. As you stand at the rear of the drive, you'll see a small circuit board on your left-hand side. Make sure that the line-termination circuit board has its component-side facing away from you, and is securely seated into 'CNP42' on this board ('CNP42' is located next to 'CNP41', the 60-pin connector for the control cable). The grounding strap must be fastened to 'TBI' in the center of the circuit board. 'TBI' is a small silver square topped by a Phillips screw. DO NOT connect the grounding strap to 'TRM1' or 'TRM2'. The line-termination board is required only on the last disk drive in a daisy-chain configuration.
8. For mounting instructions, see the Fujitsu manual, section 2.4.
9. The M2351 is connected to the card cage backpanel via two flat ribbon cables. The wider of the cables is a 60-pin control cable. Plug the control cable into the socket entitled 'DISK COMMAND' on the backpanel. The narrow cable is the disk data cable, and must be plugged into the socket labelled 'DISK DATA #x' on the backpanel, where x is 0 through 3. The first disk drive must be cabled to connector 'DISK DATA #0'; the second should be cabled to 'DISK DATA #1'.
Internally, the 60-pin control cable is inserted into location ‘CNP41’ on the small circuit board near the left rear of the disk drive. The 25-pin data cable is inserted into location ‘CNP43’ on the circuit board near the left rear of the drive. The disk controller installed in the card cage of both the Model 100U and the 150U is a Xylogics 450 controller which can control up to four drives; however, the system is configured for a maximum of two drives per controller. As shipped, each drive is assumed to be the first drive, that is, unit number 0, by setting all sections of the drive address switch to OFF. The drive address switch is a 4-section DIP switch located on the small circuit board near the left rear of the drive.

**3.3.1. M2351 (Fujitsu 474-MByte) Disk Expansion**

If you are adding a second drive to your system, a bit of configuration work is necessary. Proceed as follows:

1. Remove the terminator board from your first drive, and replace it in your new drive, making sure it is centered and seated securely into ‘CNP42’ on the second drive. ‘CNP42’ is the 60-pin connector closest to the center of the circuit board as you stand at the rear of the drive. The grounding strap must be fastened to ‘TB1’ in the center of the circuit board. ‘TB1’ is a small silver square topped by a Phillips screw. DO NOT connect the grounding strap to ‘TRMI’ or ‘TRM2’. The line-termination board is required only on the last disk drive in a daisy-chain configuration.

2. Set the new drive’s logical address to unit number 1, by setting the first section on the drive address switch to ON, and leaving the other sections OFF. The drive address switch is a 4-section DIP switch located on the small circuit board near the left rear of the drive.

3. Daisy chain the drives together using the command cable which comes with the expansion drive. Run this cable from the 60-pin command cable socket (‘CNP41’) on the drive with the line terminator (your expansion drive, if you followed the procedures above), to ‘CNP42’ on the intermediate drive, and run this disk’s command cable from ‘CNP42’ to the ‘DISK COMMAND’ connector on the Sun backpanel.

4. Plug the 25-pin data cables in. The first disk drive (unit #0) must be cabled to connector ‘DISK DATA #0’; the second (unit #1, the expansion drive if you followed the procedures above) must be cabled to the ‘DISK DATA #1’ connector.

No changes to the Xylogics 450 Disk Controller Board are necessary.

Before you mount and/or power up the drive, make sure you have followed the directions for removing the ‘VPB’ and unlocking the rotary actuator given in the previous section.

**3.4. Quarter-Inch Magnetic Tape**

The Sun Workstation may be shipped with a quarter-inch magnetic tape cartridge subsystem as part of the Fujitsu disk drive. A tape cartridge can store up to 20 Megabytes of data.

The magnetic tape unit is controlled via a controller board manufactured by Sun Microsystems. The tape controller contains the hardware for managing the magnetic tape, and also contains 256 Kbytes of memory which is accessible on the Multibus (which must be disabled if you are using the board with a Sun-2 CPU board). The tape controller connects to the tape drive via a 50-pin flat ribbon cable. The tape controller cable comes out of the back of the Fujitsu disk enclosure, and is labelled “TAPE”. This cable plugs into the back panel connector labelled ‘TAPE COMMAND A’. 
The tape cartridge is in the correct orientation when the little window with the word "SAFE" in it is at the top left-hand corner of the cartridge. Firmly push the cartridge into the slot in the front of the tape drive. It will suddenly go all the way in with a definite "click". The tape is now ready for use.

To protect the tape from being written on, turn the arrow that is on the left of the window marked "SAFE" until the arrow points at the word "SAFE" — the tape cannot be written when the arrow is in this position. Turning the arrow 180 degrees so that it points away from the word "SAFE" allows writing on the tape.
3.5. Nine-Track Magnetic Tape

The Sun Workstation may be shipped with a Control Data Streaming Tape Unit (STU) in the 92180 family of tape drives. This tape unit is an industry-standard tape unit which can operate at either 25 inches per second start/stop mode, or at 100 inches per second streaming mode. The magnetic tape unit is controlled via a Computer Products Corporation TAPEMASTER tape controller.

To unpack, mount, and cable up the nine-track magnetic tape subsystem, follow the procedures below.

Note that the shipping weight of the 1/2-inch Streaming Tape Unit and its shipping carton is approximately 117 pounds. The weight of the unit itself is about 100 pounds.

1. Cut open the top of the Streaming Tape Unit's shipping carton with a short-bladed knife (so as not to damage the finish on the unit).
2. Fold back the flaps of the shipping carton to expose the inner tray, with manual and installation kit taped on top. Remove these items.
3. Remove the tray, and you will see the top of the unit with attached shipping frame.
4. Have a cohort grasp one side of the frame while you grasp the other; remove the unit from the carton and bottom inner tray, and place it on a table top.
   We recommend that you save the salvageable shipping carton and all packing material for future use in case the product must be reshipped.
5. Carefully cut and remove the non-metallic band securing the Streaming Tape Unit's door.
6. Remove the filler blocks which are between the upper and lower PC board rear-mounted hinges.
7. Remove the door support blocks from under the door assembly. Leave the one-inch frame support block in place until the unit is ready for rack mounting. If you're not mounting the unit, simply leave the shipping frame on.
8. Remove the filler block which is between the shipping frame and underside of the PC boards by carefully pressing downward on the block, and sliding it backward and out from under the PC boards.
9. Remove the door stud from the installation kit. Screw the threaded end into the receptacle block inside the dust cover door. The tape unit will not run if the stud is not in place to engage the interlock switch.
10. To mount the unit in a 19-inch rack, first bolt the two hinge assemblies and stiffener bar to the rack. If you're not mounting the unit, skip to step 18 for cabling instructions.
11. Support the Streaming Tape Unit in a vertical position on the shipping frame, so that you can remove the four screws and detach the unit from the frame. After unmounting the unit, make sure your cohort is around to help you lift and maneuver it — definitely a two-person operation.
12. When the frame is removed, install the two hinges on the top of the unit with the provided screws and lockwashers.
13. Position the Streaming Tape Unit onto the mounting hinges. The unit must be perpendicular to the rack for the hinges to be mated.
14. Place the unit in a closed position. Mark the area at which the adjustable pawl fastener of the unit contacts the mounting rail.

15. With the unit in an open position, install the bumper assembly into the mounting rail approximately one to two inches above the point at which the pawl fastener contacts the mounting rail.

16. Adjust the bumper assembly so that the tape deck is parallel to the mounting rack when the Streaming Tape Unit is in a closed position.

17. Place the unit in a closed position and secure it with the adjustable pawl fastener. Continue turning the pawl fastener clockwise until the unit is secure against the bumper assembly.

18. The TAPEMASTER tape controller inside the Sun Workstation card cage connects to the tape drive via two 50-pin flat ribbon cables. Look at the back of the tape drive: there is a large square circuit board with two edge connectors on it. Connect the top edge connector ("J1") via its 50-pin ribbon cable to the socket labelled 'TAPE COMMAND A' on the backpanel of the card cage enclosure. Connect the bottom ("J2") edge connector via its 50-pin ribbon cable to the socket labelled 'DISK/TAPE COMMAND B' on the backpanel of the enclosure.

Please read the Control Data STU Reference Manual for more information on the nine-track drive (tape).
Chapter 4

Installing UNIX for the First Time

This chapter describes how to install the UNIX system for the very first time on the Sun Workstation. We assume that you are starting with a Sun Workstation fresh from the box, bare disk(s), and the distribution software either on a nine-track half-inch tape or on two quarter-inch cassette tapes. This chapter guides you through the installation procedure.

CAUTION: If you are upgrading your Sun Model 100 or 150 rather than starting with a system 'fresh from the box,' please heed the warning on the front page of this manual. Sun-1 systems which are upgrading must not attempt to install this release of the system until all Multibus memory has been removed or disabled, and — for an upgrade from a Xylogics 440 to a Xylogics 450 — cabling has been re-routed. Disk drive switches may also need to be reset. Make sure you have followed the procedures given in Sun's Upgrade Installation Guide before proceeding.

CAUTION: If you are upgrading your system software, rather than beginning for the first time, read the Upgrading System Software chapter BEFORE proceeding here. The chapter gives procedures for saving and rebuilding your existing root and /usr file systems.

If you are installing UNIX on a machine without a tape drive, see the procedures in the next chapter, Installing UNIX on Systems Without Tape Support.

We begin with a description of what is on the distribution tape. This is followed by an overview of the installation procedure, including some non-trivial remarks on device abbreviations, what to do when you make mistakes while performing the procedure, and what facilities are and are not available to you during installation. Then we do a walkthrough of an actual installation.

4.1. What is on the Distribution Tape?

The software needed to load the UNIX system is contained either on a 9-track magnetic tape, or on two quarter-inch tape cartridges, distributed with the system. The tape(s) contains eleven files, as follows:
Table 4-1: Contents of Distribution Tape(s)

<table>
<thead>
<tr>
<th>1/2&quot; Tape File Number</th>
<th>1/4&quot; Tape File Number</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>A general purpose boot program which knows how to boot from the various devices that can be attached to the Sun Workstation. The PROM monitor boots this general purpose boot program.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>A copy of a program called diag. <strong>Diag</strong> is used to format and label disks.</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>A stand alone <em>copy</em> program which can copy from specified sources to specified destinations.</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>An image of a miniature version of the root file system, which contains enough machinery to get something up and running.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Copyright file.</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>The complete root file system for the UNIX operating system. This is on the tape in <strong>dump</strong> format and is loaded in by the <em>tar</em> utility.</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>A <em>tar</em> format dump of the <em>/usr/man</em> (online manual sources), <em>/usr/games</em> (games), and <em>/usr/demo</em> (demonstration programs) directories. These directories are optional and need only be loaded onto your system if they are specifically required. Note that in a system which is being formatted for diskless users, these directories may be too big to fit into the public partition. You can exclude parts of the dump, such as demonstration programs, if you wish.</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Copyright file.</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>Copyright file. Note that this is file 8 on the nine-track distribution tape; if the distribution is on a quarter-inch cartridge tape, this file is actually the first file on the second cartridge.</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>An image of the <em>/usr</em> file system that was placed on the tape by the <em>tar</em>(1) command. The <strong>setup</strong> program transfers this file system to the disk.</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>Copyright file.</td>
</tr>
</tbody>
</table>
4.2. Overview of the Installation Procedure

The object of this exercise is to load the UNIX system from the magnetic tape onto the disk sub-system of your Sun Workstation.

Please note: If you are installing a network disk server, you must have a tape drive on the server machine.

The major steps in the procedure for loading the UNIX system are:

1. If you have an Ethernet board in your system, determine network information for use later in installation. If you are installing a standalone workstation, you will need your complete Internet address (network number and host number). If you are installing a cluster of workstations, you will need each client's hardware Ethernet address as well.

2. Use the resident PROM monitor to boot the general purpose bootstrap program from the magnetic tape.

3. Use the bootstrap program to load the diag utility from the magnetic tape.

4. Format and label the disk with the diag utility.

5. Use the bootstrap program to load the stand alone copy utility from the magnetic tape.

6. Use the stand alone copy utility to copy the mini UNIX root file system from the tape onto the swap area on the disk.

7. Boot the mini UNIX operating system from the swap area.

8. Restore the full root file system using the str utility.

9. Boot the UNIX operating system in single-user state.

10. Run the setup program to configure your system.

11. Boot the full UNIX system.

12. Reconfigure the system kernel. This step is mandatory for systems with only 1 MByte of memory, and highly recommended for systems with more.

13. Load the manuals, demos, and games directories if desired.

14. Continue with network configuration and installation of utilities such as the mail system and uucp, configuration of a line printer, etc., by following procedures in the System Set-up and Operation chapter.
The procedures described below walk you through what you actually have to do. In all the examples, what you type is shown in boldface text like this. Everything shown in boldface should be typed exactly as it appears. Where parts of a command are shown in italic text like this, they refer to a variable which you have to substitute from a selection; it is up to you to make the proper substitution.

Two very important variables in the examples are disk, which should be replaced with the UNIX device name for your disk:

Table 4-2: UNIX Disk Device Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>xy</td>
<td>Xylogics 450/440 disk controller</td>
</tr>
<tr>
<td>ip</td>
<td>Interphase SMD-2180 disk controller</td>
</tr>
</tbody>
</table>

and tape, which should be replaced with the abbreviation for your tape drive:

Table 4-3: UNIX Tape Device Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>mt</td>
<td>Nine-track magnetic tape</td>
</tr>
<tr>
<td>ar</td>
<td>Archive quarter-inch tape</td>
</tr>
</tbody>
</table>

For example, a common configuration would load from the Archive quarter-inch magnetic tape cartridge, onto a Fujitsu disk driven from a Xylogics 450 disk controller. In this case, tape would be replaced by ar (Archive tape) and disk would be replaced by xy (Xylogics disk) everywhere.

If (when) you make a typing mistake during the installation procedure, you can use the DEL key to erase and back up over incorrect characters you just typed. You can use the control-U key to kill the entire line you just typed, and start typing a new line.

If you need to get back to the monitor for any reason, you can abort by typing 'SET-UP A' (hold down the SET-UP key while typing 'A') or 'ERASE-EOF A' on your workstation keyboard at any time. On a terminal, the BREAK key generates an abort.

During a large part of the installation procedure, you are using the single-user UNIX system. Please note that the shell you are talking to in this state is the Bourne Shell, not the C Shell — this means that there is no history facility available.
4.3. Installation Walkthrough

4.3.1. Determining Network Information

This step is only necessary if you have an Ethernet board in your Sun system. You can skip to the Loading the Tape subsection just below if you don’t have Ethernet.

4.3.1.1. Procedure for New Systems

If you are installing a standalone workstation on a network or a cluster of networked workstations, there are three pieces of information needed during the phase of installation which establishes network configuration which are best obtained ‘ahead of time’:

1) your network number,
2) the hardware Ethernet address for each client, and
3) the host number for each client.

The Network Number

The network number is an arbitrary value used to uniquely identify multiple networks. If you have been assigned a network number by ARPA use that number. If not, or if the entire network number business has you confused, use Sun’s default network number of 192.9.200.

Client’s Hardware Ethernet Address

The hardware Ethernet address of each client is taken from the ID PROM on the Sun-2 CPU Board. The address is a 6-byte hexadecimal value; each byte of the address is separated by a colon. A typical Ethernet address is “8:0:20:0:14:76”.

If you are installing a cluster of workstations, you will need to have each client’s hardware Ethernet address at hand. To obtain it, power on the client’s workstation. You will see the Sun logo, and a message like:

Self Test completed successfully.
Sun Workstation, Model Sun-1/100U or Sun-1/150U, Sun-1 keyboard
ROM Rev N, some_number_MB memory installed
Serial #some_number, Ethernet address 8:0:20:1:1:A2
Auto-boot in progress . . .
abort by typing L1-A or BREAK here
Abort at some address
>

The entire six bytes of the displayed “Ethernet address” must be used whenever you are prompted for the client’s address; copy them down.

Host Numbers

Each node on a given network is uniquely identified with a host number. The size of the host number varies with the network number, but in all cases the host number can be from
During network configuration, the setup program asks if you want host numbers automatically assigned. If you answer yes, the server is given a host number of one (1) and the clients are assigned 2, 3, etc. This is fine if you are installing a new network with no existing nodes attached to it. However, if you are adding workstations to an existing network you should assign the host numbers yourself. Remember that the host number must be unique across the entire network. If an existing node on the network has a host number of 10, do not assign 10 to a new node.

To find the host numbers for existing nodes, take a look at the /etc/hosts file on one of the existing nodes. The entries should look something like this:

```
192.9.200.1 alpha
192.9.200.2 beta
192.9.200.3 gamma
192.9.200.4 delta
192.9.200.5 epsilon
```

The last component of the network address is the host number. For instance, epsilon's host number is 5.

Before beginning installation, have this address information at hand.

### 4.3.1.2. Procedure for Existing Systems

If you have an existing network (of older Suns or mixed machines) that you do not intend to upgrade, compatibility with a 1.1 network may need addressing. Proceed with the procedures given above for any machines you are installing or upgrading, and when you are finished, see the Making 1.1 Networks Compatible with Existing Networks section in the System Set-up and Operation chapter.

### 4.3.2. Loading the Tape

**PLEASE NOTE:** If you are installing a network disk server, you must have a tape drive on the server machine.

In the case of the nine-track tape, load the tape onto the tape drive and put the drive on-line. In the case of a quarter-inch tape, insert the tape cartridge into the drive as described in the section, Quarter-Inch Magnetic Tape Subsystem in the Subsystem Set-up chapter.

### 4.3.3. Loading the Bootstrap Program

Power up the Sun Workstation (if you haven't already done so). You should see the PROM monitor's sign on messages. Stop the auto boot immediately by aborting — type 'SET-UP A' (hold down the SET-UP key while typing 'A') or 'ERASE-EOF A' on a workstation keyboard, or BREAK on a terminal keyboard — and then you should see the monitor's prompt, which is a > sign:
Self Test completed successfully.

Sun Workstation, Model Sun-1/100U or Sun-1/150U, Sun-1 keyboard
ROM Rev N, some_number_MBytes memory installed
Serial #some_number, Ethernet address n:n:n:n:n

Auto-boot in progress . . .
[ abort by typing 'SET-UP A', 'ERASE-EOF A', or BREAK here ]

Abort at some address
>

You boot the general purpose bootstrap program from the tape by typing a b (for boot) command to the monitor. The monitor echoes the boot command back to you, with the parameters filled in. A default boot command looks like this:

```
> b tape()
Boot: tape(0,0,0)
```

Remember: tape here should be replaced by mt for the nine-track tape, or ar for the Archive quarter-inch tape.

The bootstrap program displays a sign on message, and then displays a prompt sign of Boot: to indicate that it is ready to accept commands. Whenever the bootstrap program is ready for a command, it displays the Boot: sign at the start of the line. So, the initial dialogue with the monitor and the bootstrap program would look like this:

```
> b tape()
Boot: tape(0,0,0)
Boot:
```

Now the standalone boot program is in control. All programs that it loads in eventually return control to it. When the bootstrap program loads another program, it displays some numbers which are the sizes of the different portions of the program it loaded.

4.3.4. Using the Diag Utility

Next, you use diag (the interactive disk format utility) to format and label your disk(s). diag has two 'phases': in the first, it prompts for information about the type of disk drive and controller it is working with, and essentially 'configures' itself to work with that disk and controller; and in the second, it responds to commands like the format and label commands we use during this installation phase. If you have multiple drives/controllers, go all the way through the format and labelling phase for the first drive on your first controller, and then loop back to begin the cycle for your next device — we give directions for this at the appropriate time.

You boot diag from tape by typing a bootstrap command like this:

```
Boot: tape(0,0,1)
```

where tape is replaced with mt for the nine-track tape, or ar for the Archive quarter-inch tape. The bootstrap program should boot diag from the magnetic tape. When diag starts up, it displays a sign on message:
Disk Initialization and Diagnosis

When asked if you are sure, respond with 'y' or 'Y'

Diag first asks you a series of questions about the disk you will be using. The answers to these questions 'configure' diag to work with that specific hardware. The first thing diag wants to know is the type of disk controller to use:

specify controller:
   0 - Interphase SMD-2180
   1 - Xylogics 440 (prom set 926)
   2 - Xylogics 450
   3 - Adaptec ACB 4000 - SCSI

which one?  2

In this example, we specified that the controller is type 2 for a Xylogics 450 SMD controller.

Diag's next question is what address this controller occupies on the Multibus. When you give diag the address (see table below)\(^2\), it echoes the address back to you:

Specify controller address on the Multibus (in hex): select the address from the table below

Device address: whatever address you selected

The table below shows the default addresses for disk controllers.

<table>
<thead>
<tr>
<th>Controller Type</th>
<th>Address (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st Controller</td>
</tr>
<tr>
<td>Interphase SMD-2180</td>
<td>40</td>
</tr>
<tr>
<td>Xylogics 450/440</td>
<td>ee40</td>
</tr>
</tbody>
</table>

Next, diag wants to know the unit number of the disk:

Which unit? 0

"0" is the correct response for the first drive on the controller specified above. "1" is the unit number for the second drive on a controller, and so on.

Now diag wants to know the type of disk drive you are working with. Diag displays a menu of the different disks. When you have specified which disk drive you wish to operate on, diag displays a table of the physical data about that disk, which includes the number of cylinders, number of alternate cylinders, number of heads, and number of sectors per track. In this example, we are formatting the Fujitsu M2312K (Sun "D84") 84-MByte disk:

---

\(^2\) Note that if you have one Xylogics SMD Controller and one Interphase Controller in your system, you cannot use the default addresses (as the Interphase Controller sees only four bytes). The Interphase must be configured to be at address 48, and the Xylogics at ee40. See the Hardware Configuration and Expansion chapter for board configuration procedures.

4-8 Revision H of 12 March 1984
Specify drive:
0 - Fujitsu-M2312K (Sun D84)
1 - Fujitsu-M2284 (Sun D169)
2 - Fujitsu-M2351 Eagle (Sun D474)
3 - Fujitsu-M2294
4 - CDC-Lark-cartridge
5 - CDC-Lark-fixed
6 - CDC-LarkII-cartridge
7 - CDC-LarkII-fixed
8 - CDC-9768
9 - CDC-9730-160
10 - CDC-9730-80
11 - CDC-9730-24
12 - CDC-9730-12
13 - Ampex-Scorpio
14 - Ampex-Capricorn
15 - Other
which one? 0
ncyl 586 acyl 3 nhead 7 nsect 32 interleave 1
status:
drive status ready

At this point, diag knows all it needs to know about the disk, and it displays its 'second phase' prompt:

diag>

One of the responses to this prompt can be the h (for help) command. If you use the h command, diag displays a list of its commands.

4.3.4.1. Formatting the Disk

Next, you use diag's disk formatting function. Begin by clearing any outstanding errors (especially important if you are using an Interphase disk controller) with the clear command:

diag> clear
clear
diag>

Now, type the format command. Diag warns you that formatting a disk destroys all information which might already be stored on that disk, and asks for confirmation before going ahead and doing the job:

diag> format
format/verify, DESTROYS ALL DISK DATA!
are you sure? y
# of surface analysis passes? 5

When formatting a disk for the first time, five surface analysis passes is a reasonable number. The surface analysis phase of the formatting is for detecting bad spots on the disk and mapping them to alternate places. The format phase takes about 30 minutes for one surface analysis
pass on an 84-MByte disk, using an Interphase SMD-2180 disk controller, and slightly less time using a Xylogics 450 or 440 controller. It is well worth taking the time to do this when formatting a new disk. \textit{Diag} then formats the disk, and displays the current cylinder number as it formats each cylinder. You can use just one surface analysis pass if you are reformatting a disk that has been formatted before, and is known not to have any bad spots.

At the end of the format process, \textit{diag} displays a message telling you that it has finished, then tells you to use the label command to label the disk:

\begin{verbatim}
cyl 588 format complete - 0 bad sector(s)
Use the label command to label the disk
\end{verbatim}

\texttt{diag>}

This is the next step.

4.3.4.2. Labelling the Disk

A disk must be labelled after it has been formatted. The purpose of a label is to record (in a well known place on the disk) information about how the disk is divided into partitions for different purposes such as paging space and file systems.

\textit{Diag} labels a disk in response to the \texttt{label} command. Here is an example of using \texttt{label} to write a label on the disk which was formatted in the discussion above. When you give the command to \texttt{diag}, it asks if you want to use the logical partition map that is 'built in' to the program:

\begin{verbatim}
diag> label
label this disk ...
OK to use logical partition map 'Fujitsu-M2312K (Sun D84)' ? y
Are you sure you want to write? y
\end{verbatim}

Then, when \texttt{diag} has labelled the disk, it verifies the label it has just written. There is in fact a separate verify command, but \texttt{diag} automatically does a verify as part of the labelling process:

\begin{verbatim}
verify label
id: <Fujitsu-M2312K cyl 588 alt 3 hd 7 sec 32>
Partition a: starting cyl=0, # blocks=15884  [#s vary with disk controller]
Partition b: starting cyl=67, # blocks=33440
Partition c: starting cyl=0, # blocks=131264
Partition g: starting cyl=208, # blocks=81760
\end{verbatim}

\texttt{diag>}

This is the end of the disk formatting and labelling process. If you have a second drive/controller, you now get back to the beginning of \texttt{diag}'s first phase by responding to \texttt{diag}'s prompt with the \texttt{diag} command:

\begin{verbatim}
diag> diag
\end{verbatim}

If you have only one disk on your controller, or if you are done formatting and labelling all of your disks, you are ready to continue with the next phase of installation. Get back to the bootstrap program by typing the \texttt{q} (quit) command, and the bootstrap displays its prompt sign again:

\begin{verbatim}
diag> q
\end{verbatim}

Boot:
4.3.5. Loading the Mini UNIX System

After your disk(s) are formatted and labelled, the next job is to load in the mini UNIX root file system from the tape. This is done by loading in the standalone copy program which is the third file (#2) on the tape(s). The copy program prompts you for the source and destination of the copy, as shown here. (As always, the example assumes that tape stands for a tape device — such as — and that disk stands for a disk device — such as xy for the Xylogics disk controller, or ip for the Interphase disk controller):

```
Boot: tape(0,0,2)
20480+4096+113384 bytes [ #s vary with system level ]
Standalone Copy
From: tape(0,0,3)
To: disk(0,0,1)
```

Note that after you have typed in the tape information for the ‘From’ prompt, the copy program moves the tape for about ten seconds before displaying the ‘To’ question. Copying in the mini-UNIX system takes about three minutes, using a half-inch tape, and about six minutes, using a quarter-inch cartridge. At the end of the copy, the copy program returns control to the bootstrap program:

```
Copy completed
Boot:
```

4.3.6. Booting the Mini UNIX System

Next, you tell the bootstrap program to boot the mini-UNIX system from the disk. Because this mini-UNIX system boot is an unusual one, you must specify the -a (for ask me) option on the boot command, and also the -s (come up single user) option, as follows:

```
Boot: disk(0,0,1)vmunix -as
243712+30720+49768 start 0x4000 [ #s vary with system level ]
[ ... about a dozen lines of configuration messages ... ]
```

The mini-UNIX system displays some messages about the configuration of the system on which it is running, and finally displays a message asking for its root file system. The root file system at this stage is "disk0*", which has a special meaning to the small UNIX system. Since this notation looks ambiguous, let me specify: if you have a Xylogics disk controller, your root device is xy0*; if you have an Interphase controller, it is ip0* — the asterisk is part of the device name:

```
root device? disk0*
```

At this point the system may gently remind you to reset the system date and time:

```
WARNING: clock gained 16845 days -- CHECK AND RESET THE DATE!
```

You can do this when your prompt returns, with the date(1) command.
4.3.7. Setting the Date

Now the mini-UNIX system starts up and displays its prompt sign: a # character. Set the correct system date and time by using date(1):

```
    # date ymmmdhhmm[.ss]
```

yy here is the last two digits of the year; mm specifies month; dd designates day of the month; hh is hour (on a 24-hour clock); the next mm is minutes elapsed; and the optional .ss specifies seconds. The system echoes the date set back to you.

4.3.8. Loading the Root File System

Next, tell the mini-UNIX system to extract the real root file system from the tape with the following sequence:

```
    # disc=disk tape=tape xtr
```

Replace disk with xy for the Xylogics disk controller, or ip for the Interphase disk controller. Replace tape with mt for the nine-track tape, or ar for the Archive quarter-inch tape. The xtr command is a shell script which creates a root file system and extracts it from the tape. The xtr shell script displays a lot of messages which look something like this:
At this point, the UNIX system has a complete root file system. There is as yet no /usr file system, and that is part of the next job.

4.3.9. Booting the UNIX System

Now you must escape back to the monitor by typing 'SET-UP A' (hold down the SET-UP key while typing 'A') or 'ERASE-EOF A' on a workstation keyboard, or BREAK if you are using a terminal. The monitor responds by displaying a message to the effect:

Abort at some address
When you see the monitor > sign, you boot the UNIX system by typing a b command to the monitor, and the monitor responds by filling in the full details of the boot command. We must boot the UNIX system with the -s option (come up single user), because there is no user file system as yet:

```
> b vmunix -s
Boot: disk(0,0,0)vmunix -s
Load: disk(0,0,0)boot
```

The UNIX system startup procedure is automatic. It displays a progress report as it goes through the initialization sequence.

```
Boot: disk(0,0,0)vmunix
219136+ 28672+ 29088
Sun UNIX 4.2 (GENERIC) #145: Tue Feb 21 20:35:13 PDT 1984
Copyright (c) 1984 by Sun Microsystems, Inc.
[ ... Several lines of configuration messages ... ]
#
```

### 4.3.10. Using Setup to Configure Your System

At this point you invoke the **setup** program to configure your system.

**Setup** is essentially a system configurator in two parts: it consists of an interactive front-end which gathers the information necessary to configure your system by conducting a dialogue with you, and a non-interactive back-end which uses this information to do the actual configuration. During the dialogue, **setup** does consistency and error checking to ensure that the configuration will work. If errors are detected, they are reported to you, and you are asked to enter corrected information.

You will use a different *setup* path if you are configuring

- A standalone system,
- A network server with equal-sized client partitions, or
- A network server with different-sized client partitions.

If you are configuring a standalone system, the dialogue with **setup** is fairly straightforward. Glance through the next section, *Path 1: Standalone System Configuration*, to familiarize yourself with the program's conventions.

If you are configuring a network server with one or more diskless nodes ("clients"), you can establish equal or unequal client partitions. The choice depends in part on your clients' storage needs — do they vary or are they roughly equivalent? — and also on your available resources — you may need to allocate unequal partitioning to accommodate all clients, for example. Please read the following before making the decision, as it is extremely difficult to 'undo' an established configuration.

This release of the system runs diskless workstations off one or more server disks by keeping a read-only copy of common files in the `/pub` directory (partition) of one of your drives. On this drive, the server machine uses partition 'a' as its root and partition 'b' as its paging area. The remainder of the drive (partition 'g') contains the `/pub` filesystem, followed by 'subpartitions' for the diskless clients. Each diskless client machine is allocated a file system area for private disk storage and a paging area (see *nd*(4) to understand how this is done). A generic abstract disk
would thus look something like this:

Table 4-5: Layout of a Generic Abstract Disk

<table>
<thead>
<tr>
<th>User</th>
<th>Minimum (MBytes)</th>
<th>Default (MBytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a Server's root</td>
<td>7.7</td>
<td>7.7</td>
</tr>
<tr>
<td>b Server's paging area</td>
<td>16.3</td>
<td>16.3</td>
</tr>
<tr>
<td>g /pub</td>
<td>17.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Client 1's file system area</td>
<td>x.0</td>
<td>x.0</td>
</tr>
<tr>
<td>Client 1's paging area</td>
<td>4.0</td>
<td>6.0</td>
</tr>
<tr>
<td>Client 2's file system area</td>
<td>x.0</td>
<td>x.0</td>
</tr>
<tr>
<td>Client 2's paging area</td>
<td>4.0</td>
<td>6.0</td>
</tr>
</tbody>
</table>

Partitions 'a' and 'b' were sized during the disk-labelling phase of diag, above; they are 7.7 and 16.3 MBytes respectively. The minimum configuration provided by setup then allocates 17 MBytes for /pub, 4 MBytes for each client's paging area, and then equally subdivides the remaining disk space for each client's file system. These minimum values are bare minimums; we suggest using them only if you have to (for example, to fit two clients on an 84-MByte disk). The default values noted above are much more livable partition sizes.

If you want to include the files containing the online manual pages, demos, or games, the public partition must be increased by the following amounts:

Table 4-6: Increase in /pub Size to Accommodate Online Manual Pages, Demos, or Games

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Pages</td>
<td>2.1 MBytes</td>
</tr>
<tr>
<td>Demos</td>
<td>3.75 MBytes</td>
</tr>
<tr>
<td>Games</td>
<td>1.8 MBytes</td>
</tr>
</tbody>
</table>

If you want to allocate equal amounts of disk space to each client, read through the subsection, Path 3: Standard Server Configuration. This path allows you to select the size of the public partition, and a size for a standard paging partition which will be given to each client. The remaining disk space will be divided equally among the clients as their file system area. Note that if you use this path, you must put /pub on the first disk you configure.

To assign different amounts of disk space to clients, read through the subsection, Path 3: Non-standard Server Configuration. This path allows you to select the size of the public partition, each client's paging area, and each client's file system area. This path also allows you to put /pub on a disk other than your first disk configured. Since the path for establishing a non-standard system configuration is slightly more complex than the other two paths, it requires some pre-planning. We have therefore included two copies of a worksheet which will help you consolidate the information you need before using setup, as well as one sample completed worksheet to use as a guide. PLEASE complete the worksheet before attempting configuration.
4.3.10.1. Path 1: Standalone System Configuration

This subsection provides an example of the setup interactive dialogue which is typical for a standalone workstation. In the example, what you might type in is shown in **boldface type** like this; whatever is simply displayed on the workstation monitor is shown in *Roman type* like this.

We invoke the *setup* program by typing the *setup* command. *Setup* begins by requesting global information:

```
# setup
Sun Microsystems Configuration System
Global Information
  1) network disk server
  2) standalone
  3) standalone with remote tape
Enter the number for your environment: 2
You have a standalone system; is this correct? (y/n): y
```

Note that the *setup* program asks you to verify configuration information after each 'phase' of configuration is complete. It is extremely difficult to undo system configuration, so be careful with your responses: 'y' casts things in concrete, and 'n' allows you to start the phase over again. You can also type 'q' to any prompt to quit the *setup* program and return to the shell — this allows you to annihilate what you have done and start over, if you have to.

Next, *setup* establishes your workstation's identity: name and — if you are tied into a network — address:
Host Information

Enter your hostname: lucifer

Is this workstation on a network? (y/n): y

Network numbers may be either class A, B or C
Their formats are:

Class A: nnn \[ 0 \leq nnn \leq 127 \]
Class B: nnn.b1 \[ 128 \leq nnn \leq 191 \]
Class C: nnn.b1.b2 \[ 192 \leq nnn \leq 255 \]

b1 and b2 are one byte (0-255) quantities

Enter your network number (default is 192.9.200): <RETURN>

Your network number is 192.9.200; is this correct? (y/n): y

Enter your host number

This is a number between 1 and 255: 14

Your hostname and host number are:

lucifer: 14

Is this correct? (y/n): y

Note that you need your network and host numbers here if your machine is on a network (see the earlier section, Determining Network Information, for an explanation of network and host numbers). Make sure you have these at hand before starting the dialogue.

The last phase of setup configuration requests information about your tape subsystem:

Tape Information

1) 1/4\" SCSI tape (st)
2) 1/2\" magnetic tape (mt)
3) 1/4\" archive tape (ar)

Enter the number for the type of tape: (1-3): 3

You have specified a 1/4\" archive tape; is this correct? (y/n): y

When this last phase has been completed, setup asks you whether you want to institute the configuration you have designed and, if you confirm, proceeds to edit several of the database files:

You have completed the configuration questions.
Continuing will destroy any existing files under /usr and any client partitions if you are a server.

Do you want to begin configuration? (y/n): y

Updating /etc/hosts ...
[... A few lines of configuration messages ...]

If your distribution is on two 1/4-inch tape cartridges, setup will prompt you to change to the second cartridge about two minutes into its back-end routine. Insert the second tape and type
'RETURN' to continue the routine; it takes approximately 25 minutes to complete. When \texttt{setup} completes its back-end work, your shell prompt returns. You can then continue your installation by booting the full UNIX system, as described near the end of this chapter.

4.3.10.2. Path 2: Standard Server Configuration

Note that if you follow this path, you are constrained to put /\texttt{pub} on the first disk you configure.

This subsection provides an example of the \texttt{setup} interactive dialogue which is typical for a standard network server configuration. This path allows you to select the size of your public partition, and establish the size of a standard paging partition which is allotted to each client. The remaining disk space is equally divided to make each client’s file system subpartition.

In the example, what you might type in is shown in boldface type like this; whatever is simply displayed on the monitor is shown in Roman type like this.

We invoke the \texttt{setup} program by typing the \texttt{setup} command. \texttt{Setup} begins by requesting global information:

```
# setup
Sun Microsystems Configuration System
Global Information
  1) network disk server
  2) standalone
  3) standalone with remote tape
Enter the number for your environment: 1
You have a network disk server system; is this correct? (y/n): y
```

Note that the \texttt{setup} program asks you to verify configuration information after each ‘phase’ of configuration is complete. It is extremely difficult to undo system configuration, so be careful with your responses: ‘y’ casts things in concrete, and ‘n’ allows you to start the phase over again. You can also type ‘q’ to any prompt to quit the \texttt{setup} program and return to the shell — this allows you to annihilate what you have done and start over, if you have to.

After determining your environment, \texttt{setup} gathers information about your disk configuration. It determines the type of disk controller being used for installation, and asks you for device information:

```
You have booted off of a Xylogics disk controller
Enter the number of disks attached to the Xylogics controller: (1-2): 2
The Xylogics disk controller has 2 disk(s):
  xy0
  xy1
Is this configuration correct? (y/n): y
```
For the remainder of the dialogue, setup will refer to your disk(s) by their UNIX device abbreviations. Next, setup begins the partitioning process for the first-named disk. If your configuration includes more than one disk on a single controller, as in our example, setup will completely finish partitioning the first disk and then turn to the second. If you have multiple controllers, you will be asked to repeat this entire phase for each controller. Your configuration will be validated after all disks have been partitioned. If errors are found at that time — for example, if all clients have not been allocated both file system and paging areas — you will be asked to edit your specified configuration until it is correct.

The partitioning phase of the dialogue with setup looks like this:

```
Disk Partition Information
Would you like to partition device xy0 into "n" equal sized clients? (y/n): y
Enter the pub partition size (nnnM or nnnK)
  - minimum is 17408K
  - default is 20480K: 17M
Partition size rounded to 17472K.
Enter the swap partition size (nnnM or nnnK)
  - minimum is 4096K
  - default is 6144K: 8M
Partition size rounded to 8288K.
Enter the number or clients on device xy0: (1-3): 2
Enter a client's name: adam
Enter a client's name: eve
```

You are asked to declare partition sizes in K Bytes or M Bytes, and setup takes care of rounding your partitions to cylinder boundaries for performance reasons. You can always type 'RETURN' as a response to a sizing prompt to get the default partitioning.

This partitioning phase of the dialogue sets up a /pub partition of whatever size you select (if this is the first device being partitioned), allocates a standard paging area for each of your specified clients (again, you determine the size), and carves the remaining free disk space into equally sized user subpartitions for each of them.

This completes partitioning for the first disk. If you have another controller, you are asked for its type, and the number of devices attached to it at this time, and are then asked to repeat the partitioning process for each disk. When all disks have been partitioned, setup asks for network information for each of your specified clients and your network server:
Network Information

Network numbers may be either class A, B or C
Their formats are:

- **Class A**: nnn (000 <= nnn <= 127)
- **Class B**: nnn.b1 (128 <= nnn <= 191)
- **Class C**: nnn.b1.b2 (192 <= nnn <= 255)

b1 and b2 are one byte (0-255) quantities

Enter your network number (default is 192.9.200): <RETURN>

Your network number is 192.9.200; is this correct? (y/n): y

Enter the 6-byte hexadecimal ethernet address for each of the clients
The correct form is xx:xx:xx:xx:xx:xx

Client adam: 8:0:20:0:14:76
Client eve: 8:0:20:0:8:14

The clients and their ethernet addresses are:

1) adam: 8:0:20:0:14:76
2) eve: 8:0:20:0:8:14

Are the ethernet addresses correct? (y/n): y

Do you want host numbers automatically assigned? (y/n): y

Server Information

Enter the name of the server: sun

The server's hostname is: sun

Is this correct? (y/n): y

Note that you use your network number, and your clients' hardware Ethernet addresses (established at the outset of the installation procedure) here. You may assign host numbers yourself, or have setup assign them automatically. For a discussion of network and host numbers, and clients' hexadecimal Ethernet addresses, see the earlier section, *Determining Network Information*.

The last phase of setup configuration requests information about your tape subsystem:

Tape Information

1) 1/4" SCSI tape (st)
2) 1/2" magnetic tape (mt)
3) 1/4" archive tape (ar)

Enter the number for the type of tape: (1-3): 3

You have specified a 1/4" archive tape; is this correct? (y/n): y

When this last phase has been completed, setup asks you whether you want to institute the configuration you have designed and, if you confirm, proceeds to edit several of the database files:
You have completed the configuration questions. Continuing will destroy any existing files under /usr and any client partitions if you are a server.

Do you want to begin configuration? (y/n): y

Updating /etc/hosts

[... A few lines of configuration messages ...]

If your distribution is on two 1/4-inch tape cartridges, setup will prompt you to change to the second cartridge about two minutes into its back-end routine. Insert the second tape and type 'RETURN' to continue the routine; it takes approximately 25 minutes to complete. When setup completes its back-end work, your shell prompt returns. You can then continue your installation by booting the full UNIX system, as described below.

4.3.10.3. Path 3: Non-Standard Server Configuration

Note that if you follow this path, you need not put /pub on the first disk you configure, as you must in Path 2.

4.3.10.3.1. Configuration Worksheet for Path 3

The purpose of this worksheet is to allow you to gather and stabilize your system configuration before you cast it in concrete with the setup program. Basically the worksheet helps you determine your real storage resources, the number of users you want to support, how you want to allocate your free disk space, and certain items of network configuration information necessary for the setup dialogue. We have provided two copies of the worksheet so that you can draft and redraft if necessary; it will be to your advantage to write everything out. We have also provided a sample completed worksheet to use as a guide.

There are several things to keep in mind during this process. One important point is that once the disk has been divided up in a certain way, it is extremely difficult to alter that structure. It can be changed only by going through the entire first time installation procedure again. Therefore, you might give some thought to what your future requirements are before you complete the worksheet. For example, you may want to "reserve" a client or two for future diskless stations that will be added in the coming months. The storage that you allocate for those future clients does not have to remain unused. You can make such partitions available as extra mounted filesystems for the original set of clients.

Now simply fill in the blanks.
Example Worksheet

The following is a sample worksheet for a configuration of 1 file server and 3 clients with 2 84-MByte (unformatted) disks. Each client has different disk space needs, so the disk space has been allocated accordingly.

Network Number: 192.9.200

<table>
<thead>
<tr>
<th>User</th>
<th>User Name</th>
<th>User Space</th>
<th>Paging Space</th>
<th>Hostnumber</th>
<th>Ethernet Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>dad</td>
<td>7.70MBytes</td>
<td>10.3MBytes</td>
<td>32</td>
<td>Not Needed</td>
</tr>
<tr>
<td>Client 1</td>
<td>robbie</td>
<td>30.0MBytes</td>
<td>15.0MBytes</td>
<td>33</td>
<td>8:0:20:31:41:59</td>
</tr>
<tr>
<td>Client 2</td>
<td>chip</td>
<td>20.0MBytes</td>
<td>10.0MBytes</td>
<td>34</td>
<td>8:0:20:27:18:28</td>
</tr>
<tr>
<td>Client 3</td>
<td>ernie</td>
<td>15.0MBytes</td>
<td>6.0MBytes</td>
<td>35</td>
<td>8:0:20:02:25</td>
</tr>
<tr>
<td>Client 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0  Disk #0

<table>
<thead>
<tr>
<th>7.7</th>
<th>Server's User</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>Server's Paging</td>
</tr>
<tr>
<td></td>
<td>/pub</td>
</tr>
<tr>
<td></td>
<td>(games, demos, man)</td>
</tr>
<tr>
<td>49</td>
<td>Ernie's User</td>
</tr>
<tr>
<td>65</td>
<td>Ernie's Paging</td>
</tr>
</tbody>
</table>

0  Disk #1

| 30   | Robbie's User  |
| 40   | Robbie's Paging |
| 45   | Chip's User    |
| 65   | Chip's Paging  |
Sample Worksheet #1

1. Number of disks and size (MBytes) of each disk:

<table>
<thead>
<tr>
<th>First Controller</th>
<th>Second Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Unit</td>
<td>Size</td>
</tr>
<tr>
<td>#0</td>
<td></td>
</tr>
<tr>
<td>#1</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows approximate free space for allocating client file system, client paging, and /pub subpartitions for Sun-supplied disk subsystems. The numbers are approximate because formatted capacity depends on the type of controller being used with the drive. Use the table to determine your real total storage resources.

<table>
<thead>
<tr>
<th>Available Disk Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
</tr>
<tr>
<td>Fujitsu SMD 8-inch</td>
</tr>
<tr>
<td>Fujitsu SMD 14-inch</td>
</tr>
<tr>
<td>Fujitsu Eagle</td>
</tr>
</tbody>
</table>

2. On the next page, allocate partitioning, complete network information, and block out 'real resource' disks.

When you allocate partitioning, remember:

- to include the /pub partition with the clients for disk units with /pub partitions.
- that setup's minimum values for client subpartitions are 4 MBytes file system area + 4 MBytes paging area = 8 MBytes per client. You must use these minimums to accommodate 2 clients on an 84- MByte (unformatted) disk. Default values — to give you more livable figures — are 6 MBytes file system area + 6 MBytes paging area = 12 MBytes per client.
- that the /pub partition must be at least 17 MBytes (for a /pub without manual pages, demos, or games); 20 MBytes is the default. If you wish to include the manual pages, demos, and games, allow 2.1 MBytes, 3.75 MBytes, and 1.8 MBytes respectively.
- that the server's root and server's paging partitions are allocated automatically by diag, and are 7.7 MBytes and 16.3 MBytes respectively for SMD disks.
- that setup rounds partitions to cylinder boundaries which vary depending on controller type and disk size, so these numbers are approximate — variance will be about ± 5%.

Revision H of 12 March 1984
Network Number: ____________

<table>
<thead>
<tr>
<th>User</th>
<th>User Name</th>
<th>User Space</th>
<th>Paging Space</th>
<th>Hostnumber</th>
<th>Ethernet Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td></td>
<td>7.7MBytes</td>
<td>16.3MBytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total</th>
<th>Partition</th>
<th>Total</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk #0</td>
<td>0</td>
<td>Disk #1</td>
</tr>
<tr>
<td></td>
<td>Server's User</td>
<td>7.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Server's Paging</td>
<td>16.3</td>
<td></td>
</tr>
</tbody>
</table>
Sample Worksheet #2

1. Number of disks and size (MBytes) of each disk:

<table>
<thead>
<tr>
<th>First Controller</th>
<th>Second Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Unit</td>
<td>Size</td>
</tr>
<tr>
<td>#0</td>
<td></td>
</tr>
<tr>
<td>#1</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows approximate free space for allocating client file system, client paging, and /pub subpartitions for Sun-supplied disk subsystems. The numbers are approximate because formatted capacity depends on the type of controller being used with the drive. Use the table to determine your real total storage resources.

<table>
<thead>
<tr>
<th>Available Disk Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk</td>
</tr>
<tr>
<td>Fujitsu SMD 8-inch</td>
</tr>
<tr>
<td>Fujitsu SMD 14-inch</td>
</tr>
<tr>
<td>Fujitsu Eagle</td>
</tr>
</tbody>
</table>

2. On the next page, allocate partitioning, complete network information, and block out 'real resource' disks.

When you allocate partitioning, remember:
- to include the /pub partition with the clients for disk units with /pub partitions.
- that setup's minimum values for client subpartitions are 4 MBytes file system area + 4 MBytes paging area = 8 MBytes per client. You must use these minimums to accommodate 2 clients on an 84-MByte (unformatted) disk. Default values — to give you more livable figures — are 6 MBytes file system area + 6 MBytes paging area = 12 MBytes per client.
- that the /pub partition must be at least 17 MBytes (for a /pub without manual pages, demos, or games); 20 MBytes is the default. If you wish to include the manual pages, demos, and games, allow 2.1 MBytes, 3.75 MBytes, and 1.8 MBytes respectively.
- that the server's root and server's paging partitions are allocated automatically by diag, and are 7.7 MBytes and 16.3 MBytes respectively.
- that setup rounds partitions to cylinder boundaries which vary depending on controller type and disk size, so these numbers are approximate — variance will be about ± 5%.
### Network Number:

<table>
<thead>
<tr>
<th>User</th>
<th>User Name</th>
<th>User Space</th>
<th>Paging Space</th>
<th>Hostnumber</th>
<th>Ethernet Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td></td>
<td>7.7MBytes</td>
<td>16.3MBytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Total Partition

<table>
<thead>
<tr>
<th>Total</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk #0</td>
</tr>
<tr>
<td>7.7</td>
<td>Server's User</td>
</tr>
<tr>
<td>24.0</td>
<td>Server's Paging</td>
</tr>
</tbody>
</table>

### Total Partition

<table>
<thead>
<tr>
<th>Total</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk #1</td>
</tr>
</tbody>
</table>
4.3.10.3.2. Setup Walkthrough for Path 3

This subsection provides an example of the setup interactive dialogue which is typical for the non-standard network server configuration. This path allows you to select the size and disk placement of the public partition, the size of each user's paging partition, and the size of each user's file system area. In the example, what you might type in is shown in boldface type like this; whatever is simply displayed on the monitor is shown in Roman type like this.

We invoke the setup program by typing the setup command. Setup begins by requesting global information:

```
# setup
Sun Microsystems Configuration System
Global Information
  1) network disk server
  2) standalone
  3) standalone with remote tape

Enter the number for your environment: 1

You have a network disk server system; is this correct? (y/n): y
```

Note that the program asks you to verify configuration information after each 'phase' of configuration is complete. It is extremely difficult to undo system configuration, so be careful with your responses: 'y' casts things in concrete, and 'n' allows you to start the phase over again. You can also type 'q' to any prompt to quit the setup program and return to the shell — this allows you to annihilate what you have done and start over, if you have to.

After determining your environment, setup gathers information about your disk configuration. It determines the type of disk controller being used for installation, and asks you for device information:

```
You have booted off of a Xylogics disk controller
Enter the number of disks attached to the Xylogics controller: (1-2): 2
The Xylogics disk controller has 2 disk(s):
  xy0
  xy1

Is this configuration correct? (y/n): y
```

For the remainder of the dialogue, setup will refer to your disk(s) by their UNIX device abbreviations. Next, setup begins the partitioning process for the first-named disk. If your configuration includes more than one disk on a single controller, as in our example, setup will completely finish partitioning the first disk and then turn to the second. If you have multiple controllers, you will be asked to repeat this entire phase for each controller. Your configuration will be validated after all disks have been partitioned. If errors are found at that time — for example, if all clients have not been allocated both /usr and paging areas — you will be asked to edit your specified configuration until it is correct.

The partitioning phase of the dialogue begins with an opportunity to allocate equal partitioning (Path 2); this is followed by optional allocation of the /pub subpartition:
Disk Partition Information

Would you like to partition device xyO into "n" equal sized clients? (y/n): n

Partitions for disk xyO - 40880K bytes free

Do you want the public partition on this disk unit? (y/n): y

Enter the pub partition size (nnnM or nnnK)
  - minimum is 17408K
  - default is 20480K: 17M
Partition size rounded to 17472K.

Throughout this phase, setup declares the amount of free space on your disk and lists any allocated partitions before asking you to alter the partitioning in any way. You are asked to declare partition sizes in K Bytes or M Bytes, and setup takes care of rounding your partitions to cylinder boundaries for performance reasons. You can always type 'RETURN' as a response to a sizing prompt to get the default partitioning.

The next step in partitioning is to allocate client /usr and paging areas (which need not be on the same disk, although each client must have both). If you want to 'reserve' space for future users, allocate "other" partitions of appropriate size:

Partitions for disk xyO - 23408K bytes free
1) public: 17472K

Do you want to add or edit a partition? (y/n/l): y

Enter the partition type (user/swap/other): user

Enter the user partition size (nnnM or nnnK)
  - minimum is 4096K
  - default is 6144K: 8M
Partition size rounded to 8288K.

Enter the name of the client: blaise

Continue this process until you have finished partitioning your first disk. Setup then allocates any remaining disk space to "other", and asks you to verify your work. If something has gone wrong, you can edit a partition by responding with its number when you are prompted for adding or editing. Both size and type fields can be changed. Our final screen might look something like this:

Partitions for disk xyO - 0K bytes free
1) public: 17472K
2) blaise's user: 6272K
3) blaise's swap: 4256K
4) albert's user: 6272K
5) albert's swap: 4256K
6) other: 2352K

Is this partitioning correct? (y/n): y

This completes partitioning for the first disk. If you have another controller, you are asked for its type, and the number of devices attached to it at this time, and are then asked to repeat the partitioning process for each disk. When all disks have been partitioned, and the configuration has been validated, setup asks for network information for each of your specified clients and
your network server:

Network Information

Network numbers may be either class A, B or C. Their formats are:

Class A: \text{nnn} \quad (000 \leq \text{nnn} \leq 127)
Class B: \text{nnn.b1} \quad (128 \leq \text{nnn} \leq 191)
Class C: \text{nnn.b1.b2} \quad (192 \leq \text{nnn} \leq 255)

\text{b1} and \text{b2} are one byte (0-255) quantities.

Enter your network number (default is 192.9.200): <RETURN>

Your network number is 192.9.200; is this correct? (y/n): y

Enter the 6-byte hexadecimal ethernet address for each of the clients.
The correct form is xx:xx:xx:xx:xx:xx

Client blaise: 8:0:20:0:14:76
Client albert: 8:0:20:0:8:55
Client isaac: 8:0:20:0:6:23
Client johannes: 8:0:20:0:11:22
Client immanuel: 8:0:20:0:13:49
Client tycho: 8:0:20:0:7:33

The clients and their addresses are:

1) blaise: 8:0:20:0:14:76
2) albert: 8:0:20:0:8:55
3) isaac: 8:0:20:0:6:23
4) johannes: 8:0:20:0:11:22
5) immanuel: 8:0:20:0:13:49
6) tycho: 8:0:20:0:7:33

Are the ethernet addresses correct? (y/n): y

Do you want host numbers automatically assigned? (y/n): y

Server Information

Enter the name of the server: archimedes

The server's hostname is:

archimedes

Is this correct? (y/n): y

Note that you use your network number, your clients' hardware Ethernet addresses (established at the outset of the installation procedure), and host numbers for clients and server here. You can choose your own host numbers, or have setup assign them automatically. For a discussion of network and host numbers, and Ethernet addresses, see the earlier section, Determining Network Information.
The last phase of `setup` configuration requests information about your tape subsystem:

**Tape Information**

1) 1/4" SCSI tape (st)
2) 1/2" magnetic tape (mt)
3) 1/4" archive tape (ar)

Enter the number for the type of tape: (1-3): 2

You have specified a 1/2" magnetic tape; is this correct? (y/n): y

When this last phase has been completed, `setup` asks you whether you want to institute the configuration you have designed and, if you confirm, proceeds to edit several of the database files:

You have completed the configuration questions.
Continuing will destroy any existing files under /usr and any client partitions if you are a server.

Do you want to begin configuration? (y/n): y

Updating `/etc/hosts`

[... A few lines of configuration messages ...]

If your distribution is on two 1/4-inch tape cartridges, `setup` will prompt you to change to the second cartridge about two minutes into its back-end routine. Insert the second tape and type 'RETURN' to continue the routine; it takes approximately 25 minutes to complete. When `setup` completes its back-end work, your shell prompt returns. You can then continue your installation by booting the full UNIX system, as described below.

### 4.3.11. Booting the Full UNIX System

Finally, you boot the full UNIX system. You must first halt the system, using the `/etc/halt` command. This shuts the system down in an orderly manner, and returns control to the monitor:

```
# /etc/halt
Syncing disks . . . . done
>
```

and now you can simply boot the UNIX system:

```
> b
Boot: disk(0,0,0)vmunix
Load: disk(0,0,0)boot
Boot: disk(0,0,0)vmunix
Size: 266240+ 32768+ 35316 bytes
Sun UNIX 4.2 (GENERIC) #145: Tue Feb 21 20:35:13 PDT 1984
Copyright (c) 1984 by Sun Microsystems, Inc.
[... Many lines of configuration messages ...]
gaia login: root
#
```

You can now use the UNIX system on this machine, and now boot any clients that will be served
by this machine. Continue now with kernel configuration and, if you have an Ethernet, with network configuration. Both of these steps are crucial for system performance reasons.

4.3.12. Kernel Configuration

Sun Microsystems' implementation of UNIX provides for a configurable kernel — that is, certain system parameters that were hardwired in previous implementations can now be changed. As a final step in system installation, all 1-MByte systems must — and all other systems should — configure the UNIX system kernel to suit your particular system. This reconfiguration reduces the kernel size, thus giving a larger effective memory size to programs. This is especially important if you intend to run the Sun Window System.

This section begins with a lockstep walkthrough of configuration procedures, which we hope provides enough information to take you safely through reconfiguration. If there is anything you don't understand or feel comfortable with, please read the sections just following the walkthrough; they give some explanation of what's going on (describe the layout of the kernel code, and the format of the configuration file used by the utility /etc/config to build your system configuration).

For a full discussion of configuring and building system images, see the document Building Sun Workstation Kernels in the Tutorials section of this manual.

This configurable system also provides for adding new device drivers to the system, since all the kernel object files required to build a new system are present. For procedures, see the Device Driver Tutorial in the Sun System Internals Manual.

4.3.12.1. Making a New Configuration

This subsection walks you through the procedure for making a new system configuration. If your system includes server(s) and client(s), you probably want to go through this procedure twice, making one kernel for the server machine (install it in /vmunix), and one kernel for all the clients (install it in /pub/vmunix).

1. Choose a name for your configuration of the system; here, SYS_NAME. Note that — by convention — the name should be in all uppercase letters.

2. Change directory to the /sys/conf directory, and create the configuration file and the directory in which the new system will be built. In this example, we assume that you will make a copy of the GENERIC file provided with this release, and edit the copy to create your new configuration file. Another (perhaps easier) method is to copy one of the stripped-down files in /sys/conf and edit it. XY100, for example, makes a good starting point for a Model 100U with one Xylogics disk; XYAR100 is a standard file for a Model 100U with a Xylogics disk and a 1/4" tape. XYMT150 is a Model 150U server with 2 Xylogics disks and a 1/2" tape.

   ```
   # cd /sys/conf
   # cp GENERIC SYS_NAME
   # chmod +w SYS_NAME
   # mkdir ../SYS_NAME
   ```

3. Edit SYS_NAME to reflect your system. This is the part of the procedure which takes some thought. On the next page, we provide a copy of the GENERIC file. You will notice that the file has two different 'types' of lines: lines which describe general aspects of the system
(which are, generally, parameters global to all kernel images specified in the configuration file), and lines which specify devices 'attached' to the system. We have indicated both classes of lines as follows:

- General system description lines are tagged "**mandatory**". These lines must be included in your system configuration file, either exactly as they stand or, if commentary indicates variables, with the variables adjusted to fit your system.
- Device description lines are given and explained. Often, we also refer you to the manual entry which covers the device in question. When you edit the GENERIC file, you should include only the lines which describe your system's devices. Remember that if you are creating a kernel for several clients, the configuration file must include the entire set of devices used by all the machines.
# GENERIC SUN

| machine | sun | # Mandatory
| cpu     | "SUN2" | # Mandatory
| ident   | GENERIC | If you use "GENERIC" as your system identifier, you may use the "swap generic" clause in the "config" line below. If you customize the identifier to SYS_NAME, you must either include an "options GENERIC" line or specify at least the device where your root file system lives in place of "swap generic". For example, the "config" line for a standard Sun 100U might read: "config vmunix root on xy". Finally, if SYS_NAME contains both alpha and numeric characters (as in XY100, for example), you must enclose the name in double quotes or you will get a syntax error when you run /etc/config ("XY100"). |
| timezone | 8 dst | # Mandatory
| maxusers | 4 | # Mandatory
| options | INET | # Mandatory INET means include Internet code
| options | SYSACCT | # Optional; include only with pseudo-device sysacct. Controls inclusion of code to do process accounting — see acct(2) and acct(5).

```plaintext
config
vmunix swap generic

pseudo-device
pty # Mandatory Specify root and swap devices

pseudo-device
bk # Mandatory

pseudo-device
sysacct # Mandatory

pseudo-device
inet # Mandatory

pseudo-device
ether # Mandatory

pseudo-device
loop # Mandatory

pseudo-device
nd # Mandatory

pseudo-device
win128 # Mandatory

pseudo-device
dtop4 # Mandatory

pseudo-device
ms3 # Mandatory

pseudo-device
kb3 # Mandatory

pseudo-device
ingres # Mandatory

controller
mb0 at nexus # Mandatory

controller
ip0 at mb0 csr 0xe4 priority 2 # Mandatory

controller
ip1 at mb0 csr 0x4 priority 2

disk
ip0 at ip0 drive 0 # Mandatory

disk
ip1 at ip0 drive 1 # Mandatory

disk
ip2 at ip1 drive 0 # Mandatory

disk
ip3 at ip1 drive 1

controller
xy0 at mb0 csr 0xe4 priority 2 # Mandatory

controller
xy1 at mb0 csr 0xe48 priority 2

disk
xy0 at xy0 drive 0 # Mandatory

disk
xy1 at xy0 drive 1

disk
xy2 at xy0 drive 0
```

Revision H of 12 March 1984
Installing UNIX for the First Time
Sun 100/150 Installation Manual

| disk     | xy3 at xyc1 drive 1          | #2nd disk on 2nd Xylogics controller |
| controller | sc0 at mb0 csr 0x80000 priority 2 | #1st SCSI controller |
| disk     | sd0 at sc0 drive 0 flags 0   | #1st disk on 1st SCSI controller |
| disk     | sd1 at sc0 drive 1 flags 0   | #2nd disk on 1st SCSI controller |
| tape     | st0 at sc0 drive 32 flags 1  | #SCSI tape |
| device   | ropc0 at mb0 csr 0xce0000 priority 1 | #Mandatory Raster Op chip – see ropc(4) |
| device   | skyo at mb0 csr 0x2000 priority 2 | #Sky Floating Point board |
| device   | ss0 at mb0 csr 0xce0000 flags 3 priority 2 | #CPU ports – see ss(4) |
| device   | ss1 at mb0 csr 0xce0000 flags 3 priority 2 | #Sun-2 Video Board ports; required if using Sun-2 keyboard and mouse |
| device   | ss2 at mb0 csr 0x80000 flags 3 priority 2 | #1st SCSI Board ports |
| device   | ss3 at mb0 csr 0x81000 flags 3 priority 2 | #1st SCSI Board ports |
| device   | oct0 at mb0 csr 0x520 flags 0xff priority 4 | #Central Data Octal Card – see oct(4) |
| device   | mt0 at mb0 csr 0x620 flags 0xff priority 4 | #Systech terminal MUX — see mti(4) |
| device   | ie0 at mb0 csr 0x80000 priority 3 | #1st Sun-2 Ethernet Controller |
| device   | ie0 at mb0 csr 0x80000 flags 2 priority 3 | #2nd Sun-2 Ethernet Controller |
| device   | ec0 at mb0 csr 0x00000 flags 3 priority 3 | #1st 3COM Ethernet Controller – see ec(4) |
| device   | ec1 at mb0 csr 0x00000 flags 3 priority 3 | #2nd 3COM Ethernet Controller |
| controller | tm0 at mb0 csr 0xa00 priority 3 | #1st TAPEMASTER tape controller – see tm(4) |
| controller | tm1 at mb0 csr 0xa02 priority 3 | #2nd TAPEMASTER tape controller |
| tape     | mt0 at tm0 drive 0 flags 1   | #1st 1/2" tape drive on 1st controller |
| tape     | mt1 at tm1 drive 0 flags 1   | #1st 1/2" tape drive on 2nd controller |
| device   | ar0 at mb0 csr 0x200 priority 3 | #1st 1/4" tape drive – see ar(4) |
| device   | ar1 at mb0 csr 0x200 priority 3 | #2nd 1/4" tape drive |
| device   | cgone0 at mb0 csr 0xe0000 priority 3 | #1st Sun Color Board – see cg(4) |
| device   | btwc0 at mb0 csr 0x70000 priority 3 | #1st monochrome Sun-2 monitor |
| device   | bwise0 at mb0 csr 0x00000 priority 3 | #1st monochrome Sun-1 monitor |
| device   | vp0 at mb0 csr 0x400 priority 2 | #Ikon Versatec Board – see vp(4) |
| device   | vpc0 at mb0 csr 0x480 priority 2 | #1st Systech Centronics/Versatec Board – see vpc(4) |
| device   | vpc0 at mb0 csr 0x500 priority 2 | #2nd Systech Centronics/Versatec Board |
| device   | pi0 at mb0 csr 0x3e20000 priority 1 | #Parallel input. Only used on Sun Models 100U and 150U, for keyboard and mouse. |
|          |                              | Not needed if using terminal for console |

Revision H of 12 March 1984
The following configuration XY100 is a stripped down version of the GENERIC system for a system with only a single local peripheral: a Xylogics controller supporting one disk drive. It's a good example of a standard configuration file:

```
# # Model 100 with one Xylogics Disk #

machine    sun
cpu        "SUN2"
ident      "XY100"
timezone   8 dst
maxusers   2
options    INET

cfg        vmunix root on xy

pseudo-device    pty
pseudo-device    inet
pseudo-device    ether
pseudo-device    loop
pseudo-device    win32
pseudo-device    dtop1
pseudo-device    ms1
pseudo-device    kb1
controller     mb0 at nexus ?
controller     xyc0 at mb0 csr 0xee40 priority 2
disk          xy0 at xyc0 drive 0
device        ropc0 at mb0 csr 0xee0800 priority 1
device        zs0 at mb0 csr 0xeec800 flags 3 priority 2 # cpu ports
device        ec0 at mb0 csr 0xe0000 priority 3
device        bwone0 at mb0 csr 0xc0000 priority 3
device        pl0 at mb0 csr 0xee2000 priority 1
```

For more examples of standard configuration files, see Appendix C in the paper Building Sun Workstation Kernels in the Tutorials section of this manual.

4. When you have finished editing SYS_NAME, run config:

```
# /etc/config SYS_NAME
```

5. Now change directory to the new configuration directory, ..SYS_NAME, and make the new system:

---

2 Note that if the "ident" field (system name) in your configuration file contains any digits, /etc/config prints: "config: Line 5: syntax error" at this point. You may ignore the error message.
The make depend command creates the dependency tree for all system source files.

6. Now you can install the new system and try it out.

```
# cd ..SYS_NAME
# make depend
[ lots of output ... ignore the error message
# make
[ lots of output ]
```

The `make depend` command creates the dependency tree for all system source files.

7. If the system appears to work, all is well. If the new kernel doesn't seem to be functioning properly, boot `/vmunix.org`, move it back to `/vmunix`, and go about fixing your new kernel:

```
# /etc/halt
> b vmunix -s
The system boots up in single user state, and
then you can try things out
#
```

You can now continue with the next phase of installation — loading the manuals, demos, and games directories — or, if you do not wish to include this material in your system, with the Network Configuration section in the System Set-up and Operation chapter. The subsections immediately below give background information for configuring the kernel.

### 4.3.12.2. Configuration Directory Layout

The system as shipped contains the essential source and object files in the `/sys` directory. The `/sys` directory contains several subdirectories that in turn contain the object code modules required to build different parts of the system. The subdirectories in `/sys` are:

- **OBJ**
  - Contains the kernel object code plus the header files describing the number of devices of each type.

- **conf**
  - Contains the files describing the configuration of the system and what files are required to build the new system. Also contains a `README` file describing how to make a new kernel. Finally, contains the `GENERIC` system configuration file.

---

\* The `make depend` may generate a spurious error message like `.../sun/locore.s: no such file or directory` at this point, which you may safely ignore.
supplied with the distribution. This file may be copied and edited to produce your specific system configuration file; procedures are given above.

- h: Header files containing definitions and data structures.
- machine: Machine-dependent routines for the Sun system — this is a symbolic link to the sun directory.
- net: Networking files.
- netimp: Arpanet IMP support code.
- netinet: Internet protocols.
- netpup: 3 Mbit Xerox PUP protocol support.
- sun: Sun-specific mainline code.
- sundev: Sun-specific device drivers.
- sunif: Sun network interface code.
- sys: General system routines.

### 4.3.12.3. Building a Configuration

Building a new system is a semi-automatic process; most of the fine detail is handled by a configuration build utility called `/etc/config`. `/etc/config` uses three files as input:

- **Specific Configuration File**
  
  This file contains a description of the characteristics of a specific system, plus descriptions of the devices that that system can support. Every new version of the system has another file of this type.

- files: Contains a list of the files required to build the basic kernel.
- files.sun: Contains a description of the files required to build the specific Sun system. You add the names of new driver modules in here when adding drivers to the kernel.

/`etc/config` should be run from the `conf` subdirectory of the system source or object files. `/etc/config` assumes that there is already a directory `../System_Name` created, and it places all its output files in there. The output of `/etc/config` consists of a number of files:

- `ioconf.c`: Contains a description of I/O devices attached to the system.
- `makefile`: For building the system.

**Header Files**

A set of header files which contain the number of various devices that will be compiled into the system.

After running `/etc/config`, you must then change directory to the directory in which the new `makefile` was created, and use `make` to create the dependency tree for the new system. `/etc/config` reminds you of this when it completes:

```bash
# cd ../System_Name
# make depend
#
```

If you get any other error messages from `/etc/config`, fix the problems in your configuration file and try again. If you try to compile a system that had configuration errors, you will meet with failure.
### 4.3.12.4. Format of the Configuration File

There are two main classes of information in the configuration file:
- Lines which describe general things about your system.
- Lines which describe the devices on the system, and what those devices are connected to.

The two subsections below contain descriptions of the two classes of information.

In these descriptions, a number can be a decimal integer, a whole octal number or a whole hexadecimal number. Hexadecimal and octal are specified to `etc/config` in the same way they are specified to the C compiler, namely, a number starting with '0x' is a hexadecimal number and a number starting with just a '0' is an octal number. When specifying the timezone, you may also use a floating point number.

Comments are specified in a configuration file with the character '#' All characters from a '#' to the end of a line are ignored.

Lines beginning with tabs are considered continuations of the previous line.

#### 4.3.12.4.1. General System Description Lines

The first seven general description lines in the configuration file are mandatory. They are:

- **machine type**
  - This system is to run on the machine type specified. Only one machine type can appear in the configuration file. The legal type for a Sun system is `sun`.

- **cpu type**
  - This system is to run on the cpu type specified. For a Sun system, legal type is "SUN2" (enclose in double quotes).

- **ident name**
  - Gives the system identifier — a name for the machine or machines that run this kernel. name must be enclosed in double quotes if it contains both letters and numbers. Also, note that if name is GENERIC, you can specify the unique config clause swap generic in the config line described below. If you use any other string for name, and you also include an options GENERIC line, you can still use the swap generic line. However, if you use any other string for name and omit the options GENERIC line, you must specify your root and, optionally, swap devices in the config clauses field.

- **timezone number [ dst ]**
  - Specifies the timezone you are in. This is measured in the number of hours west of GMT you are. 5 is EST, 8 is PST. Negative numbers indicate hours east of GMT. If you specify dst, the system will convert to and from daylight savings time when appropriate.

- **maxusers number**
  - The maximum expected number of simultaneously active users on this system is number. This number is used to size several system data structures. In particular, it controls the size of the process table which sets the upper bound on the 'legal' number of user processes. Since this table is statically allocated, it cuts into your buffer space; it is thus important to set number no higher than necessary for your system. If you are configuring a kernel for a single-user Sun Workstation, for a single-user networking node, or — especially — a 1 MByte system, set number to "2". Use "4" only if you anticipate running an unusually high number of user processes per machine — for example, if you plan to run 7+ windows at a time.
options optlist

Compile the listed options into the system. Options in this list are separated by commas. There is a list of options that you may specify in the generic makefile. A line of the form 'options FUNNY,HAHA' yields -DFUNNY -DHAHA to the C compiler. An option may be given a value, by following its name with '=' then the value enclosed in (double) quotes. None of the standard options use such a value.

config kernelname config clauses

Specifies a bootable system image; multiple bootable images may be specified in a single configuration file. Here kernelname is the name of the loaded kernel image (normally, vmunix). The config clauses specify at minimum the root device, and may also specify the swap device. If you are using GENERIC as your system identifier (described above), or if you customized your identifier and also included an options GENERIC line, then your config clause may legally be swap generic. In any other case, you must specify at least your root device. For example, if you are configuring a kernel for a Sun 100U, and your ident line looks like this:

ident "MY100"

your config line might look like this:

cfg config vmunix root on xy

4.3.12.4.2. Device Description Lines

The second class of line in the configuration file describes what devices your system has and what they are connected to (for instance, I have a Xylogics xy450 on the Multibus). The device description lines for all supported devices are given in the section 4 pages of the Sun System Interface Manual; see the 'synopsis' section of the entry for each device.

Each device description line has the following format:

dev_type dev_name at con_dev more_info

The meaning of the fields is:

Dev_type

is one of controller, tape, disk, device, or pseudo-device. These types have the following meanings:

controller

is a Multibus disk controller or a Multibus tape controller.

disk or tape

are devices that are connected to a controller.

device

is something which plugs into the Multibus, like a cartridge tape interface.

pseudo-device

is something which should be conditionally loaded, but is not really a device. Current examples are the pseudo-tty driver and various network subsystems. For pseudo-devices, more_info may be specified as an integer, that gives the value of the symbol defined in the header file created for that device, and is generally used to indicate the number of instances of the pseudo-device to create. If you load a subsystem you may find it necessary to enable conditional code using an options specification.
Installing UNIX for the First Time

4.3.13. Loading the Manuals, Demos, and Games Directories

The seventh file on the distribution tape(s) contains tar images of the /usr/man directory (online manuals), /usr/demo directory (demonstration programs), and /usr/games directory (games). You can load all or part of this file if you want to. For some configurations, disk storage space will be a consideration: the online manuals require approximately 2.1 MBytes, /usr/demo requires 3.75 MBytes, and /usr/games takes 1.8 MBytes.

There is one basic procedure for loading to a standalone system from a quarter-inch tape, and one for the nine-track tape. These follow. If your system is a file server rather than a standalone system, simply change directory to /pub/usr rather than /usr initially, then complete the identical sequence of commands.

To load from quarter-inch tape:

1) Become super-user.
2) Insert the first quarter-inch tape cartridge.
3) Type the following sequence of commands (choosing the directories you want, at the last):

```
# cd /usr  (or /pub/usr if you are loading on a fileserver)
# mt -f /dev/nrar0 rew
# mt -f /dev/nrar0 fsf 0
# tar xvf /dev/rar0 man demo games
```
To load from half-inch tape:

1) Become super-user.
2) Mount the half-inch tape.
3) Type the following sequence of commands (choosing the directories you want, at the last):

   # cd /usr (or /pub/usr if you are loading on a fileserver)
   # mt rew
   # mt fsf 8
   # tar xv man demo games

4.3.14. The Next Step

Unless you have to install UNIX on systems without tape drives (procedure given in the following chapter), you have now completed the portion of installation in which you need the distribution tape. However, there are a few other installation matters — like network configuration and setting up the mail system — which you may not want to ignore. We continue with these topics in the chapter System Set-up and Operation. Take a glance at it now.
Chapter 5

Installing UNIX on Systems without Tape Support

This chapter describes the process for installing the UNIX system onto a workstation which does not have a tape drive installed. The process involves installing the UNIX system across the Ethernet from a system which does have a tape drive. Throughout this chapter we refer to the "remote host" and "target" machines. The remote host is the machine with the tape drive; the target is the machine without it.

CAUTION: if you are upgrading your Sun Model 100 or 150 rather than starting with a system 'fresh from the box,' please heed the warning on the front page of this manual. SUN-1 systems which are upgrading must not attempt to install this release of the system until all Multibus memory has been removed or disabled, and — for an upgrade from a Xylogics 440 to a Xylogics 450 — cabling has been re-routed. Disk drive switches may also need resetting. Make sure you have followed the hardware upgrade procedures given in Sun's Upgrade Installation Guide before proceeding.

CAUTION: If you are upgrading your system software, rather than beginning for the first time, read the Upgrading System Software chapter BEFORE proceeding here. The chapter gives procedures for saving and rebuilding your existing root and /usr file systems.

5.1. Overview of the Installation Procedure

Here are the steps required to install the system over the network:

1. Complete installation of the UNIX system on a Sun System equipped with a tape drive — your "remote host" or "network disk server" — as described in the chapter, Installing UNIX for the First Time.

2. Determine network information necessary for installation: the target's hardware Ethernet address and Internet address, and the remote host's host number (in hexadecimal).

3. If your remote host is configured as a standalone system rather than as a network disk server, you must turn its /usr file system into a public network disk. If your remote host is already configured as a server, this step is unnecessary.

4. Load the mini file system onto the public disk from the boot tape.

5. Boot diag over the network; run diag to format (if necessary) and label your disk.
6. Boot the standalone copy program over the network. Run copy to copy a mini-file system over the network into the swap area on your disk.
7. Boot the mini-UNIX system.
8. Edit the /etc/hosts and ~/.hosts files.
9. Mount the release tape on the remote host.
10. Extract the root file system from the tape.
11. Boot UNIX in single user state.
12. Run the setup program to extract the /usr file system from the tape and initialize the network files.
13. Boot the full UNIX system
14. Load the manuals, demos, and games directories if you wish.

Now read the sections which follow. There should be enough detail there to walk through the remote installation procedure. In all the examples, what you type is shown in boldfaced text like this. Boldfaced text is typed exactly as it appears. Some places we have shown parts of command in italic text like this, and this means that there is something that must be substituted at that point. Italic characters are variables.

Two very important variables which you will see in the examples — and have already seen if you have installed UNIX on your remote host — are tape and disk. Where tape appears, replace it with the correct device abbreviation for your tape drive:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>mt</td>
<td>Nine-track magnetic tape</td>
</tr>
<tr>
<td>ar</td>
<td>Archive quarter-inch tape</td>
</tr>
</tbody>
</table>

Where disk appears, replace it with the correct device abbreviation for your disk:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>xy</td>
<td>Xylogics 450 disk controller</td>
</tr>
<tr>
<td>ip</td>
<td>Interphase 2180 disk controller</td>
</tr>
</tbody>
</table>

You can correct typing errors at any time. The DEL key backspaces over and erases characters, and the control-U key (hold down the CONTROL key while typing the letter U) erases the entire line typed so far.

If you need to get back to the monitor for any reason, you can abort by typing 'SET-UP A' (hold down the SET-UP key while typing 'A') or 'ERASE-EOF A' on the Sun Workstation keyboard at any time. If you are using a standard terminal as a console, the BREAK key generates an abort.
5.2. Determining Network Information

Before beginning the installation procedure, you need to know the hardware Ethernet address of the target machine, the target's full Internet address (network number and host number), and the host number (in hexadecimal) of the remote host. You will use these addresses later in the installation procedure.

To obtain the hardware Ethernet address of the target, power up the target workstation. You should see the PROM monitor's sign on messages, one of which is the Ethernet address. Stop the auto boot immediately by aborting — type 'SET-UP A' (hold down the SET-UP key while typing 'A'), 'ERASE-EOF A', or BREAK — and then you should see the monitor's prompt, which is a > sign:

Self Test completed successfully.

Sun Workstation, Model Sun-1/100U or Sun-1/150U, type of keyboard
ROM Rev N, some_number_MBytes memory installed
Serial #some_number, Ethernet address n:n:n:n:n:n

Auto-boot in progress ...

abort by typing 'SET-UP A', 'ERASE-EOF A', or BREAK here

Abort at some address

>

Copy down the full hardware Ethernet address. Addresses are given as a six-byte hexadecimal value. Each byte of the address is separated by a colon. A typical address might be '8:0:20:0:14:76'.

Internet addresses consist of two parts: a network number, and a host number. For remote installation, you need the target's full Internet address, and the remote host's host number (in hexadecimal).

Network Number

The network number is an arbitrary value used to uniquely identify multiple networks. If you have been assigned a network number by ARPA, use that number; if not, you may use Sun's default network number: 192.9.200. By this point in the installation procedure, you probably know your network number. If in doubt, check the /etc/hosts file on the remote host machine. Entries consist of each machine's full Internet address and name, for example:

192.9.200.48  augustus
192.9.200.50  julius
192.9.200.52  claudius

Here, julius' Internet address is 192.9.200.50; his network number is 192.9.200, and his host number (in decimal) is 50.

Host Number

The host number is also arbitrary — except that it must be between 1 and 255 — and is used to uniquely identify each node on your network. The host number is the final component of the full Internet address; it follows the network number. Since you must assign your target a unique number, check the /etc/hosts file on the remote host to make sure you're not using an address already in use.

Finally, find the entry for the remote host in /etc/hosts, and note the machine's host number. Since the host number entries in /etc/hosts are in decimal, and you need the remote host's host
number in hexadecimal, you will probably need to convert. You can use adb for this if you wish:

    host% adb
    0host_number_in_decimal = X
    host_number_in_hex
    'D
    host%

5.3. Setting up the Remote Host

DO THIS IF THE REMOTE HOST IS NOT A NETWORK DISK SERVER.

If your remote host is already configured as a server, skip to the next step.

If the remote host machine is not a network disk server, turn the /usr file system into a public network disk by adding two lines to nd.local to reference /dev/ip0g (for an Interphase disk controller) or /dev/zy0g (for a Xylogics disk controller). The lines should look like this:

    user 0 0 /dev/disk0g -1 -1 -1
    son

Then enable the network disk server:

    # cd /dev
    # MAKEDEV nd
    # /etc/nd < /etc/nd.local

Next, make the /usr disk into a public disk so that it can be accessed across the network by doing the following sequence of commands:

    # mkdir /usr/stand
    # cp /stand/* /usr/stand
    # ln -s /usr /pub
    # cp /boot /pub/boot
    # cd /usr/mdec
    # installboot bootnd /dev/disk0g
    # sync
    #

Then proceed with the next step.

5.4. Loading the Mini File System to the Server

Now load the mini file system onto the public disk from the boot tape with the following sequence of commands. Remember to replace tape with mt for the nine-track tape, or ar for the Archive quarter-inch tape. Also, if you are using a nine-track tape, use 20 for blk_factor ('bs=20b'); use 126 for a 1/4-inch tape ('bs=126b');
Sun 100/150 Installation Manual Installing UNIX on Systems without Tape Support

This takes about three minutes using a 1/2-inch tape, and less using a 1/4-inch cartridge.

5.5. Running Diag

Now, you start to work on your target machine, and install UNIX from the remote host. The first step is to format the disk with the diag utility. You boot diag from the network server with a boot command:

```
> boot(0,host_number)stand/diag
```

Remember to replace host_number with the remote host's hexadecimal host number (obtained earlier). If you have more than one Ethernet Controller Board in your system, and you are booting from the second, third, etc., replace the "0" in the above command with the controller's address on the Multibus (in hex).

The monitor should boot diag from the network disk server. When diag starts up, it displays a sign on message:

```
Disk Initialization and Diagnosis
When asked if you are sure, respond with 'y' or 'Y'
```

Diag first asks you a series of questions about the disk you will be using. The answers to these questions 'configure' diag to work with that specific hardware. The first thing diag wants to know is the type of disk controller to use. Diag displays a menu of the different disk controller types.

Diag's second question concerns the address of the controller on the Multibus. The table below shows the default addresses for disk controllers.

<table>
<thead>
<tr>
<th>Controller Type</th>
<th>Address (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interphase SMD-2180</td>
<td>40</td>
</tr>
<tr>
<td>Xylogics 450/440</td>
<td>ee40</td>
</tr>
</tbody>
</table>

Next, diag wants to know the unit number of the disk — 0 is usually the correct answer if you only have one disk, and the type of disk drive you are working with — diag displays a menu of

---

3 Note that if you have one Xylogics SMD Controller and one Interphase Controller in your system, you cannot use the default addresses (as the Interphase Controller sees only four bytes). The Interphase must be configured to be at address 48, and the Xylogics at ee40. See the Hardware Configuration and Expansion chapter for board configuration procedures.
the different disks. When you have specified which disk drive you are using, `diag` then displays a table of the physical data about that disk, which includes the number of cylinders, number of alternate cylinders, number of heads, and number of sectors per track.

At this point, `diag` knows all it needs to know about the disk, and it displays its prompt:

```
diag>
```

One of the responses to this prompt can be the `h` (for help) command. If you use the `h` command, `diag` displays a list of its commands.

### 5.5.1. Formatting the Disk

If you are starting with a completely new disk, you will need to format the disk first. If you are not starting with a completely new disk, you can skip to the next topic, *Labelling the Disk.*

Before formatting, clear any outstanding errors by typing the `clear` command:

```
diag> clear
    clear
    diag>
```

Next, type the `format` command. `Diag` then warns you that formatting a disk destroys all information which might already be stored on that disk, and asks for confirmation before going ahead and doing the job:

```
diag> format
    format/verify, DESTROYS ALL DISK DATA!
    are you sure? y
    # of surface analysis passes? 5
```

Use five surface analysis passes for a completely new disk; use one pass for a disk which has been formatted in the past and is known not to have any bad spots. Formatting a disk takes anywhere from a half-hour to four hours depending on how many passes you select, the size of the disk, and the type of controller you happen to have. For example, the format phase takes about thirty minutes for one surface analysis pass on an 84-MByte disk, using an Interphase SMD-2180 disk controller, and a slightly shorter time using a Xylogics 450 or 440 controller. `Diag` displays the current cylinder number as it formats each cylinder.

### 5.5.2. Labelling the Disk

A disk must be labelled after it has been formatted. The purpose of a label is to record (in a well known place on the disk) information about how the disk is divided up into partitions for different purposes such as swap space and file systems. `Diag` labels a disk in response to the `label` command. Here is an example of using the `label` command to write a label on the disk which was formatted in the discussion above. When you give the `label` command to `diag`, it asks if you want to use the logical partition map that is “built in” to the program:

```
diag> label
    label this disk...
    OK to use logical partition map 'Fujitsu-M2312K (Sun D84)' ? y
    Are you sure you want to write? y
```
Then, when diag has labelled the disk, it verifies the label it has just written. There is in fact a separate verify command, but diag automatically does a verify as part of the labelling process. When the ‘diag>’ prompt returns, get back to the monitor by responding “q” (for ‘quit’).

5.6. Loading the Mini UNIX System

Now boot the standalone copy program from the disk server. Remember to replace disk with xy for the Xylogics controller, or ip for the Interphase disk controller; and replace host_number with the remote host’s host number (in hex). Also, if you are not booting from the first Ethernet Controller Board in your system, use the board’s Multibus address (in hexadecimal) rather than “0” in the boot command:

```bash
> b ec(0,host_number)stand/copy
Boot: ec(0,host_number)stand/copy
Load: ec(0,host_number,0)boot
Boot: ec(0,host_number,0)stand/copy
[... messages displaying sizes of copy program ...]
Standalone Copy
From: ec(0,host_number,0)minifs
To: disk(0,0,1)
```

The load process takes about four minutes. This process loads the mini file system into the swap area on the disk. When it completes, the copy program returns control to the monitor:

```
Copy completed
>
```

5.7. Booting the Mini UNIX System

The next step is to boot UNIX in single user state and ask for its root file system:

```
> b ec(0,host_number)boot
```

When UNIX comes up and asks for its root file system, tell it “disk0*”. Since this notation looks a bit ambiguous, let me clarify: if you are using a Xylogics controller, your root device is xy0*; if you are using an Interphase controller, it is ip0* — the asterisk is part of the device name:

```
Boot: disk(0,0,1)vmunix -as
[...lots of messages ...]
root device: disk0*
#
```

At this point, the system may gently remind you to reset the date and time:

```
WARNING: clock gained 16845 days -- CHECK AND RESET THE DATE!
```

You can do this when your prompt returns, with the `date(1)` command.
5.8. Setting the Date

Set the correct system date and time by using date(1):

```
# date yymmddhhmm[.ss]
```

`yy` here is the last two digits of the year; `mm` specifies month; `dd` designates day of the month; `hh` is hour (on a 24-hour clock); the next `mm` is minutes elapsed; and the optional `.ss` specifies seconds. The system echoes the date set back to you.

5.9. Editing the /etc/hosts and /.rhosts Files

Now you must edit the `/etc/hosts` file on both the remote host and target machines, to make them aware of each other's existence. Also, edit `/.rhosts` on the remote host only. Proceed as follows:

1. On the remote host machine, edit the `/etc/hosts` file, and add an entry for the target machine. There must be an entry for the remote host as well; if it's not there, add it. Entries in `/etc/hosts` consist of each machine's full Internet address (network number and host number), and the machine's name. For example, assuming your remote host is called 'krypton,' and your target is 'gaia,' the entries might look like this:

```
192.9.200.1    krypton
192.9.200.23   gaia
```

Here, 192.9.200 is being used for the network number; the remote host's host number is 1, and the target's host number is 23.

By now, you probably know your network number, and have a host number ready to assign your target — if you're confused, see the earlier section, Determining Network Information.

2. Still on the remote host machine, edit the `/.rhosts` file, adding an entry (hostname only) for the target machine. This will allow you to perform remote processes 'on' the remote host 'from' the target machine at the super-user level (for example, the remote extract in the next phase of installation). If the file does not exist, create it.

3. Now move to the target machine, and edit `/etc/hosts` (which should be nearly empty). Add the same two lines discussed in step 1: one entry for the remote host and one for the target machine.

4. Run `ifconfig` on the target machine:

```
# /etc/ifconfig ec0 your_target_name
```

5.10. Loading the Root File System

Now you must mount the distribution tape on the remote host's tape drive. Then run the `rxtr` (remote extract) shell script on the target machine to copy the root file system across the Ethernet. Again, use `xy` for the Xylogics disk controller, or `ip` for the Interphase disk controller to replace `disk`, and `mt` for the nine-track tape, or `ar` for the Archive quarter-inch tape to replace `tape`:

5-8 Revision H of 12 March 1984
The extraction process takes ten to twenty minutes. The next job is to configure your system and load the /usr file system.

5.11. Booting the UNIX System in Single-User State

Now type a couple of sync commands to flush all I/O activity to the disks, then get back to the monitor by typing 'SET-UP-A' or 'ERASE-EOF-A'. The monitor responds by displaying a message like:

Abort at some address

When you see the monitor > sign, boot the UNIX system in single-user state:

> b vmunix -s

[...incredible amounts of messages ...]
#

5.12. Using Setup to Configure Your System

At this point you invoke the setup program to configure your system.

Setup is essentially a system configurator in two parts: it consists of an interactive front-end which gathers the information necessary to configure your system by conducting a dialogue with you, and a non-interactive back-end which uses this information to do the actual configuration. During the dialogue, setup does consistency and error checking to ensure that the configuration will work. If errors are detected, they are reported to you, and you are asked to enter corrected information.

For standalone systems booting from a remote tape drive, the setup dialogue runs as follows. In the example, what you might type in is shown in boldface type like this; whatever is simply displayed on the workstation monitor is shown in Roman type like this. *Italic items* are variables which you must substitute.
We invoke the `setup` program by typing the `setup` command. `Setup` begins by requesting global information:

```
# setup
Sun Microsystems Configuration System
Global Information
  1) network disk server
  2) standalone
  3) standalone with remote tape
Enter the number for your environment: 3
You have a standalone system with remote tape; is this correct? (y/n): y
```

Note that the program asks you to verify configuration information after each 'phase' of configuration is complete. It is extremely difficult to undo system configuration, so be careful with your responses: 'y' casts things in concrete, and 'n' allows you to start the phase over again. You can also type 'q' to any prompt to quit the `setup` program and return to the shell — this allows you to annihilate what you have done and start over, if you have to.

Next, `setup` establishes your workstation's identity: name and address:

**Host Information**

Enter your hostname: `your_hostname`

Network numbers may be either class A, B or C

Their formats are:

- Class A: `nnn` (0 <= nnn <= 127)
- Class B: `nnn.bl` (128 <= nnn <= 191)
- Class C: `nnn.bl.b2` (192 <= nnn <= 255)

`b1` and `b2` are one byte (0-255) quantities

Enter your network number (default is 192.9.200): `<RETURN>`

Your network number is 192.9.200; is this correct? (y/n): y

Enter your host number

This is a number between 1 and 255: `your_host_number`

Your hostname and host number are:

`your_hostname: your_host_number`

Is this correct? (y/n): y

Note that you use your network and host numbers here. See the section above, *Determining Network Information*, if you need explanation of these.

The last phase of `setup` configuration requests information about your remote host's tape subsystem:
Tape Information

1) 1/4" SCSI tape (st)
2) 1/2" Magnetic tape (mt)
3) 1/4" Archive tape (ar)

Enter the number for the type of tape: (1-3): 3

You have specified a 1/4" Archive tape; is this correct? (y/n): y

Enter the name of the remote host that the 1/4" Archive tape is attached to: server_name

Enter the host number for server_name: host_number

The 1/4" Archive tape is attached to server_name with host number host_number

Is this correct? (y/n): y

When this last phase has been completed, setup asks you whether you want to institute the configuration you have designed and, if you confirm, proceeds to edit several of the database files:

You have completed the configuration questions.
Continuing will destroy any existing files under /usr and any client partitions if you are a server.

Do you want to begin configuration? (y/n): y

Updating /etc/hosts
[ ... A few lines of configuration messages ... ]

If your distribution is on two 1/4-inch tape cartridges, setup will prompt you to change to the second cartridge very shortly after it begins its back-end routine. Insert the second cartridge and type 'RETURN' to continue the routine; it takes approximately 25 minutes to complete. When setup is done with its back-end work, your shell prompt returns. You can then continue your installation by booting the full UNIX system, as described below.

5.13. Booting the Full UNIX System

Finally, you boot the full UNIX system. You must first halt the system, using the /etc/ha�t command. This shuts down the system in an orderly manner, and returns control to the monitor:

```bash
# /etc/ha�t
Syncing disks . . . . done
>
```

and now you can simply boot the UNIX system:
You can now use the UNIX system on this machine.

5.14. Loading the Manuals, Demos, and Games Directories

The seventh file on the distribution tape(s) contains tar images of the /usr/man directory (online manual pages), /usr/demo directory (demonstration programs), and /usr/games directory (games). If these have been loaded on your remote host, you can load all or part of these files if you want to. Space may be a consideration: /usr/man consumes 2.1 MBytes, /usr/demo consumes 3.75 MBytes, and /usr/games takes 1.8 MBytes.

1) Become super-user.
2) Mount the 1/2" distribution tape or insert the first 1/4" cartridge.
3) Type the following on your target machine:

```bash
# cd /usr
# rsh server_name mt -f /dev/nrtape0 rew
# rsh server_name mt -f /dev/nrtape0 fsf 8

Then type the following command, specifying only the directories you want (man and/or demo and/or games). If you are using a 1/2" tape, use 20 for blk_factor; use 128 if you are using a 1/4" tape:

```bash
# rsh server_name dd if=/dev/nrtape0 bs=blk_factorb | tar xfpB - man demo game
```
Chapter 6

System Set-Up and Operation

This chapter describes procedures used to set-up and operate a Sun Workstation UNIX system. Set-up material applies to a first-time installation: in most cases, it includes background information on and installation procedures for the various available utilities — like the mail system, syslog, uucp, and USENET — you may wish to use. Operations procedures described here are used periodically to reboot the system, analyze error messages from devices, do disk backups, monitor system performance, and so on.

6.1. Network Configuration

6.1.1. Setting up a Gateway Machine

If you have a machine with two Ethernet Boards, you must edit /etc/hosts and /etc/rc.local 'by hand' to set up a proper configuration. Note that since each machine on a network must have a unique hostname associated with each unique Internet address, a machine with two boards must have a 'second identity'; it must be assigned a second hostname and Internet address. Proceed as follows:

1. Edit /etc/hosts on the gateway machine, and add a line for the machine's 'second' hostname and address. For example, say "jekyll" has two Ethernet Boards: the machine is a server for a network of clients, and also is a gateway to another network containing a printer and so on. Several lines of /etc/hosts might look like this:

   # Fantasy Local Net 192.9.200 -- 10Mb/s Ethernet -- Engineering
   #
   192.9.200.1  jekyll loghost datehost
   192.9.200.2  usher
   192.9.200.3  lenore
   [ etc. ]
   # Fantasy Local Net 192.9.4 -- 10Mb/s Ethernet -- Marketeering
   #
   192.9.4.1  quasimoto
   192.9.4.2  godzilla
   192.9.4.3  rodan
   [ etc. ]

   Add an entry for the machine's second address:
For administrative purposes, it makes sense to use two similar hostnames — people on both nets can address the machine by the same name, and you can tell the difference.

2. Edit `/etc/rc.local`, and add an `ifconfig` line mapping the machine's second hostname and Ethernet Board. Taking our example above, here is the relevant line from the file before:

```
/etc/ifconfig ec0 jekyll -trailers up
```

and after:

```
/etc/ifconfig ec0 jekyll -trailers up
/etc/ifconfig ec0 jekyll-hyde -trailers up
```

3. Finally, if you wish, you can copy the gateway's `/etc/hosts` file to all machines on both networks. This is not necessary, but may make for ease of administration.

### 6.1.2. Reducing Network Overhead

It is possible to have most of the advantages of living in a network environment without running the network daemons — `rwhod(8C)` and `routed(8C)` — on every machine. For systems with only 1 MByte of local memory, it may be desirable to disable the daemons, or run them only on specific machines, so that the space consumed by them can be made available for other purposes. When running, `routed` and `rwhod` are actively doing something many times a minute, and so leave many of their pages in memory almost all the time. For a 1-MByte system with 600K of available user memory, the memory thus effectively consumed is about 7%. Combined, both daemons use up about 14% of available user memory.

`rwhod` allows the programs `rwho(1C)` and `ruptime(1C)` to return status information about machine usage of hosts on the local network. Unless you really need this information, we suggest you do not run `rwhod`. By default, `rwhod` is not run. If you choose to run it, remove the comment mark (a `#` in column 1) from the following line in `/etc/rc`:

```
#/usr/etc/in.rwhod & echo -n 'rwhod' >/dev/console
```

Normally, `routed`, the routing daemon, runs on each Sun Workstation, and maintains network routing information that enables your machine to pick the best path for sending packets to external networks. `routed` consumes a small amount of memory and CPU time running an algorithm to keep accurate information about the topology of the local network.

Whether or where you should run `routed` depends on your system configuration. In general, if you have more than 1 MByte of memory in your workstation then it probably isn't worth
thinking about whether you should run *routed*, just run it. If your memory is limited, see which one of the following best applies to you, and follow through:

1. If you have no gateway machines in your local network then you don’t need to run *routed*. You can disable the daemon by removing (or commenting out — insert a ‘#’ before each line) the following three lines in your */etc/rc.local* file:

   ```
   if [-f /usr/etc/in.routed ]; then
     /usr/etc/in.routed & echo -n ' routed' > /dev/console
   fi
   ```

2. If you are a gateway or a server for diskless nodes then you must run *routed*.

3. If you have only 1 MByte of memory and only one gateway in your local network, then you can run *routed* only on the gateway machine, and disable it on all other nodes. All machines on the local network can then redirect packets going to another accessible network through this gateway machine. To do this, edit the */etc/rc.local* file on all machines except the gateway (gateway, in the example). Find the line that says:

   ```
   echo -n 'local daemons:' > /dev/console
   ```

   Insert the following two lines just before it:

   ```
   /usr/etc/route add 0 gateway 1
   echo ' /usr/etc/route add 0 gateway 1' > /dev/console
   ```

   Then, find the following three lines and comment them out (insert a ‘#’ before each line) or remove them:

   ```
   if [-f /usr/etc/in.routed ]; then
     /usr/etc/in.routed & echo -n ' routed' > /dev/console
   fi
   ```

4. Finally, if you have only 1 MByte of memory and you have more than one gateway in your local network and you are running diskless, you can use your server’s routing tables. Edit */etc/rc.local* file on all clients of the server machine (server, in the example), and add the following lines:

   ```
   /usr/etc/route add 0 server 1
   echo ' /usr/etc/route add 0 server 1' > /dev/console
   ```

   Then comment out or remove the daemon’s lines:

   ```
   if [-f /usr/etc/in.routed ]; then
     /usr/etc/in.routed & echo -n ' routed' > /dev/console
   fi
   ```

Since a diskless node cannot run when its server is down, this always works; however, it causes packets to be sent through the server, loading it down.

In all other cases you should run *routed*.

### 6.1.3. Network Security — */etc/hosts.equiv* and */rhosts*

Network security is implemented at two levels: first, at the machine or node level, and, second, at the individual user level. The */etc/hosts.equiv* and */rhosts* files, respectively, control access at these levels. The security-checking process goes like this:
1. When a user initiates a remote process on another machine (rlogin, rsh or rcp, for example), the system first checks for an entry for this user in /etc/passwd on the remote machine. If no entry is found, the user will be denied access: if he is trying to rlogin to the machine, he will be prompted for a password and then get a "Login incorrect" message; if he is attempting a rcp or rsh, he will get a "Login incorrect" message.

2. If an entry for the user is found in /etc/passwd, the system next checks for his machine's hostname in the other machine's /etc/hosts.equiv file. If the hostname is found, the user gains access.

3. If no /etc/hosts.equiv entry is found, the system checks for a line with his hostname (and, optionally, username) in the .rhosts file in his home directory on the other machine. If the entry is found, the user gains access.

If no entry is found in either /etc/hosts.equiv or "USERNAME/.rhosts", but the user is in /etc/passwd, the user is allowed to rlogin to the machine after giving his password, but gets "Permission denied" messages when attempting remote processes like rcp or rsh.

The single exception to this security scenario is the super-user: the system skips the second-level check (/etc/hosts.equiv is not checked), and goes directly to looking at /.rhosts.

So, if you want to allow access to your machine by all users on another specific machine, include an entry for each user in /etc/passwd and include the machine's hostname in your /etc/hosts.equiv file. For example, if my machine's hostname is gaia, and I want to allow anyone on host kepler to gain access to gaia, I simply edit my /etc/hosts.equiv file as follows. The file is just a list of hostnames, one per line:

```plaintext
core
ganymede
krypton
```

Add kepler's name to the list:

```plaintext
core
ganymede
krypton
kepler
```

Now all users who can gain access to kepler can also freely rlogin(1) to gaia (without being asked for a password), and can rcp(1) from and use rsh(1) on gaia, provided they are in gaia's /etc/passwd file.

If you want to allow access to some users on a particular machine but not all, do not put the machine's hostname in /etc/hosts.equiv. Instead, put it in the .rhosts file in each user's home directory on your machine ("USERNAME/.rhosts"). Note that, to avoid some security problems, this file must be owned by either this user or root, and must not be a symbolic link. The .rhosts file has a slightly different format than /etc/hosts.equiv: /etc/hosts.equiv accepts only hostnames; .rhosts accepts a hostname and, optionally, a user name on each line. Format is best illustrated by an example. I can allow user donald at host canard to have access to my machine, gaia, and keep other users of canard out by (1) removing other users' entries from /etc/passwd (or changing their passwords), (2) making sure canard is not in my /etc/hosts.equiv file, and (3) adding an entry for donald at canard to /usr/donald/.rhosts. The entry looks like this:

```plaintext
canard donald
```

Note also, that this means donald only has access when he's coming from canard; if he tries to use gaia from another host, he must know his password to be able to rlogin and can't complete
remote processes. Of course, if he can \texttt{rlogin} he can always doctor his \texttt{.rhosts} file (if you have given him a home directory). This brings up two points:

1. The only way to achieve anything resembling security in a hostile environment is to exclude users from the \texttt{/etc/passwd} file; once someone knows a password, he's in. If this concerns you, you should also be especially careful to protect \texttt{.rhosts} properly — make sure it is writable only by root. Clearly, tight security has not really been an issue in the administration of previous UNIX systems; the goal has been to facilitate, rather than deny, access.

2. If you are dealing with a more open environment, the easiest way to administer machine access at the user level is to give each 'trusted' user an account (in \texttt{/etc/passwd}) and a home directory on your machine(s), and ask each user to create his/her own \texttt{.rhosts} file in his/her home directory on the machine(s).

\section*{6.1.4. Making 1.1 Networks Compatible with Existing Networks}

The shift in network addressing format in Release 1.1 should only be a concern for you if

\begin{itemize}
  \item You have a network of machines running version 1.0 (or lower) of Sun software, and you want them to communicate with your new workstation(s) running 1.1, or
  \item You have a machine which cannot perform ARP (an older VAX, for example), and you want it to talk to your 1.1 Sun's.
\end{itemize}

If neither of these is the case, don't worry about it.

To make machines running 1.1 talk to 1.0 networks or machines which do not respond to or perform ARP, you can do one of three things:

1. The best path is to convert your 1.0 network to the Class C addressing scheme described in the previous section. If this is impossible in your system — for example, if you have older Vaxen that cannot perform ARP (some 4.1c machines) — then try solution 3. Otherwise, we urge you to convert.

   This is easier than it sounds. You'll need to assign your old network a new network number (use "192.200.1" if you wish); assign each machine on the old network a new, unique host number between 1 and 254; and, finally, edit the \texttt{/etc/hosts} file and change your old network number to your new network number, and identify each host with its host number. For example, if my old \texttt{/etc/hosts} file looked like this:

   \begin{verbatim}
   125.5143 alpha
   125.5204 beta
   125.6422 gamma
   125.0x5245 delta
   125.0x2226 epsilon
   \end{verbatim}

   My new one might look like this:

   \begin{verbatim}
   192.9.200.1 alpha
   192.9.200.2 beta
   192.9.200.3 gamma
   192.9.200.4 delta
   192.9.200.5 epsilon
   \end{verbatim}

   In the example above, "192.9.200" is the network number, and "1" is alpha's host number. Then, install \texttt{/etc/hosts} on all systems. Finally, reboot any machine whose host number you have changed.
2. A second solution is to retain your Class A addressing scheme from 1.0, but make sure all host numbers are smaller than 1024, so that ARP can be performed.

As above, this means editing /etc/hosts and assigning each machine a host number — the last component of the entire address — under 1024. You can simply retain your old network number. Remember to install the /etc/hosts file on all systems, and to reboot any machine whose host number you have changed.

3. Finally, if you have machines that either cannot perform ARP (old Vaxen, for example), or cannot respond to ARP (older Sun systems with Class A networks and host numbers greater than 1024, for example), proceed as follows.

You can allow machines which cannot perform ARP talk to a 1.1 network by 'forcing' your 1.1 machines to respond to an address which non-ARPers understand. Do this by editing the /etc/rc.local file on each 1.1 machine, and adding an extra 'ifconfig' line with each 1.1 machine's 'old' 6-byte hexadecimal Ethernet address:

a) You can obtain this address for a 3COM Ethernet Board by using the PROM Monitor to interrogate the Multibus memory-space where the address is stored. As super-user, run the following sequence of commands (type <RETURN> where indicated):

```
# /etc/fasthalt
syncing disks . . . done
Unix Halted
> k1
> e fe0400
> FE0400: 0260? <RETURN>
> FE0402: 8C00? <RETURN>
> FE0404: 9920? q
```

b) Then, edit the /etc/rc.local file, and add the following line, substituting the machine's entire 6-byte hexadecimal Ethernet address for old_ethernet_address:

```
/etc/ifconfig ec0 old_ethernet_address 'hostname' -trailers up
```

For instance, the lines for the machine used in the example above might look like this:

```
/etc/ifconfig ec0 'hostname' -trailers up
/etc/ifconfig ec0 2:60:8C:0:99:20 'hostname' -trailers up
```

c) Finally, reboot any machine whose file you have altered in this way.

If you have machines that will not respond to ARP, you can use the /etc/arp command to specify the Ethernet addresses for the machines (see arp(8)).

### 6.2. Setting Up the Mail System

The mail system consists of the following commands and files:
Special note for file server configurations:

On file servers and diskless clients the directory /usr/lib resides on the shared, public disk. Since this disk is read-only and is shared by all the systems, files that need to be writable or need to be different on each machine cannot reside here. Instead there is a directory called /private that contains the files that would normally reside in /usr/lib. The setup program replaces the files in /usr/lib by symbolic links to the files in /private so the files can still be referenced by their normal names. The files (and directories) so affected are listed below:

```
/bin/mail
/usr/ucb/mail
/usr/lib/sendmail
/usr/lib/sendmail.cf
/usr/lib/sendmail.domain.cf
/usr/lib/sendmail.generic.cf
/usr/spool/mail
/usr/spool/mqueue
/usr/spool/secretmail
/usr/bin/xsend
/usr/bin/xget
/usr/lib/aliases
/usr/lib/aliases.dir
/usr/lib/aliases.pag
/usr/lib/crontab
/usr/lib/uucp
/usr/lib/news
```

When the following instructions tell you to copy or edit one of the above files, instead use the corresponding file in /private.

Mail is normally sent and received using the `mail(1)` command, which provides a front-end to edit the messages sent and received, and passes the messages to `sendmail(8)` for routing. The routing algorithm uses knowledge of the network name syntax, aliasing and forwarding information, and network topology, as defined in the configuration file /usr/lib/endmail.cf, to process each piece of mail. Local mail is delivered by giving it to the program /bin/mail which adds it to the mailboxes in the directory /usr/spool/mail, using a locking protocol to avoid problems with simultaneous updates. After the mail is delivered, the local mail delivery daemon /usr/etc/in.comsat notices, and it notifies users who have issued a "biff y" command that mail has arrived.

Mail for `username` is normally accessible in the file /usr/spool/mail/username. In the distribution system, your mail file is readable only by you; however, anyone with the super-user password can read others' files, including their mail. To send mail which is secure against any possible perusal (except by a code-breaker), use the secret mail facility, which encrypts the mail so

---

Revision H of 12 March 1984
that no one can read it. See send(1). Note that this facility does not work over the network. The following subsections give some instruction on sendmail installation; for more detailed information, see the Sendmail Installation and Operation Guide in the final section of this manual.

6.2.1. Picking a Name for your Domain

The network mail delivery program, sendmail, uses Internet standard mail addressing formats which make it possible for any Internet system in the world to send or receive mail with any other Internet system. To accomplish this, each system must have a unique name. To make it easier to assign names, the world of mail addresses is broken up into domains and subdomains.

For example, many systems funded by the U.S. Government’s Advanced Research Projects Agency are in the domain “arpa”. Many of the systems which send mail via the uucp protocols are in the domain “uucp”. Within each domain, names have to be unique — there can’t be two machines called “MIT-Multics.arpa” or it would be impossible to decide how to deliver mail to them.

Domains nest inside one another like directories, except that the names go from right to left. Thus “joe.sun.uucp” is host “joe” in subdomain “sun” which is in domain “uucp” just like “usr/lib/crontab” is file “crontab” in subdirectory “lib” in directory “usr”. To make life easier for the people who maintain mailers, there are only a small number of “top-level” domains like “uucp” and “arpa”.

If your workstation is on the Arpanet, or shares an Ethernet with a machine on the Arpanet, or with a machine on the Arpa Internet, you should probably pick a name in the “arpa” domain, and register it with the Internet naming authority at the Arpanet Network Information Center. Otherwise, pick a name in the “uucp” domain.

If your workstation and/or Ethernet have no phone lines or connections to other networks, the name you pick within the “uucp” domain doesn’t matter much. You may, however, have to change the name if you get other network connections and somebody else is already using that name.

If your organization already has hosts on the uucp network or the Usenet, ask a system administrator for help in picking a name.

If you are connected to the uucp network, you must be talking with at least one other computer on the network. Check with the system administrator of that machine to see if the name you have picked is already in use in the uucp network. If they are on the Usenet, they can look in newsgroup “net.map”; if not, they just have to guess.

In a network of Suns, the name of the “main machine” becomes the name of your subdomain, and each other machine can truly have any name you want. For example, a main machine at a site might be called “fred”; various other machines on its Ethernet could be “shirl,” “sal,” etc. The fully qualified name of the “fred” machine is “fred.uucp” and must be unique in the uucp world; but the full name of “shirl” is “shirl.fred.uucp” and its uniqueness is guaranteed by the uniqueness of “fred.uucp”.

If you only have one machine, your host name and your domain name (within the “uucp” or “arpa” top-level domain) are the same.
6.2.2. Picking a “Main Machine” for Mail Forwarding

To begin configuration, you must tell sendmail whether your system is the “main machine” in a network, or a “subsidiary machine” in a network.

If your machine is attached to an Ethernet and is also attached to phone lines, it is a “main machine”. Pick one such machine on the network and make it your mail machine; treat all the rest as “subsidiary machines” for configuration purposes. Note that if you have a standalone system (a machine which is not attached to an Ethernet), sendmail treats it like the “main machine” in a one-machine network.

If your machine is attached to an Ethernet and you don’t have any phone lines, but there is an Arpanet host on your Ethernet, you can pick any workstation as your “main machine”. If your Arpanet host runs sendmail, it can be your “main machine”. All the other Suns will be subsidiary machines.

Similarly, if you have several machines on an Ethernet, and none of them have phones, pick one as the main machine and leave the rest as subsidiary machines.

6.2.2.1. Configuring for Mail Forwarding

Next, you must define each machine’s mailer configuration. The following commands accomplish this. Remember that if your machine is a file server or a diskless client, the sendmail.cf file is in /private/sendmail.cf.

To configure a “main machine”:

```
# cp /usr/lib/sendmail.domain.cf /usr/lib/sendmail.cf
```

To configure a “subsidiary machine”:

```
# cp /usr/lib/sendmail.generic.cf /usr/lib/sendmail.cf
```

6.2.2.2. Telling Sendmail your Domain Name

To tell sendmail what your domain is, edit the file /usr/lib/sendmail.cf on the “main machine” and all the subsidiary machines. Remember that if your machine is a file server or a diskless client, the sendmail.cf file is in /private/sendmail.cf. Near the top of the file is a block of lines that looks like this:

```
# local domain names
DDsun
CDsun
```

This defines our domain name as “sun” within the “uucp” domain — in other words, “sun.uucp”. Change these lines to reflect the name you picked. For example,

```
# local domain names
DDfred
CDfred
```

defines your domain name as “fred.uucp”. The hostname of your main machine (as set up by the hostname command) is “fred,” and subsidiary machines, if you have any, will be called (for example) “shirl.fred.uucp” for a machine whose hostname is “shirl.”
If your top-level domain is not "uuucp", find the lines that look like:

```
# domain-spec for local domain within universe (e.g., what domains are above?)
# should be integrated into mainline (e.g. Berkeley) config files
DUuucp
```

They should be the next thing in the file. Change "uuucp" to your top-level domain name (for example, "DUarpa").

Now, if you are editing `sendmail.cf` for a subsidiary machine, look for a block of lines like this:

```
# major relay host
DRmailhost
CRmailhost
```

Change the name "mailhost" to the name of your domain:

```
# major relay host
DRfred
CRfred
```

If you are editing `sendmail.cf` for a subsidiary machine which has phone lines attached to it, you can have `sendmail` route `uuucp` mail to certain hosts via the local phone lines, rather than having all `uuucp` traffic go through the "main machine". To do this, find the lines that look like:

```
# local UUCP connections -- not forwarded to mailhost
CV
```

Put the names of the local `uuucp` sites on the end of the "CV" line, or on additional CV lines. For example,

```
CV rome prussia georgia
```

This completes the `sendmail.cf` editing for the subsidiary machine. Note that if you have more than one subsidiary with no local `uuucp` connections, you can edit the file just once and then copy it to all the subsidiary machines with `rcp(1)`.

On your main machine, you can make one optional change. If one of the machines with which you have a `uuucp` or Ethernet connection is on the Arpanet and will relay mail for you, look for a block of lines like this:

```
# major relay mailer
DMuuucp

# major relay host
DRucbarpa
CRucbarpa
```

Note that similar lines appear twice in the file — you should change the second set. Edit in the mailer that you connect to this host with ("uuucp" or "ether") and the name of the relay host. For example, if you share an Ethernet with a machine called "cmu-cs-vlsi," which is on the Arpanet, your entry might look like this:
# major relay mailer
DMether

# major relay host
DRcmu-cs-vlsi
CRcmu-cs-vlsi

On the other hand, your relay point might be uucp host "ucbvax":

# major relay mailer
DMuucp

# major relay host
DRucbvax
CRucbvax

This change will let you mail to addresses like "charlie@mit-ai.arpa" and even though you aren't on the Arpanet, the message will get through. If you don't have an Arpanet relay point, don't worry about this.

This completes the mailer configuration process.

6.2.2.3. Setting up the "Postmaster" Alias

Edit the file /usr/lib/aliases (or /private/aliases for a file server or diskless client workstation) and look for the entry for "postmaster". This is where people will send mail if they want to get in touch with someone at your site who can deal with mailer problems (you can send mail to "postmaster"s at other network sites if you have problems with mail that comes from there). The line will look like this:

```
# Following alias is required by the new mail protocol, RFC 822
# Set it to the address of a HUMAN who deals with this system's mail problems
Postmaster:root
```

If you manage the mail system for your site, change the name "root" to the user name you usually log in with, so that messages directed to the Postmaster of your machine will arrive with your usual mail.

If you manage the mail system for several workstations, change the file on all of them to forward Postmaster mail to wherever you usually read mail. For example, if you are "henry" on machine "shirl," make the line read:

```
Postmaster: henry@shirl
```

You can also create aliases for people or groups of people ("mailing lists") at this time. See the examples in the aliases file. You can edit this file now or at any later time. Each time you edit the file, run the newaliases program to rebuild the alias database with your changes.
6.2.3. Testing your Mailer Configuration

The first thing to do is to reboot all the systems whose configuration files you have changed. Then, send test messages from various machines on the network with a command like this:

```
hal% /usr/lib/sendmail -v <?dev/null addresses
```

This will send a null message to the specified address, and display messages about what it is doing. Test that:

- You can send mail to yourself, or other people, on the local machine, by addressing the message to a plain user name ("root", for example).
- If you have an Ethernet, you can send mail to someone on another machine ("root@hobo," for example). Try this in three directions — from the main machine to a subsidiary machine, vice versa, and from a subsidiary machine to another subsidiary machine, if you have two. (Note that /etc/hosts.equiv must be set up on at least the main machine before this will work. See the subsection, Handling Network Security with /etc/hosts.equiv and /.rhosts in the Installing UNIX for the First Time chapter.)
- If you have a phone line and you have set up a uucp connection to another host, you can send mail to someone there and they can send it back (or call you on the phone, if they receive it). Try having them send mail to you. For example, you could send to "ucbvax!joe" if you have a connection to ucbvax. Sendmail won't be able to tell you whether the message really got through — since it just hands it off to uucp for delivery — so you have to ask a human at the other end. You might be able to get some idea of what's going on by looking in /usr/spool/uucp/LOGFILE; see the Uucp Implementation Description in the Tutorials section of this manual.
- Mail something to Postmaster on various machines and make sure that it comes to your usual mailbox, so when other sites send you mail as Postmaster, you're sure you will see it.

6.2.4. Diagnosing Troubles with Mail Delivery

The best tools for diagnosis of mail problems are:

- "Received" lines in the header of the broken message. These give a trace of which systems the message was relayed through on its way. Note that in the uucp network there are many sites that do not update these lines, and that in the Arpanet the lines often get rearranged. You can straighten them out by looking at the date and time in each line. Don't forget to take time zones into account.
- Messages from "MAILER-DAEMON" on various systems. These messages typically report delivery problems. More and more systems are producing these messages, rather than simply throwing away mail that they can't deliver.
- The system log, for delivery problems in your group of workstations. Sendmail records what it is doing all the time, and this information is kept in the system log. In the distributed system, the logs are kept for a week, then discarded. Log files are kept in /usr/spool/log on your network server machine (the system log configuration is taken care of by the setup program during UNIX installation — see syslog(8)). Today's log is in file syslog; the previous day's is syslog.0; two days' back is syslog.1, etc. If you have chronic trouble with mail, look at the log once in a while. At Sun, cron(8) runs a shell script nightly which searches the log for SYSERR messages and mails any that it finds to "Postmaster". This way, problems are often fixed before anyone notices them, and the mail system runs
more smoothly.

6.3. System Log Configuration

Various system daemons and programs record information in the system log to aid in problem analysis. They send this information as Internet datagrams directed to the 'syslog' daemon on host 'loghost'. The daemon receives these datagrams and records the information or notifies users of problems. See *syslog*(8) for more details on this process.

The default configuration runs a syslog daemon on each machine, and also keeps all datagrams on the local machine (by maintaining the 'loghost' alias in the */etc/hosts* entry for the local machine). If you are running standalone, this is how your system is configured. However, in a network environment it's much easier to track problems if all machines log their information in a single place. Therefore, during first time UNIX installation, the setup program re-configures things so that only the designated server is the 'loghost': setup strips the 'loghost' alias from the */etc/hosts* entries for the clients, and adds the alias for the server machine. This means that, for example, if the machine named krypton is your network server, the beginning of your machines' */etc/hosts* files might look like:

```
192.9.1.1  krypton  loghost
192.9.1.2  wally
192.9.1.3  beaver
192.9.1.4  june
192.9.1.5  eldridge
```

Now all datagrams sent to 'loghost' (from wally, beaver, etc.) are sent to krypton. There might also be other aliases on the same line of the */etc/hosts* entry, like 'lprhost' or 'mailhost' — that's OK. Note also that, since the syslog daemon only starts up when messages must be handled, only the 'loghost' runs the daemon. Getting rid of the daemon on all other machines frees up resources and makes the system start up faster.

If you want to change this configuration — for example, if you have more than one server, and you want only one 'loghost' — simply change the placement of the 'loghost' alias, and then re-copy */etc/hosts* to all machines. Test your system log configuration by running:

```
% tail -f /usr/spool/log/syslog
```

on the loghost machine, then sending any kind of mail on the various other machines. Each message sent will generate four or five lines of output if things are working.

6.4. Setting Up a UUCP Connection

*uucp* (UNIX to UNIX copy) is a series of programs designed for communication, via dial-up or hardwired lines, between two systems running UNIX. *uucp* may be used to transfer files between UNIX systems, and also to run commands on remote machines. For more detailed background, see the *UUCP Implementation Description* in the *Tutorials* section of this manual.

Support for *uucp* is located in three major directories: */usr/bin* (which contains user commands), */usr/lib/uucp* (operational commands), and */usr/spool/uucp* (spooling area).
The commands in /usr/bin are:

- /usr/bin/uucp: file-copy command
- /usr/bin/uux: remote execution command
- /usr/bin/uusend: binary file transfer using mail
- /usr/bin/uuencode: binary file encoder (for uusend)
- /usr/bin/uudecode: binary file decoder (for uusend)
- /usr/bin/uulog: scans session log files

The important files and commands in /usr/lib/uucp are:

- /usr/lib/uucp/L-devices: list of dialers and hardwired lines
- /usr/lib/uucp/L-dialcodes: dialcode abbreviations
- /usr/lib/uucp/Lcmds: commands remote sites may execute
- /usr/lib/uucp/L.sys: systems to communicate with, how to connect, and when
- /usr/lib/uucp/SEQF: sequence numbering control file
- /usr/lib/uucp/USERFILE: remote site pathname access specifications
- /usr/lib/uucp/uucico: uucp protocol daemon
- /usr/lib/uucp/uucp.day: script for daily polling/cleanup
- /usr/lib/uucp/uucp.hour: script for hourly polling
- /usr/lib/uucp/uucp.night: script for nightly polling
- /usr/lib/uucp/uucp.noon: script for midday polling
- /usr/lib/uucp/uucp.week: script for weekly cleanup of uucp log files
- /usr/lib/uucp/uupoll: site-polling script
- /usr/lib/uucp/uuclean: cleans up garbage files in spool area
- /usr/lib/uucp/uuxqt: uucp remote-execution server

The spooling area, /usr/spool/uucp, contains the following important files and directories:

- /usr/spool/uucp/C: directory for command, “C.” files
- /usr/spool/uucp/D: directory for data, “D.”, files
- /usr/spool/uucp/D.hostname: directory for local “D.” files
- /usr/spool/uucp/LOGFILE: log file of uucp activity
- /usr/spool/uucp/SYSLOG: log file of uucp file transfers

Note that C., D., and D.hostname are subdirectories, unlike earlier implementations of uucp. In older versions, C. and D. files are placed directly into the spooling directory, /usr/spool/uucp; in the current version, they are placed in their appropriate subdirectory. So, in the old version you’d have, say, /usr/spool/uucp/C.dea45n0031; in the new version the file would be /usr/spool/uucp/C./C.dea45n0031.

As uucp operates it creates (and removes) many small files in the directories underneath /usr/spool/uucp. Sometimes files are left undeleted; these are most easily purged with the uuclet program. Instructions in the uucp.day file take care of doing this daily clean-up for you. The uucp log files can grow without bound unless trimmed back; uulog is used to maintain these files. uucp.day and uucp.week manage this housekeeping. If you decide to prune these directories yourself, be careful: randomly removing files from /usr/spool/uucp may cause uucp to generate error messages when it tries to access a file another file claims is there. (For instance, each mail transaction creates three files.) You do, however, need to clean the /usr/spool/uucppublic directory 'manually'; this is a place for people at other sites to send to when sending files to
users at your site.

_uuep_ occasionally sends mail about minor problems to "uucp" or "root". Your maintenance person should also read and toss these messages. You can redirect the mail to your mailbox by putting an entry for "uucp" in _/usr/lib/aliases_. Look at the examples there.

Under normal conditions, _uuep_ calls your designated sites at specified times and, while it's at it, checks to see if anything should come back. If you ever need to invoke _uucp_ 'on command', the line:

```
# /usr/lib/uucp/uucico -r1 -s sitename
```

forces _uucp_ to poll _sitename_, even if there is nothing waiting. If you do run _uucp_ in this fashion, don't run it as super-user, since the suid* bit will not be honored. If you are having trouble with the connection, run _uucp_ with the debugging option, as described in installation step 7, below.

### 6.4.1. _UUCP_ Installation Overview

A _uucp_ network link using modems or dedicated lines may be established between two machines running UNIX systems. To establish a connection between two sites that both have modems, one site must have (at least) an automatic call unit (an auto-dial modem) and the other must have (at least) a dialup port (an auto-answer modem). It is better if both sites have one of each or, as is standard, have modems which both call and answer, like Ventel's.

If both you and the site(s) you wish to connect with have autodial units and ports, install _uucp_ as follows. If you have only a dialup your situation will be different; procedures are described near the end of this section. For more information, read the _Uucp Implementation Description_ in the final section of this manual. It describes in detail the file formats and conventions, and will give you necessary context.

1. Select a site name (less than 8 characters): the name of the machine which will be your _uucp_ connection to the outside world. We will call this machine your 'uucp host'; its name is your 'uucp hostname'.

   If this machine is your "main machine" (see _Setting Up the Mail System_, above), then your _uucp_ hostname should be the same as your domain name.

2. Change the _/usr/spool/uucp/D.hostname_ directory to your own site's _/usr/spool/uucp/D.hostname_ directory with the following:

   ```
   # mv /usr/spool/uucp/D.hostname /usr/spool/uucp/D.hostname
   ```

   Use your _uucp_ hostname for _hostname_.

3. Create a _uucp_ account in the password file on your _uucp_ host machine by entering a line of the following form in _/etc/passwd_

   ```
   Uhostname:*:4::/usr/spool/uucp:/usr/lib/uucp/uucico
   ```

   Now use the _passwd_ command to establish a password for your host:

   ```
   # passwd Uhostname
   ```

* "suid" stands for "set user i.d."; if you set this bit on an executable file, UNIX will grant or deny file access based on the permissions of the file's owner, rather than the permissions of the person who executes the file. _Uucp_ uses this facility to ensure that all the files in its spool directories are readable and writeable no matter who invokes the _uucp_ program.
4. The *L.sys* file contains the phone numbers and login sequences required to establish a connection with a uucp daemon on another machine. Edit `/usr/lib/uucp/L.sys`, adding a line of the following form for each site you want to talk to:

```
their_host Any device baud phone# login:-EOT-login: uucp ssword: pass
```

The first field is the uucp hostname of the other site, the second indicates when their host may be called, the third field specifies how their host is connected (through an ACU, a hardwired line, etc.), then comes the baud rate of the line, phone number to use in connecting through an auto-call unit, and finally a login sequence. The phone number may contain common abbreviations which are defined in the *L-dialcodes* file. The device specification (third field) should refer to devices specified in the *L-devices* file. Note that the only modem type currently supported by Sun Microsystems is the Ventel MD212 modem. Indicating only “ACU” causes the uucp daemon, uucico, to search for any available auto-call unit in *L-devices*. For example, our *L.sys* file looks something like:

```
adiron Any ACUVENTEL 1200 7620883 login:-EOT-login: uucp ssword: secret
ucbvax Any,20 ACUVENTEL 1200 6728212% login:-EOT-login: uucp ssword: almama
ucbarpa Any,20 ACUVENTEL 1200 6424351 login:-EOT-login: uucp ssword: netnut
decvax Any ACUVENTEL 1200 6039941241 login:-EOT-login: Ujedi ssword: cannon
```

For hardwired lines, your *L.sys* lines should contain the tty device name in the third and fifth fields:

```
anyname Any ttyb 1200 ttyb login:-EOT-login: uucp ssword: sunrise
```

5. Connect your (auto-dial/auto-answer) modem to ttya (the port labelled 'RS-232-A' on the backpanel of your Sun Workstation) on your host machine.

6. Create the appropriate device for your modem with the following series of commands:

```
# cd /dev
# mknod cuaO c 12 128
# ln -tta y ttya cua0
# chmod 600 cua0
# chown uucp cua0
# p< x ttya ttyd0
```

7. Edit `/etc/ttyd` to include an entry for `ttyd0` (see `ttyd(4)`). Insert the line: "13ttyd0". Then type the following to initialize everything properly:

```
# kill -1 1
```

8. Make sure your modem is hooked up properly by running `tip` with the phone number of a known machine:

```
# tip 6423345
```

If you get a “dialing . . . connected” response, all is well. If you get any other response, reconnect your modem.

9. As a final test, run uucp with the debug option (`-x`), as follows:

```
#/usr/lib/uucp/uucico -r1 -saitemane -x7
```

With the `-x` option, the higher the number, the more debugging output you get; 1, 4, and 7 are reasonable choices. If you get immense quantities of output from this command, everything is fine; you can go on to edit the following files.
10. Edit the files `uucp.day`, `uucp.noon`, and `uucp.night`, in the `/usr/lib/uucp` directory. Each of these files has a ‘for’ loop which arranges to call sites you want to call at designated times for mail. In each file, find the line that says:

```
for i in sys.name
```

and change “sys.name” to the site name(s) you want to poll. For example:

```
for i in ucbvax ucbarpa Shasta navajo
```

11. Add the `uucp.day`, `uucp.noon`, `uucp.night`, `uucp.hour`, and `uucp.week`, files to `/usr/lib/crontab` (see `cron(8)`), which arranges for the appropriate sites to be polled at the appropriate times. For example, the entries in `crontab` might look like:

```
5 6 * * * su uucp < /usr/lib/uucp/uucp.day  
15 12 * * * su uucp < /usr/lib/uucp/uucp.noon 
30 23 * * * su uucp < /usr/lib/uucp/uucp.night 
10,30,50 * * * su uucp < /usr/lib/uucp/uucp.hour 
0 7 * * 2 su uucp < /usr/lib/uucp/uucp.week
```

If you have only a dialup, you can be a second-class citizen on the uucp net. You must find another site that has a dialer, and have them poll you regularly (once a day is about the minimum that is reasonable). When you send mail to another site, you must wait for them to call you. To handle installation for a passive node, just complete steps one through four in the procedures above. When you come to the final step, editing `/usr/lib/uucp/L.sys`, you don’t need to specify all the information called for in the step: only the first two fields of `L.sys` are necessary, and in practice only the first field (site name) is looked at. A typical `L.sys` for a passive node might be:

```
ucbvax None
research None
```

where the first field on each line is a site that will poll you. Next, put a password on the `uucp` login. Then let the other site know your phone number, `uucp` login name, and password.

### 6.5. USENET Installation/Conversion

This section is intended to help a new USENET site install the network news software, and to give conversion guidelines to sites currently running USENET version 2.9. The news software is publicly available over the USENET; the news system released with the Sun 1.0 and the current software distribution is a Sun-supported version of USENET version 2.10.1. For more information on installation and maintenance (information on files, setting up links, control messages, maintenance of the news system, and how to create new newsgroups), see the `USENET Installation and Maintenance` paper in the `Tutorials` section of this manual.

#### 6.5.1. USENET Installation

The overall order of things to do is:

1. Find somebody to link up with. You need a network connection of some kind — most likely, `uucp`. If you use `uucp` and have no connections, you must have at least a dialup and preferably a dialer, and find someone willing to call your machine. You can use the USENET directory to locate some other site near yours to hook up to.

---

Revision H of 12 March 1984
2. Edit the /usr/lib/news/sys file, adding an entry for the site(s) you wish to connect to, and an entry for your own site. Sys is similar to the uucp L.sys file: it lists all your neighbors, which newsgroups they subscribe to, and describes how to send news to them. The format of the sys file is as follows:

Each line of sys contains four fields; fields are separated by colons:

1. The system name of a site to which you forward news. Normally, you should include a line for all systems you have links to, as well a line for your own system.

2. The newsgroups to be forwarded to them. This is a pattern in the sense of a subscription. Generally, you will list classes of newsgroups, that is, using all for everything. A typical forwarding list for a new site would be:

   net.all,fa.all,to.sysname

   where sysname is the name of your contact site. Note that you should not forward all, in particular, since local newsgroups (those without dots) should not be sent. In the line describing your own system, this field describes the newsgroups your site will accept from contact sites. Thus, if another site insists on sending you a newsgroup you don't want, say net.jokes, include !net.jokes here.

3. Flags describing the connection between sites. These are:
   - A Indicates that the contact site is running an A version of netnews, or
   - B The contact site is running a B version. If neither A nor B is indicated here, default is B. If you are running the latest release of Sun software, you have a B version. If you aren't sure which version your contact site is running, ask them before proceeding.
   - F Indicates that the fourth field is the name of a file. The full path name of a file containing the article in /usr/spool/news will be appended to this file.
   - L Prevents transmission unless the article was created on this site. Feeding an L link to a backbone site ensures that your submissions will be more likely to get to the entire network, even in the event of a local problem. Please make sure that a mail link exists too, so you can get replies.
   - U Arranges that the parameter to the optional %s in the command field be filled in with a permanent file name from /usr/spool/news instead of a temporary customized file name.

4. The command to be run to send news to the remote site. The article will be on the standard input. A %s in the command will be replaced with the name of the file that contains the article. Leaving this field blank means an ordinary uucp link is being used; that is, the command defaults to:

   uux -r -s sysname!rnews

Options here are:

- Tells uux to expect input on stdin.
- Tells uux not to start up a daemon right away. This eases the load on the system, and makes news transmission only a bit slower. News is sent when the next daemon is started, usually the next time uucico is invoked by cron.
- Shuts off the annoying message you would otherwise get mailed to you telling you that your article was successfully broadcast. The -s option is nonstandard; the remote system may need to be modified to understand it. To avoid using -s, put the uux command in the fourth field.
Here is a sample *sys* file for a site "zenith" with connections to "nadir". "Zenith" also passes news on to "lowerreaches". We assume that "zenith" and "lowerreaches" exchange a local newsgroup class lng.all as well as the network wide newsgroups.

zenith:net.all,fa.all,lng.all::
nadir:net.all,fa.all::
lowerreaches:net.all,fa.all,lng.all::

3. After adding the site you want to connect to to your *sys* file, have your contact at the other end of the link add you to their *sys* file.

4. Have the site you want to connect to send you their active file, and use the makeactive.sh shell script to create your local directory hierarchy and active file.

Active lists active newsgroups. The order of the list dictates the order in which news will be presented by readnews, so you might want to edit active later and arrange the newsgroups accordingly.

Each line of the active file contains two fields: the newsgroup name, and the highest local article number (for the most recently received article). The fields are separated by a space. Initially, your local active file will have 00000 as the article number for each newsgroup; local article numbers begin at 1 for the first article received in each group and increment within the newsgroup as articles are received. They do not usually correspond to local article numbers at other sites. The article number is always stored as a 5 digit number (with leading zeros) to allow updating of the file in place.

Active is automatically updated as new newsgroups come in. For information on how to create new newsgroups, see the Creating New Newsgroups section of the USENET Installation and Maintenance paper in the final section of this manual.

5. Post a message to the to.sysname newsgroup -- which should be set up to go only to the site you are linked to -- as a test (please don't use net.test unless there is no alternative. It is almost always possible to use test, or to.sysname, or some local.test group, instead of net.test). Have your contact send a message to your system using the same mechanism. If this doesn't work, find the problem and fix it.

6. Fill out a USENET directory form and post a copy to the USENET newsgroup net.news.newsite.

7. Post the etiquette and tencmdts files (in the /usr/lib/news/doc directory) to your general newsgroup with a long expiration date. Running rnews separately on each of these files will do.

6.5.2. Conversion from USENET Version 2.9 to 2.10

In this section, we give procedures for converting from version 2.9 (released with the Sun 0.4 and earlier software distributions) to version 2.10 (released with the Sun 1.0 and current distributions) of the USENET software.

There are some incompatibilities between 2.9 and 2.10. None of them should cause a problem, but you should be aware of them. For a discussion of the differences between the versions, see the paper in the Tutorials section of this manual, USENET Installation and Maintenance.

Overall recommended conversion order:

1. Become user "news":

Revision H of 12 March 1984

Sun 100/150 Installation Manual System Set-Up and Operation
hackster% su
Password:
SU: shannon /dev/console
hackster# su news

2. Restore your /usr/spool/news hierarchy and the following files from /usr/lib/news: active, sys, history.

3. Check your /usr/lib/news/active file and clean out any duplicates and old newsgroups.

4. Make the new spool directories:
   
   hackster% cd /usr/lib/news/cvt
   hackster% sh cvt mkdirs.sh /usr/lib/news /usr/spool/news

5. Convert the history file:
   
   hackster% cc cvt.hist.c -ldbm
   hackster% a.out /usr/lib/news

6. Convert the active file:
   
   hackster% sh cvt.active.sh /usr/lib/news /usr/spool/news

7. Make links to spooled articles (this takes some time):
   
   hackster% sh cvt.links.sh /usr/lib/news /usr/spool/news

8. Clean out the old spool hierarchy:
   
   hackster% sh cvt.clean.sh /usr/spool/news

Several shell scripts and C programs are provided (in the directory /usr/lib/news/cvt) to help with this conversion. The above scripts will convert active and create the new directory tree. The final script removes the old dot files and directories.

After the conversion, you should create two new newsgroups: junk and control. Junk is for messages that are in an invalid newsgroup (so you can see what's getting thrown away). Control is for control messages. Ordinary users probably won't want to read either, but as administrator you might want to. To create them:

   hackster% mkdir /usr/spool/news/junk /usr/spool/news/control

and add the following two lines to the end of /usr/lib/news/active:

junk 00000
control 00000
6.6. Setting Up the Line Printer System

The line printer system consists of at least the following files and commands:

- `/usr/ucb/lpq` spooling queue examination program
- `/usr/ucb/lprm` program to delete jobs from a queue
- `/usr/ucb/lpr` program to enter a job in a printer queue
- `/etc/printcap` printer configuration and capability data base
- `/usr/lib/lpd` line printer daemon, scans spooling queues
- `/etc/lpc` line printer control program

The file `/etc/printcap` is a master data base; it describes line printers directly attached to a machine and printers accessible across a network. The manual page `printcap(5)` describes the format of this data base and also indicates the default values for such things as the directory in which spooling is performed. The line printer system handles multiple printers, multiple spooling queues, local and remote printers, and printers attached via serial lines which require line initialization such as baud rate. Raster output devices such as a Varian or Versatec, and laser printers such as an Imagen, can also be handled by the line printer system.

Remote spooling via the network is handled with two spooling queues, one on the local machine and one on the remote machine. When a remote printer job is initiated with `lpr`, the job is queued locally and a daemon process is created to oversee the transfer of the job to the remote machine. If the destination machine is unreachable, the job will remain queued until it is possible to transfer the files to the spooling queue on the remote machine. The `lpq` program shows the contents of spool queues on both the local and remote machines.

6.6.1. Printer Configuration

You may connect a line printer with a serial interface to one of the serial ports on the Sun Workstation backpanel. Although this release of Sun software also supports the Centronics and Versatec interfaces provided by the Systech board, the following section is geared only to printers with serial interfaces.

The basic procedure for configuring a line printer into the system is described in the following. For more detail — or if you have problems — refer to the Line Printer Spooler Manual in the Tutorials section of this manual.

1. Attach a line printer with a serial interface to one of the serial ports on the workstation backpanel. Models 100U/150U and 120 provide ports A and B; if you have a Sun-2/120 or Sun-2/170 with a SCSI Interface, ports SIO-S0, SIO-1, SIO-S2, and SIO-S3 are also available.

2. The next step is to make sure that `init(8)` does not create a login process on the port you are using for your printer when the system comes up multi-user. Do this by editing the `/etc/ttys` file. Verify that the entry for the port (ttya, ttyb, etc.) has a zero as the first character on the line. For example, if you are using serial port A for your line printer, the first few lines in `/etc/ttys` might look like:
3. If you had to change `/etc/ttys` in step 2, run the following command (as super-user) afterwards to bring the changes into effect:

   ```
   % kill -1 1
   ```

4. Check the following files by using a `"ls -ld"` command, and verify that permissions, owner, and filename are the same as this list:

   ```
   -rws--s--x 1 root daemon 57344 Oct 20 13:35 /usr/lib/lpd
   -rws--s--x 1 root daemon 30720 Oct 20 13:35 /usr/ucb/lpr
   drwxrwx--- 2 daemon daemon 512 Oct 29 12:32 /usr/spool/lpd
   ```

5. Next, edit the `/etc/printcap` file to add an entry for your printer. Here are a few sample `printcap` entries for various printers/plotters:

   ```
   #
   # DecWriter over a tty line.
   lp[ap]arapa|ucbarpa|LA-180 DecWriter III:U:
   # br#1200:fs#06320:tr==of==/usr/lib/lpf:lf=/usr/adm/lpd-errs:
   #
   # typical remote printer entry
   ucbvax|vax|vx|ucbvax line printer:U:
   :lp==rm=ucbvax|sd=/usr/spool/vaxlpd:lf=/usr/adm/lpd-errs:
   ```

   All of the above fields are explained on the `printcap(5)` manual page. For more information on the entries for printers on serial lines and remote printers, see the `Line Printer Spooler Manual` in the `Tutorials` section of this manual.

6. Most printer characteristics can be specified in `/etc/printcap`, but not all; you may need to use `stty(1)` or an output filter. Output filters and filter specifications are discussed in the `Line Printer Spooler Manual`.

When you have completed these 5 steps, you can test the interface by printing a random file:

   ```
   % lpr random
   ```

Wait a minute or two; if you get no response, consult the next section.

### 6.6.2. Line Printer Troubleshooting

If you have followed the steps in the previous section and the `lpr` command doesn't work, consider the following:

1. Run the printer's stand-alone diagnostics. If these don't run, consult the operation manual for the printer or contact the printer manufacturer.

2. Verify that the printer switch settings are correct. Check for the correct settings in the printer operation manual.
3. Try setting the terminal characteristics using `stty(1)` followed by a `cat(1)` of a file to `/dev/ttyx`, where x is the serial port the printer is attached to. For example:

```
% (stty 1200; cat /etc/passwd) > /dev/ttyx
```

If the printer is "dead", your RS-232 cable may be bad, or you may need a null modem cable. See the section entitled *Asynchronous Serial Ports* in the Hardware Configuration and Expansion chapter for wiring information on RS-232 and null modem cables.

If the printer is responding but output is garbled, the terminal characteristics may be set incorrectly. Again, look at the output filter code to see what is done there.

4. If the `cat(1)` experiment above works, but `lpr` doesn't, it may be that a file is not found, or has incorrect permissions, etc. Check the characteristics of the `lpr`-related files according to the instructions above. Make sure that your output filter (if you only have one) is executable and that it is where you say it is. Check other `/etc/printcap` entries to make sure they are correct.

5. If you are getting error messages, see the *Line Printer Spooler Manual* for interpretation.

### 6.7. Adding Users

New users can be added to the system by adding a line to the password file `/etc/passwd`. The procedure for adding a new user is described in `adduser(8)`.

You should add accounts for the initial user community, giving each a directory and a password, and putting users who will wish to share software in the same groups.

### 6.8. Bootstrap and Shutdown Procedures

In a normal reboot, the system checks the disks and comes up multi-user without intervention at the console. Such a reboot can be stopped (after it prints the date) with a "C (interrupt). This will leave the system in single-user mode, with only the console terminal active.

If booting from the console command level is needed, then the command:

```
> b
```

boots the system from the default device.

You can boot to single user mode with:

```
> b -s
```

For a more complete discussion of booting — for example, how to boot from specific devices — see the appendix to this manual: *Power-up and Bootstrap.*

To take the system down to a single user state you must be super-user. Then you kill all active processes with:

```
# kill 1
```

or use the `/etc/shutdown(8)` command (which is much more polite, if there are other users logged in) when you are up multi-user. Either command kills all processes and gives you a shell on the console, as if you had just booted single-user. Previously mounted file systems remain mounted after the system is taken single-user. If you wish to come up multi-user again, you have to change directory to root, unmount all mounted file systems, then log out of single-user state:
# cd /
# /etc/umount -a
# *D  [Brings the system up multi-user]

Each system shutdown, crash, processor halt and reboot is recorded in the file /usr/adm/shutdownlog with the cause.

### 6.9. Device Errors and Diagnostics

When errors occur on peripherals or in the system, the system prints a warning diagnostic on the console. These messages are collected regularly and written into a system error log file /usr/adm/messages by dmesg(8), which is started by cron(8).

Error messages printed by the devices in the system are described with the drivers for the devices in the Section 4 pages of the *System Interface Manual*. If errors occur indicating hardware problems, you should contact your hardware support group or field service. It is a good idea to examine and truncate the error log file regularly — see the section below on *Files which Need Periodic Attention*.

### 6.10. File System Checks, Backups, and Disaster Recovery

Periodically (say every week or so in the absence of any problems) and always (usually automatically) after a crash, all the file systems should be checked for consistency by /etc/fck (see fck(1)). The procedures of /etc/reboot (see reboot(8)) should be used to get the system to a state where a file system check can be performed manually or automatically.

Dumping of the file systems should be done regularly, since once the system is going it is easy to become complacent. Complete and incremental dumps are easily done with /etc/dump (see dump(8)). Most people do a level-0 dump on a weekly or monthly basis, and a level-9 dump daily.

Dumping files by name is best done by tar(1), but the amount of data that can be moved in this way is limited to a single tape.

It is desirable that full dumps of the root file system be made regularly. This is especially true when only one disk is available. Then, if the root file system is damaged by a hardware or software failure, you can rebuild a workable disk by doing a restore in the same way that the initial root file system was created.

In the normal nature of things, disk space gets used up until your /usr file system is full. At this point, there are several places to look for files you can remove:

- Every time a crash occurs, several MBytes of system image get dumped into the /usr/crash directory. You can clean the um* files out if you don't need them for debugging or trouble reporting. Don't remove the /usr/crash/bounds file.
- Files in /usr/adm grow and must be pruned regularly.
- Check your accumulated mail file(s) — some diligent mail writers have been known to generate about a quarter of a MByte of mail per user per month.
- Network news accumulates at a fearsome rate.
- Finally, police your own directories regularly.

Useful mechanisms for monitoring disk usage are du(1) and df(1).
6.11. Moving Filesystem Data

If you have the equipment, the best way to move a filesystem is to dump it to magnetic tape using 
/etc/dump (see dump(8)), use /etc/newfs (see newfs(8)) to create the new filesystem, and 
restore the tape, using /etc/restore (see restore(8)). If for some reason you don't want to use 
tape, dump accepts an argument telling where to put the dump; you might use another disk. 
The restore program uses an "in-place" algorithm which allows filesystem dumps to be restored 
without concern for the original size of the filesystem. Further, portions of a filesystem may 
be selectively restored in a manner similar to the tape archive program. If you have to merge 
one filesystem into another, the best bet is to use tar(1). If you must shrink a filesystem, the 
best bet is to dump the original and restore it onto the new filesystem. 

Note that a level zero dump must be done after a full restore, because the restore relocates the 
files, although it does not change their content. Thus, you must do another level zero dump to 
get a tape image reflecting the new file positions, so that later incremental dumps will be 
correct.

6.12. Monitoring System Performance

The vmstat program provided with the system is designed to be an aid to monitoring system­
wide activity — see vmstat(8). Together with the ps(1) command (as in "ps av"), it can be used 
to investigate systemwide virtual memory activity. By running vmstat(8) when the system is 
active you can judge the system activity in several dimensions: job distribution, virtual memory 
load, paging and swapping activity, disk and cpu utilization. Ideally, there should be few 
blocked (b) jobs, there should be little paging or swapping activity, there should be available 
bandwidth on the disk devices (most single arms peak out at 30-35 tps in practice), and the user 
cpu utilization (us) should be high (above 60%).

If the system is busy, then the count of active jobs may be large, and several of these jobs may 
often be blocked (b). If the virtual memory is active, then the paging daemon will be running 
(s will be non-zero). It is healthy for the paging demon to free pages when the virtual memory 
gets active; it is triggered by the amount of free memory dropping below a threshold and 
increases its pace as free memory goes to zero.

If you run vmstat(8) when the system is busy (a "vmstat 1" gives all the numbers computed by 
the system), you can find imbalances by noting abnormal job distributions. If many processes 
are blocked (b), then the disk subsystem is overloaded or imbalanced. If you have several non-
DMA devices or open teletype lines that are "ringing", or user programs that are doing high­
speed non-buffered input/output, then the system time may go high (60-70% or higher). It is 
often possible to pin down the cause of high system time by looking to see if there is excessive 
context switching (cs), interrupt activity (in) or system call activity (sy).

If the system is heavily loaded, or if you have little memory for your load, then the system may 
be forced to swap. This is likely to be accompanied by a noticeable reduction in system performance 
and pregnant pauses when interactive jobs such as editors or window system programs 
swap out. If you expect to be in a memory-poor environment for an extended period you might 
consider administratively limiting system load.


6.13. Making Local Modifications

Locally written commands are kept in `/usr/src/local` and their binaries are kept in `/usr/local`. This allows `/usr/bin`, `/usr/ucb`, and `/bin` to correspond to the distribution tape (and to the system manuals). People wishing to use `/usr/local` commands should be made aware that they aren't in the base manual.

A `/usr/junk` directory to throw garbage into, as well as binary directories `/usr/old` and `/usr/new` are useful. The `man` command supports manual directories such as `/usr/man/manj` for junk and `/usr/man/manl` for local manual page entries to make this or something similar practical.


Optionally, UNIX records two kinds of accounting information: connect-time accounting and process-resource accounting. Connect-time accounting information is stored in the `/usr/adm/wtmp` file, which is summarized by the program `/etc/ac` (see `ac(8)`). Process-time accounting information is stored in `/usr/adm/acct`, and analyzed and summarized by the program `/etc/a` (see `a(8)`).

If you need to charge for computing time, you can implement procedures based on the information provided by these commands. A convenient way to do this is to give commands to the clock daemon `/etc/cron` to be executed every day at a specified time. This is done by adding lines to `/usr/lib/crontab`; see `cron(8)` for details.

6.15. Resource Control

Resource control in the current version of UNIX is rather primitive. The resources consumed by any single process can be voluntarily limited by the mechanisms of `setrlimit(2)`. Disk space usage can be monitored by `du(1)`. No system-enforced procedure for controlling a user's disk space usage is implemented under the current system, although a modicum of control can be obtained by dividing user groups between different disk partitions.

6.16. Network Troubleshooting

If you have anything more than a trivial network configuration, from time to time you are bound to run into problems. Before blaming the software, first check your network connections. On networks such as the Ethernet a loose cable tap or misplaced transceiver cable can result in severely deteriorated service. The `netstat(8)` program may be of aid in tracking down hardware malfunctions. In particular, look at the `-i` and `-s` options on the manual page.

If you believe you have a faulty coaxial Ethernet cable, test the cable to make sure it is delivering 50 ohms: remove the terminator from the N-connector on the transceiver terminating the cable, and use an ohm meter to test resistance (use the pin 'inside' the N-connector for signal, and the housing as ground). If you are getting something other than 50 ohms, your cable may be damaged. Contact Sun Microsystems for assistance.

If you believe the routing daemon is malfunctioning, its actions — and even all the packets sent and received — may be printed out. To create a log file of routing daemon actions, just supply a file name when you start the daemon up, for example:

```bash
# /etc/routed /etc/routerlog
```
Whenever a route is added, deleted, or modified, a log of the action and a history of the previous packets sent and received will be printed in the log file. To force full packet tracing, the -t option may be specified when the daemon is started up. Beware though — on a busy network this will generate almost constant output.

### 6.17. Files which Need Periodic Attention

The following files require periodic attention or are system-specific:

- `/etc/fstab` how disk partitions are used
- `/etc/printcap` printer data base
- `/etc/remote` names and phone numbers of remote machines for `tip(1)`
- `/etc/group` group memberships
- `/etc/motd` message of the day, printed at login
- `/etc/passwd` password file; each account has a line
- `/etc/rc.local` local system restart script; runs reboot; starts daemons
- `/etc/hosts` host name data base
- `/etc/networks` network name data base
- `/etc/services` network services data base
- `/etc/hosts.equiv` hosts under same administrative control
- `/etc/security` restricted list of ttys where root can log in
- `/etc/tty` enables/disables ports
- `/etc/ttytype` terminal types connected to ports
- `/usr/lib/crontab` commands that are run periodically
- `/usr/lib/aliases` mail forwarding and distribution groups
- `/usr/adm/acct` raw process account data
- `/usr/adm/messages` system error log
- `/usr/adm/shutdownlog` log of system reboots
- `/usr/adm/utmp` login session accounting
Chapter 7

Upgrading System Software

This chapter outlines procedures for upgrading software to a new release level.

7.1. Step 1: What To Save

No matter what version of the system you may be running, you will have to rebuild your root and /usr file systems. The easiest way to do this is to save the important files on your existing system, perform a bootstrap as if you were installing a software release on a brand new machine, then merge the saved files into the new system. The following lists the standard set of files you will want to save (in addition to all user files) and indicates directories in which site-specific files should be present. This list will probably be augmented with non-standard files you have added to your system; be sure to do a tar of the directories /etc/, /lib/, and /usr/lib/ so that you don't miss anything the first time around.

Table 7-1: Standard List of Files to Save when Upgrading

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.profile</td>
<td>root sh startup script</td>
</tr>
<tr>
<td>.login</td>
<td>root csh startup script</td>
</tr>
<tr>
<td>.cshrc</td>
<td>root csh startup script</td>
</tr>
<tr>
<td>.rhosts</td>
<td>list of hosts/users trusted at the superuser level</td>
</tr>
<tr>
<td>/dev/MAKEDEV.local</td>
<td>for the LOCAL case for making devices</td>
</tr>
<tr>
<td>/etc/fstab</td>
<td>disk configuration data</td>
</tr>
<tr>
<td>/etc/gettytab</td>
<td>tty port speeds database</td>
</tr>
<tr>
<td>/etc/group</td>
<td>group database</td>
</tr>
<tr>
<td>/etc/hosts</td>
<td>hosts database</td>
</tr>
<tr>
<td>/etc/hosts.equiv</td>
<td>list of trusted hosts on your network</td>
</tr>
<tr>
<td>/etc/nd.local</td>
<td>network disk local initialization file</td>
</tr>
<tr>
<td>/etc/networks</td>
<td>list of internet networks</td>
</tr>
<tr>
<td>/etc/passwd</td>
<td>user data base</td>
</tr>
<tr>
<td>/etc/printcap</td>
<td>printer capability database</td>
</tr>
<tr>
<td>/etc/rc</td>
<td>system startup file</td>
</tr>
<tr>
<td>/etc/rc.local</td>
<td>for any local additions</td>
</tr>
<tr>
<td>/etc/remote</td>
<td>remote hosts description database</td>
</tr>
<tr>
<td>/etc/ttyS</td>
<td>terminal line configuration data</td>
</tr>
<tr>
<td>/etc/ttytype</td>
<td>terminal line to terminal type mapping data</td>
</tr>
<tr>
<td>/etc/termcap</td>
<td>for any local entries which may have been added</td>
</tr>
</tbody>
</table>
You can save your system and user files by running the following commands as "root". Substitute the correct device abbreviation for your tape controller for tape, and the correct unit number for #:

Table 7-2: UNIX Tape Device Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>mt</td>
<td>Nine-track magnetic tape</td>
</tr>
<tr>
<td>ar</td>
<td>Archive quarter-inch tape</td>
</tr>
</tbody>
</table>

For example, if you are using your first quarter-inch tape drive, use /dev/rar0. If your system doesn't have a tape drive, you'll have to use slightly different commands, also given below, which access a tape drive elsewhere on the network. We'll call the machine with the tape drive your "remote host". Thus, before you begin, you'll have to know the device abbreviation for the remote host's tape drive (abbreviations are the same as in table above), and the remote host's hostname.

1. First, tar off the system files.

   For a machine with a tape drive:

   ```
   # cd /
   # tar cf /dev/rtape# .??* dev/MAKEDEV.local etc lib /usr/include /usr/lib
   ```

   For a machine without a local tape drive, use the commands below. Substitute your remote host's hostname for remote_host. Use 126 for block_size on a quarter-inch tape, and 20 for a half-inch tape. Note that the command should be typed as a single line; it appears on two lines because of formatting constraints only:

   ```
   # cd /
   # tar cfb - block_size /dev/rtape# .??* dev/MAKEDEV.local etc lib /usr/include
   /usr/lib | rsh remote_host dd of=/dev/tape# obs=block_sizeb
   ```

   This makes a tape containing system files which you will need to set up your system after the upgrade.

2. NOW CHANGE TO ANOTHER BLANK TAPE, and run the following command. As described just above, substitute the correct device specification for tape#. Also, replace
"usera . . ." with the names of all users on your system. (You can double check by looking in /etc/passwd or by doing "ls usr").

For a machine with a tape drive:

```
# tar cf dev/rtape# usr/{spool,local,usera,usert,userc,userd...}
```

For a machine without a tape drive (note that the command should be typed as a single line; it appears on two lines because of formatting constraints only):

```
# tar cfb - block_size /dev/rtape# usr/{spool,local,usera, usert,userc, userd...} |
  rsh remote_host dd of=/dev/tape# obs=block_size
```

This makes a tape containing users' files.

You may want to use other options on the tar commands, such as b to specify a large blocking factor such as 128 on Archive quarter-inch tapes, or v to list each file as it is processed. See tar(1) for more information.

Once you have saved the appropriate files in a convenient format, the next step is to dump all your file systems to magnetic tape with /etc/dump (see dump(8)).

When you have completed your system dump, install the new release from the distribution tape as described in the chapter, Installing UNIX for the First Time (except that you may skip the disk-reformatting phase); or, if you are installing the release on a machine without a tape drive, in the chapter, Installing UNIX on Systems Without Tape Support. Then proceed with the section below.

### 7.2. Step 2: Merging

When your system is booting reliably and you have the root and /usr file systems fully installed, you will be ready to proceed to the next step in the conversion process: merging your old files into the new system.

1. Using the first tar tape you created in Step 1, extract the appropriate files into a scratch directory, say /usr/convert.

   For a machine with a tape drive:

   ```
   # mkdir /usr/convert
   # cd /usr/convert
   # tar xf dev/rtape#
   ```

   For a machine without a tape drive:

   ```
   # mkdir /usr/convert
   # cd /usr/convert
   # rsh remote_host dd if=/dev/tape# bs=block_size | tar xBf block_size -
   ```

2. Certain files, such as those from the /etc directory, may simply be moved back into place:

   ```
   # cp passwd group fstab ttys ttytype /etc
   # cp crontab /usr/lib
   ```

   Remember to remove the old copies after you're sure your system completeness/consistency is confirmed. If you wish, you can mv the files instead of copying them.

   Other files must be merged into the distributed versions by hand (diff(1) is often useful here). In particular, be careful with /etc/termcap.
3. The spooling and local directories, and the user files saved on the second tar tape may be restored in their eventual resting places without too much concern. Be sure to use the p option to tar so that files are recreated with the same file modes (you may want to add other options, as described in the previous section):

   For a machine with a drive:
   
   ```
   # cd /
   # tar xfp dev/rtape#
   ```

   For a machine without a drive:
   
   ```
   # cd /
   # rsh remote_host dd if=/dev/tape# bs=block_sizeb | tar xBfp block_size -
   ```

4. Whatever else is left is likely to be site-specific or require careful scrutiny before you place it in its eventual resting place. Refer to the documentation (`hier(7)`, for example) before arbitrarily overwriting a file.
Chapter 8

Hardware Configuration and Expansion

This chapter describes the seven-slot Multibus card cage in the Model 100U and the fifteen-slot Multibus card cage in the 150U, and gives brief configuration details of the boards which may be supplied with them. For specific details of the configuration and descriptions of the boards and their jumpers, see the board-level manuals for individual boards.

In general, the Sun Workstation is shipped with all its boards (tape and disk controllers and such) already installed. However, if you are upgrading a Sun to include more peripherals, you can use this section to discover how to install the boards and how to set the switches for them. This chapter can also be used during troubleshooting, if you need to poke inside the card cage and identify the boards.

8.1. The 100U Card Cage

The card cage holding the Model 100U circuit boards lives in the black cabinet below the monitor housing.

Turn off the power and disconnect the power cord before opening up the Sun Workstation's enclosure.

To get to the Model 100U card cage, first unscrew the six screws on the sides of the black cabinet.

Stand at the back of the Sun Workstation where all the cable connectors are. Gently slide out the enclosure. There are many cables and wires connecting the enclosure to various parts of the workstation. You must pull the enclosure out gently to avoid breaking anything. Eventually, you won't be able to pull it any further because the wires are stretched as far as they can go.

As you look at the enclosure from the back of the workstation, the power supply is in the metal case on the left of the enclosure, and the Multibus card cage is on the right of the enclosure. The top of the card cage itself is uppermost, and the top of the boards is to the right.

The following diagrams show a few typical configurations for the desktop workstation. In the diagrams, the notation 'P2' indicates slots spanned by the P2 bus connector. Cards are shown in the slots where they are installed at the factory. Detailed explanations of the reasons for placing certain cards in specific slots follow this section.
### Basic Standalone Workstation

**Minimum — with Disk and 1/4” Tape Subsystem**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/4&quot; Tape Controller</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sun-2 CPU</td>
</tr>
<tr>
<td>4</td>
<td>1 MByte Main Memory</td>
</tr>
<tr>
<td>5</td>
<td>One-board SMD Disk Controller</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Monochrome Display Controller</td>
</tr>
</tbody>
</table>

### Expansion Option for the Standalone Workstation

**1 MByte Additional Memory**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/4&quot; Tape Controller</td>
</tr>
<tr>
<td>2</td>
<td>1 MByte Memory Expansion</td>
</tr>
<tr>
<td>3</td>
<td>Sun-2 CPU</td>
</tr>
<tr>
<td>4</td>
<td>1 MByte Main Memory</td>
</tr>
<tr>
<td>5</td>
<td>One-board SMD Disk Controller</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Monochrome Display Controller</td>
</tr>
</tbody>
</table>

### Network Node

**with Local Disk and Tape Storage**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ethernet Controller</td>
</tr>
<tr>
<td>2</td>
<td>1 MByte Memory Expansion</td>
</tr>
<tr>
<td>3</td>
<td>Sun-2 CPU</td>
</tr>
<tr>
<td>4</td>
<td>1 MByte Main Memory</td>
</tr>
<tr>
<td>5</td>
<td>One-board SMD Disk Controller</td>
</tr>
<tr>
<td>6</td>
<td>1/4&quot; Tape Controller</td>
</tr>
<tr>
<td>7</td>
<td>Monochrome Display Controller</td>
</tr>
</tbody>
</table>
**Basic Diskless Network Node**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ethernet Controller</td>
</tr>
<tr>
<td>2</td>
<td>1 MByte Memory Expansion P2</td>
</tr>
<tr>
<td>3</td>
<td>Sun-2 CPU P2</td>
</tr>
<tr>
<td>4</td>
<td>1 MByte Main Memory P2</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Monochrome Display Controller</td>
</tr>
</tbody>
</table>

**Diskless Node with Color Option**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ethernet Controller</td>
</tr>
<tr>
<td>2</td>
<td>1 MByte Memory Expansion P2</td>
</tr>
<tr>
<td>3</td>
<td>Sun-2 CPU P2</td>
</tr>
<tr>
<td>4</td>
<td>1 MByte Main Memory P2</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Color Display Controller</td>
</tr>
<tr>
<td>7</td>
<td>Monochrome Display Controller</td>
</tr>
</tbody>
</table>

Note that the configuration for a minimum standalone workstation (with 1/4” tape and disk) normally takes five slots, leaving two for expansion options. One such option is a second MByte of main memory. Note that due to power and operating temperature considerations, a color option may not be added to a desktop workstation with local disk storage. Further, since a minimum standalone workstation requires 25 amps (and its peak current rating is 35 amps), an Ethernet, or a second MByte of memory, or a customer’s own Multibus board may be added to such a system but Sun has only tested those configurations shown above. Three principal constraints (slots, power, and cooling) affect the viability of a proposed configuration.

Also, please note that these diagrams illustrate our current understanding of the best board configurations; check with Sun Microsystems or refer to the most recent configuration guides released by Sun if you intend to change your card cage layout in any way.

### 8.1.1. 100U Multibus Mastering Scheme

The Model 100U uses a Multibus backplane employing serial bus arbitration. This places a limit on the number of Multibus master cards (cards which use direct memory access — DMA). The limit today, as recommended by Intel and verified for our systems by Sun, is three Multibus masters. Multibus arbitration logic requires approximately 30 nanoseconds per card; the 10 MHz arbitration clock imposes the 3 card limit. Of the cards sold by Sun Microsystems, the CPU, the SMD disk controllers, and the 1/2-inch (but not the 1/4-inch) tape controller board are Multibus master cards. A system with all three of these cards may not operate reliably if
you add any additional DMA cards to the system. Note that non-DMA cards, such as Sun display controllers, Sun Ethernet controllers, and many other Multibus cards, are not Multibus masters and do not come under this restriction.

In the Multibus mastering scheme, the order of the cards in the card cage determines their priority. Slot number 1 is the highest priority bus master, and slot number 7 is the lowest. Bus master devices in the 100U card cage must be in *electrically* adjacent slots — this means that they must either be physically adjacent, or separated by a board which passes the bus arbitration signal straight through. Currently, the only Sun boards which are ‘transparent’ in this sense are the 1-MByte Memory board and Rev. C or ‘greater’ 1/4” Tape Controller boards. These boards may therefore be inserted between bus masters.

### 8.1.2. 100U Multibus P2 Connector

The P2 bus is a high-speed synchronous memory bus that allows memory access without wait states. The design of the connector for the P2 bus requires that the processor board and the main memory boards be inserted into adjacent slots. The Model 100U card cage has a P2 connector across slots 2, 3, and 4. The processor board is usually inserted in slot 3, with main memory cards in slots 4 and — possibly — 2.

Sun Workstation cards use the standard Multibus backplane signals, as defined in IEEE standard 796 for the P1 bus. Various Sun cards use the P2 bus in different ways, which may not be compatible with other vendors' use of this bus. In particular, Sun processor and Sun main memory cards do not support extended (24-bit) Multibus addressing on the P2 bus. They use the P2 bus for no-wait-state access to main memory. As a general rule, Sun boards and other vendors' boards which use the P2 bus should therefore not be installed on the same P2 section.

The signalling conventions used by the various Sun cards on the P2 bus are documented in the hardware reference manual for each board. For further details, please refer to the appropriate reference manual.

### 8.1.3. 100U Card Insertion Order

There are a few restrictions on the placement of cards in the card cage:

- As stated above, the design of the P2 connector requires that the processor board and memory boards be inserted in adjacent slots.
- With serial arbitration, bus masters (DMA devices) must be in *electrically* adjacent slots; boards which pass the bus arbitration signal through (currently, memory boards and Rev. C or greater 1/4” tape controller boards) may be inserted between bus masters.
- In a desktop workstation which includes a disk controller, the disk controller must be below the processor for proper handling of the serial bus priority. If present, the 1/2-inch tape controller, a third bus master, must be inserted above the disk controller, because the very high throughput of the disk controller might otherwise lock out the lower-priority device for excessive periods.
**Standalone Workstation with Disk and 1/2" Tape Subsystem**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>P2</td>
</tr>
<tr>
<td>3</td>
<td>Sun-2 CPU P2</td>
</tr>
<tr>
<td>4</td>
<td>1 MByte Main Memory P2</td>
</tr>
<tr>
<td>5</td>
<td>1/2&quot; Tape Controller</td>
</tr>
<tr>
<td>6</td>
<td>Xylogics 450 SMD Disk Controller</td>
</tr>
<tr>
<td>7</td>
<td>Monochrome Display Controller</td>
</tr>
</tbody>
</table>

For a customer's own DMA devices, the correct placement order must be determined empirically.

- The order in which cards are inserted into the Multibus card cage affects the ability of a desktop workstation to maintain proper operating temperatures. Cards that require high current also dissipate large amounts of heat and should be placed in slots in the card cage that receive the maximum air flow. The top and bottom slots in the card cage receive the least amount of cooling. Air flow is greatest across the center three slots in the system. The color display controller consumes the most power, and is best placed toward the center of the card cage. (In fact, if there are unused slots in the cage, one should be left open directly above the color controller board.)

### 8.2. The 150U Card Cage

Open the door of the rackmount system card cage enclosure: the card cage is the vertically mounted unit with fifteen slots. The following diagram shows a typical configuration of the rack-mountable workstation. In the diagram, the notations 'P2a', 'P2b', and 'P2c' indicate slots spanned by a single connector on the P2 bus. Cards are shown in the slots where they are installed at the factory. Detailed explanations of the reasons for placing certain cards in specific card slots follow this section.

Please note that this diagram illustrates our current understanding of the best board configuration; check with Sun Microsystems or refer to the most recent *Configuration Guide* released by Sun if you intend to change your card cage layout in any way.
Model 150U Card Cage

<table>
<thead>
<tr>
<th>Slot</th>
<th>Description</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sun-2 CPU</td>
<td>P2a</td>
</tr>
<tr>
<td>2</td>
<td>1 MByte Main Memory</td>
<td>P2a</td>
</tr>
<tr>
<td>3</td>
<td>1 MByte Memory Expansion</td>
<td>P2a</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>P2a</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>P2a</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>P2a</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>P2a</td>
</tr>
<tr>
<td>8</td>
<td>Ethernet Controller</td>
<td>P2b</td>
</tr>
<tr>
<td>9</td>
<td>Half-Inch Tape Controller</td>
<td>P2b</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>P2b</td>
</tr>
<tr>
<td>11</td>
<td>Xylogics 450 SMD Disk Controller</td>
<td>P2b</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>P2b</td>
</tr>
<tr>
<td>13</td>
<td>Color Display Controller</td>
<td>P2c</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>P2c</td>
</tr>
<tr>
<td>15</td>
<td>Monochrome Display Controller</td>
<td>P2c</td>
</tr>
</tbody>
</table>

Note: the Sun-1 Model 150U card cage is mounted vertically, with slot 1 at the left.

8.2.1. 150U Multibus Mastering Scheme

The Model 150U card cage uses parallel priority bus arbitration. In these systems, there is no limitation on the number of bus master cards (DMA devices) which may be inserted in the card cage. Of the cards sold by Sun, the CPU, the SMD disk controllers, and the 1/2" tape controller are masters. The relative priority of each master is determined only by its relative slot number; slot 1 is the highest priority. In a parallel card cage, bus master cards need not be physically adjacent.

8.2.2. 150U Multibus P2 Connectors

The Model 150U card cage has three separate P2 connectors. Slots 1 through 7 share the first P2 connector; slots 8 through 12 share the second; and slots 13 through 15 share the third. Sun suggests placing the processor card and main memory cards on the first P2 connector, and the color graphics controller on the third connector. Disk, tape, Ethernet, and any non-Sun controllers should be installed on the second P2 connector.

Sun Workstation cards use the standard Multibus backplane signals, as defined in IEEE standard 796 for the P1 bus. Sun cards use the P2 bus in different ways, which may not be compatible with other vendors' use of this bus. For example, the Sun processor and Sun main memory cards do not support extended (24-bit) Multibus addressing on the P2 bus. They use the P2 bus for no-wait state access to main memory. As a general rule, Sun boards and other
vendors' boards which use the P2 bus should therefore be installed in slots with separate P2 backplane connectors.

The signalling conventions used by the various Sun cards on the P2 bus are documented in the hardware reference manual for each board. For further details, please refer to the appropriate reference manual.

8.2.3. 150U Card Insertion Order

There are a few restrictions on the placement of cards in the card cage:

- The design of the connector for the P2 bus requires that the processor board and the main memory boards be inserted into adjacent slots.
- The 3COM Ethernet Board must not share a P2 bus with any other board which uses signals from the P2 bus.
- Because of its high power dissipation, the Xylogics 450 SMD Disk Controller Board should be placed in the most central position available in the backplane, as air flow is greatest across the center of the card cage. However, the Xylogics board must not share a P2 bus with any Sun Microsystems board. For maximum air circulation, leave the slot to the left of the Xylogics controller empty if possible.
- The Sky Floating Point Board must not share a P2 backplane with any Sun Microsystems board (it uses the P2 for power and ground).
- A single Color Display Controller Board must not share a P2 backplane with the Sun-2 CPU and Memory Boards. If possible, it should be placed toward the middle of the card cage to take advantage of the greater air circulation. If there are unused slots in the cage, leave one open to the immediate left of the Color Controller Board.
- The 1/2-inch Tape Controller Board must not share a P2 backplane with the Sun-2 CPU or Memory boards. If present in a configuration which also has a Xylogics SMD disk controller, the 1/2-inch tape controller should be inserted 'above' the disk controller, because the very high throughput of the disk controller might otherwise lock out the lower-priority device for excessive periods.

8.3. Sun-2 CPU

The Sun-2 CPU (Central Processor Board) of the Sun Workstation is usually close to the top of the Multibus card cage — usually in the first slot of the 150U card cage, and the third slot of the 100U card cage. The CPU board has two fifty-pin cable connectors coming off from the top of the board. One of the cables splits in two and goes to the connectors labelled “RS232 A” and “RS232 B”. See the subsection, Asynchronous Serial Ports for wiring information. The other cable also splits in two and goes to the connectors labelled “Keyboard” and “Mouse”.

The jumpers on the CPU Board should be wired as follows:

- J700 Bus Priority In (BPRN/) Should be installed in the Model 100U only. See the subsection below, CPU Multibus Priority, for more information.
- J701 Common Bus Request (CBRQ/) (not installed) See the subsection below, CPU Multibus Priority, for more information.
- J702 Bus Clock (BCLK) (installed)
Hardware Configuration and Expansion

• J703 Constant Clock (CCLK) (installed)
• J801 Mouse VCC (installed)
• J400 27128 EPROM’s (installed)
• J401 27256 EPROM’s (not installed)

Only one of J400 and J401 must be installed at a time.

8.3.1. CPU Multibus Priority

As shipped, the Sun central processor board (CPU) is always configured as the highest-priority Multibus master by placing it in a higher priority slot than all other bus masters in the card cage.

In a Model 100U, because of its serial card cage, configuring the CPU as the highest-priority master also involves installing a jumper at location J700 (along the bottom of the processor board) to ground Bus Priority In (BPRN). The processor board can also be run at a lower bus priority by removing jumper J700, and thus allowing the processor to receive its BPRN signal from the arbitration circuit on the master with higher priority.

In the Model 150U, the CPU is configured as the highest priority master by simply placing the CPU in the highest priority slot in the parallel card cage.

For both models, if the CPU board is used in conjunction with a Multibus DMA board, such as a disk controller, that does not support Common Bus Request (CBRQ), the CPU board must be configured such that it gives up the Multibus after every Multibus cycle. This is done by jumpering location J701, which is the second jumper from the left along the bottom of the board. This also will cause three additional wait states for each Multibus access. The Interphase 2180 is an example of a Multibus DMA board requiring this configuration. No other Sun-supplied boards require this configuration if they are properly configured. See the subsection on each board to determine how to do this. If you have boards from other vendors, check this carefully.

On the other hand, if all Multibus DMA devices (bus masters) do support CBRQ (or if there are no Multibus DMA devices), the CBRQ jumper on the processor board is not required. Instead, the CPU board will retain bus mastership until a lower priority master requests it by asserting CBRQ. Following a CBRQ, the processor board will yield mastership for at least one cycle. For certain machine configurations (especially those with color), significant speed enhancements result from removing this jumper.

8.3.2. Asynchronous Serial Ports

The Sun Models 100U and 150U provide two asynchronous serial ports. The ports are controlled by the SCC (Serial Communications Controller) on the Sun-2 CPU board. Both ports have RS-232-C cabling conventions but use RS-423 signalling. The two serial port connectors are labelled “RS-232 A” and “RS-232 B” on the card cage enclosure back panel; these ports correspond to /dev/ttya and /dev/ttyb in the system software.
Note that older workstations (pre-1.0 Release) have only two serial ports — serial port A is configured as a DCE (Data Communications Equipment) and serial port B is configured as a DTE. In addition, only serial port A on the older workstations has full modem control. This means that external cables built for the Sun-1 or Sun-1.5 CPU either may have to be modified to work correctly with the Sun-2 CPU, or a null modem cable may be required.

The serial ports on the Sun Workstation were designed primarily for connecting output peripherals — such as terminals, modems, printers, and plotters — and can drive these output lines at speeds up to 19.2 Kbaud. All ports provide the CTS, RTS, DTR, DSR, and DCD control lines required by some devices (such as modems) in addition to the transmit and receive lines; the ports generate DTR and RTS, and pay attention to DSR, CTS, and DCD. All ports are wired as DTE (data terminal equipment) ports, and thus permit direct connection of modems and the like (25 pins straight through). Computers and other DTE devices can be connected by using the null modem cable supplied with your Sun system, or any similar null modem cable. The null modem cable provided by Sun has pins 2 and 3 crossed, 4 and 5 crossed, and 6 and 20 crossed; pins 1 (Frame Ground) and 7 (Signal Ground) are carried through.

On the CPU Board, I/O channels A and B appear on 50-pin connector J1. The pin assignments for J1 are:

Table 8-1: Pin Assignments for J1 on the CPU Board

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>TxD A</td>
<td>28</td>
<td>TxD B</td>
</tr>
<tr>
<td>4</td>
<td>DB A</td>
<td>29</td>
<td>DB B</td>
</tr>
<tr>
<td>5</td>
<td>RxD A</td>
<td>30</td>
<td>RxD B</td>
</tr>
<tr>
<td>7</td>
<td>RTS A</td>
<td>32</td>
<td>RTS B</td>
</tr>
<tr>
<td>8</td>
<td>DD A</td>
<td>33</td>
<td>DD B</td>
</tr>
<tr>
<td>9</td>
<td>CTS A</td>
<td>34</td>
<td>CTS B</td>
</tr>
<tr>
<td>11</td>
<td>DSR A</td>
<td>36</td>
<td>DSR B</td>
</tr>
<tr>
<td>13</td>
<td>GND A</td>
<td>38</td>
<td>GND B</td>
</tr>
<tr>
<td>14</td>
<td>DTR A</td>
<td>39</td>
<td>DTR B</td>
</tr>
<tr>
<td>15</td>
<td>DCD A</td>
<td>40</td>
<td>DCD B</td>
</tr>
<tr>
<td>22</td>
<td>DA A</td>
<td>47</td>
<td>DA B</td>
</tr>
<tr>
<td>24</td>
<td>BSY A</td>
<td>49</td>
<td>BSY B</td>
</tr>
</tbody>
</table>

Connector pins not mentioned are not connected to anything.

The flat cables that connect to J1 on your CPU board convert this pinout to standard RS-232 pin assignments. The channels are implemented with three Zilog Z8530 dual UART chips. See the Zilog Serial Communications Controller Technical Manual (available from Sun Microsystems, Part Number: 800-1052-01) for programming information.

Getting the cabling right is a problem that should be familiar to anyone who has had to connect RS-232 equipment; sometimes it is most easily solved by experimenting. A piece of equipment known as an EIA Interface Adapter and Monitor — or, colloquially, a 'breakout box' — is invaluable in these exercises. Consult a hardware technician or Sun Microsystems if you need
help.
Most devices (other than modems which are answering phones) do not assert carrier properly. There is therefore a software facility to simulate asserting carrier: see \texttt{rs(4S)} and \texttt{oct(4S)}.

\section*{8.4. Sun-2 Memory Board}

The Sun-2 Memory Board(s) must share the P2 bus with the Sun-2 CPU, and must be physically adjacent to the CPU Board due to the design of the P2 connector. They are thus usually placed in the second and/or third slots of the Model 150U card cage, and in the second and/or the fourth slots of the Model 100U card cage. In the Model 100U, Sun-2 Memory Boards may be placed between the CPU and other bus masters, as bus priority is passed through the Memory Board. This is not an issue for the 150U, as its card cage uses parallel bus arbitration.

Sun-2 Memory Boards are connected to the Processor Board via the Multibus P2 connector, and permit access by the 10MHz MC68010 without wait states. The boards use the Multibus P1 connector only for +5 volts and ground connections.

The first Memory Board provides 1 MByte of main memory for the Sun-2 CPU; each additional board provides another MByte of memory. Since the Models 100U and 150U support up to 2 MBytes total memory, one board may be added to the basic system.

To configure a Memory Board, you need only select its starting memory location at 0 or 1 MByte within the on-board memory address space. Do this by setting the eight-position DIP switch located at position U506 on the Memory Board as follows:

- The \textit{first} Memory Board should have switch 1 on the DIP switch \texttt{ON} and switches 2 through 8 \texttt{OFF}.
- The \textit{second} Memory Board should have switch 2 on the DIP switch \texttt{ON} and all others \texttt{OFF}.

\section*{8.5. Video Board}

The Sun graphics system is a high-resolution bit-mapped frame buffer and display processor on one Multibus board. This board — called the "video board" or "monochrome display controller" — lives in the bottom slot in the Multibus card cage. The video board has a 10-pin 'D shell' connector on the right hand end as you look at the top of the board.

The video board has one 8-section DIP switch, which is used to set the board's Multibus base address (in Multibus memory space). It may be set to any address multiple of 0x20000 between 0x0 and 0xE0000. The first video board in a Sun system, however, must be set to address 0xC0000; additional video boards are placed at successively lower addresses.

The following table indicates which switch to set for each of the possible addresses. Note that only the indicated switch section should be set to \texttt{ON}; all other sections should be in the \texttt{OFF} position.

\begin{tabular}{|c|c|c|}
\hline
Address & Switch 1 & Switch 2 to 8 \texttt{OFF} \\hline
0x00000 & \texttt{ON} & \texttt{OFF} \\hline
0x20000 & \texttt{OFF} & \texttt{ON} \\hline
0x40000 & \texttt{OFF} & \texttt{OFF} \\hline
0xC0000 & \texttt{OFF} & \texttt{ON} \\hline
\end{tabular}
Table 8-2: Video Board Address Selection

<table>
<thead>
<tr>
<th>Address</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000</td>
<td>8</td>
</tr>
<tr>
<td>0x20000</td>
<td>7</td>
</tr>
<tr>
<td>0x40000</td>
<td>6</td>
</tr>
<tr>
<td>0x60000</td>
<td>5</td>
</tr>
<tr>
<td>0x80000</td>
<td>4</td>
</tr>
<tr>
<td>0xA0000</td>
<td>3</td>
</tr>
<tr>
<td>0xC0000</td>
<td>2*</td>
</tr>
<tr>
<td>0xE0000</td>
<td>1</td>
</tr>
</tbody>
</table>

* Setting for First Video Board

8.6. Xylogics 450 Disk Controller

The Xylogics 450 SMD disk controller board has five cable connectors on its top: four 26-pin connectors and one 60-pin connector. In the Model 100U card cage, the disk controller board must be below the CPU for proper handling of bus arbitration; if there is a 1/2" tape controller board in the card cage, it should be placed in a higher priority position than the disk controller. In the Model 150U card cage, the disk controller is placed on the second P2 bus; again, it should occupy a lower priority slot than the 1/2" tape controller.

The Xylogics 450 SMD is an intelligent storage module controller/formatter, using bipolar microprocessor technology. It plugs directly into the Multibus and is a bus master during data transfers, using a variable burst length DMA technique. It directly connects via industry-standard A and B cables to from one to four storage module drives which are available from a number of manufacturers. Sun Microsystems supports a family of such drives: the Fujitsu M2312K and M2284 Microdisk Drives, and the M2351 Mini-Disk Drive.

If you need to install the board yourself, the following placement considerations and configuration details apply:

- The Xylogics board should not share a P2 connector with the Sun-2 CPU or Memory Boards, as it has P2 traces which are incompatible with the CPU/Memory P2 bus.
- Because the Xylogics board is a Multibus master, its relative slot number determines its priority (slot 1 is the highest-priority master). The board must be placed in a lower-priority position than the Sun-2 CPU board for proper handling of bus arbitration. It should also be placed in a lower-priority position than the 1/2-inch Tape Controller Board, if there is one in the system.
- Since the Xylogics board dissipates a fair amount of heat, it should be placed in the most central position possible in the backplane (subject to the considerations listed above). For maximum air circulation, leave the slot to the left of the board (as installed) empty if possible.

Several sets of straps are provided for configuring the disk controller board. For proper operation in the Sun environment, the following options must be selected:
Hardware Configuration and Expansion

- Configure the Xylogics 450 for 16-bit I/O addressing at address 0xee40 by installing exactly the following jumpers on jumper group JA through JE: JA3-JB3, JA8-JB8, JR1-JC1, JC2-JD2, JC3-JD3, JC4-JD4, and JE4-JE5.

- Configure the 450 for 20-bit memory addressing. Jumpers are: JM1-JM2 Out; JM3-JM4 In.

- Select interrupt level 2: wire pin E2 to pin JX4 (second pin from left in top row of JX pins).

- Enable the BPRO signal: jumper JE1 to JE2.

- Verify that jumpers JH1-JH2 and JN1-JN2 are NOT installed and leave other jumpers as shipped from the factory.

The first controller is configured at address 0xee40 by setting the JA through JE jumper group as described above. If you are configuring a second board, it should start at address 0xee48: the jumper at JC4-JD4 should be out and one at JR4-JC4 should be in. The system can support a maximum of two Xylogics 450 controller boards.

Please note that if you have an Interphase 2180 disk controller board in your system, only one Xylogics board can be supported. The Xylogics board should be configured for address ee40; the Interphase should be configured for address 48 as described below.

For the full specification and detailed description of the disk controller, see the supplied document: *Xylogics Model 450 Peripheral Processor SMD Disk Subsystems Maintenance and Reference Manual.*

### 8.7. Interphase 2180 Disk Controller

The Interphase Disk Controller Board has five cable connectors on its top: four 26-pin connectors and one 60-pin connector. In the Model 100U card cage, the Interphase board must be below the CPU for proper handling of bus arbitration; if there is a 1/2" tape controller board in the card cage, it is placed above the disk controller. In the Model 150U card cage, the Interphase disk controller is placed on the second P2 bus; again, it should occupy a lower priority slot than the 1/2" tape controller.

> **Note** that the Interphase controller does not generate CBRQ/; the CPU board must be configured to take this into account. On the CPU board, the jumper at location J701 must be installed. See the section, *CPU Multibus Priority* for more information.

The Interphase SMD 2180 is an intelligent storage module controller/formatter, using bipolar microprocessor technology. It plugs directly into the Multibus and is a Bus Master during data transfers, using a variable burst length DMA technique. It directly connects via industry standard A and B cables to from one to four storage module drives which are available from a number of manufacturers. Sun Microsystems supports two such drives: the Fujitsu M2312K ("D84") and M2284 ("D169") Microdisk Drives. Note that the Interphase 2180 is not suitable for use with the Fujitsu M2351 ("D474") Mini-Disk Drive, also supported by Sun.

Four sets of straps and two 8-bit DIP switches are provided for configuring the Interphase disk controller board. For the UNIX system on the Sun, the following options must be selected:

- Install the M3 jumper, selecting "increment by head" rather than "increment by cylinder".

- Switch #3 on DIP switch S2 must be ON, selecting level 2 interrupts.
• On DIP switch S1, set switch #8 ON, set #7 OFF, selecting 512 byte sectors, 34 sectors per track.
• Switches #1-6 on DIP switch S1 determine the address of the controller in Multibus I/O space. They supply the bits: 054321xx of the 8-bit Multibus I/O address. The board can be written to at four consecutive addresses, and read from at the lowest of the four. The Sun Workstation only uses addresses 0x40, 0x44, 0x48, or 0x4C. These addresses are obtained by setting the switches on DIP switch S1 as follows:

Table 8-3: Selecting Disk Controller Base Address

<table>
<thead>
<tr>
<th>Desired Address</th>
<th>Switch Settings</th>
<th>Switch 6</th>
<th>Switch 5</th>
<th>Switch 4</th>
<th>Switch 3</th>
<th>Switch 2</th>
<th>Switch 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>0x44</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>0x48</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>0x4C</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
</tbody>
</table>

Normally, the first Interphase controller in the system is configured at address 0x40 by setting switch #5 on DIP switch S1 ON and all others OFF. If you also have a Xylogics board in your system, configure the Interphase instead for address 0x48 by setting switch #2 ON and all others OFF.

• Finally, the Interphase controller board is normally configured to have lower Multibus priority than the CPU board by installing jumper E to F.

Sun recommends a maximum of one Interphase disk controller board per system.

For the full specification and detailed description of the disk controller, see the supplied document: SMD 2180 Storage Module Controller/Formatter User’s Guide.

8.8. Nine-Track Tape Controller

If your Sun system is shipped with a nine-track tape unit, there should be a tape controller for that subsystem installed in the card cage. The tape controller is a TAPEMASTER board from Computer Products Corporation. The TAPEMASTER nine-track tape controller board has two 50-pin cable connectors on its outside edge (viewed installed in the card cage).

As shipped, the TAPEMASTER should be configured correctly for your Model 100U or 150U. If you are installing the board, please verify the following:

• For a 100U, locations 1 to 2 are jumpered; they must be jumpered for serial backplanes.
• For a 150U, locations 1 to 2 are not jumpered; they should be jumpered for serial backplanes only.
• For both models, locations 3 to 4 and locations 52 to 53 must be jumpered for proper handling of CBRQ/; if these locations are jumpered, the tape controller will surrender the bus to a higher-priority master when that master activates CBRQ/. If you configure the board in any other fashion, you must also configure your Sun-2 CPU so that it gives up the Multibus after every cycle by installing the jumper at location J701 on the CPU board.
There are also board placement considerations involved in installation:

- The TAPEMASTER board should not be placed on the same P2 connector as the Sun-2 CPU and Memory Boards.
- The nine-track tape controller is a Multibus master. This means that the relative number of its slot determines its priority (slot 1 is the highest-priority master). The board must be placed in a lower-priority position than the CPU. If you have a Xylogics 450 SMD Controller Board in your system as well, place the TAPEMASTER Board in a higher-priority position; otherwise, the Xylogics board will lock out the TAPEMASTER (due to its higher data transfer rate).

To cable up the tape controller, connect the 50-pin ribbon cable coming from the backpanel connector labelled “DISK/TAPE COMMAND B” to the “J2” connector in the center of the TAPEMASTER board. The 50-pin ribbon cable from the back panel connector labelled “TAPE COMMAND A” should go to the “J1” edge connector of the TAPEMASTER board.

As shipped, the tape controller board should already be set up to work correctly in a Sun system. Refer to the Computer Products Corporation TAPEMASTER Manual for details of configuring the board if required.

### 8.9. Quarter-Inch Tape Controller

The Sun tape interface board consists of an interface to an Archive 1/4” streaming tape unit and 256 KBytes of Multibus memory with parity. The tape controller is not a bus master, and boards which are Rev. C or 'greater' may actually be placed in a slot between bus masters (in the Model 100U), as they pass bus priority through.

```markdown
Note that you must disable the Multibus memory on the Tape Controller Board if you have a Sun-2 CPU in your system. To do this, open all switches on switch package U50, as described below.
```

There are four 8-position DIP switches on the tape controller board. They are used to select the interrupt level at which the tape controller will interrupt the processor, select the base address for the Multibus memory on the board, and to select the base address for the Multibus I/O registers which communicate commands and status information between the tape controller and the central processor.

With the Multibus connectors facing you, the switches are all in the lower left hand corner of the board, at component locations U52, U50, U53, and U56. U52 is in the lower left-hand corner of the board; U50 is directly above U52; U53 is to the immediate right of U50; and U56 is to the immediate right of U53. For each package, switch #1 is on the right and switch #8 is on the left.

Switch package U52 selects the interrupt level; switch #1 selects interrupt level 0, switch #8 selects interrupt level 7. The desired switch should be closed (or ON); all others should be left open (or OFF). The table below shows the switch settings. The Sun system is configured to run with level 3 interrupts from the quarter-inch tape controller.

---

8-14 Revision H of 12 March 1984
Table 8-4: Selecting Quarter-Inch Tape Controller Interrupt Level

<table>
<thead>
<tr>
<th>Required Interrupt Level</th>
<th>Required Interrupt Level</th>
<th>Close Switch Number</th>
<th>Close Switch Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>4 *</td>
<td>8</td>
</tr>
</tbody>
</table>

* Note that this is the standard setting.

Switch package U50 selects the base address for the Multibus memory. This memory must be disabled for proper operation with the Sun-2 CPU: all switches on U50 should be opened. If you are not using the tape controller board with a Sun-2 CPU, and need to make use of the Multibus memory, see the next subsection for switch settings.

Switch packages U53 and U56 select the base address of the I/O registers that the tape controller uses in Multibus I/O address space. The tape controller uses eight consecutive bytes of Multibus I/O space, and this block of eight registers must be located on an eight-byte boundary. DIP switch packages U53 and U56 decode address lines A15 through A3. Switch #8 on U53 corresponds to A15; switch #1 on U53 corresponds to A8; switch #8 on U56 corresponds to A7; and switch #4 on U56 corresponds to A3. Address lines A0 through A2 are not decoded through the switches.

Switch #1 on U56 enables/disables the tape section on the board.

Switches #2 and #3 on U56 are not used and their settings are irrelevant.

The table below illustrates the switch settings in a diagrammatic form. A 0 corresponds to an open (or OFF) switch, while a 1 corresponds to a closed (or ON) switch.

Table 8-5: Selecting Quarter-Inch Tape Controller Base Address

<table>
<thead>
<tr>
<th>Required Address</th>
<th>Switch Settings</th>
<th>U53</th>
<th>U56</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF000</td>
<td>11110000</td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td>0x1238</td>
<td>00010010</td>
<td>00111000</td>
<td></td>
</tr>
<tr>
<td>0x200 *</td>
<td>00000010</td>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td>Disable Tape</td>
<td>xxxxxxxx</td>
<td>xxxxxxxx1</td>
<td></td>
</tr>
</tbody>
</table>

* Note that this is the standard setting.
8.9.1. Tape Controller Multibus Memory

This subsection is irrelevant to you if you are using your 1/4" tape controller board with a Sun-2 CPU; the tape controller Multibus memory is unnecessary with such a configuration, and must be disabled as described above.

If you need to make use of the 256 KBytes of Multibus memory on the tape controller board, select the base address for the memory as follows.

Memory on the tape controller board comes in 128K byte chunks; there are two of these chunks. The chunk(s) of memory may be located on a 128K byte boundary anywhere within a one Megabyte address space by closing appropriate switches on package U50.

With the Multibus connectors facing you, the four switch packages are in the lower left hand corner of the board, at component locations U52, U50, U53, and U56. U52 is in the lower left-hand corner of the board; U50 is directly above U52. For each package, switch #1 is on the right and switch #8 is on the left.

Two adjacent switches on U50 should be closed. This places the 256K bytes of memory on the 0, 128K, 256K, 384K, 512K, 640K, or 768K boundaries. All other switches on package U50 should be opened.

If all switches on U50 are opened, all memory accesses to the tape controller board are disabled. The table below illustrates the switch settings in detail.

**Table 8-6: Selecting Quarter-Inch Tape Controller Memory Base Address**

<table>
<thead>
<tr>
<th>For 256K Byte Tape Controller Boards</th>
<th>Selects Address Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closing Switch Numbers</td>
<td>Decimal</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>8 and 7</td>
<td>0</td>
</tr>
<tr>
<td>7 and 6</td>
<td>128K</td>
</tr>
<tr>
<td>6 and 5</td>
<td>256K</td>
</tr>
<tr>
<td>5 and 4</td>
<td>384K</td>
</tr>
<tr>
<td>4 and 3</td>
<td>512K</td>
</tr>
<tr>
<td>3 and 2</td>
<td>640K</td>
</tr>
<tr>
<td>2 and 1</td>
<td>768K</td>
</tr>
</tbody>
</table>

8.10. 3COM Ethernet Controller Board

A Sun-1 Workstation in a networked environment is currently supplied with a 3COM 3C400 Ethernet Controller Board. The 3COM Ethernet controller is a memory-mapped device in the Sun Workstation. There are several jumpers that should already be set correctly when the Controller leaves the factory. Correct settings of the jumpers and switches for the Sun environment are as follows:
- The Ethernet controller should be set for 20-bit memory addressing. Look at the board with the components up and the Multibus edge connectors facing you. At the lower left corner of the board there are two jumpers labelled JP1 and JP2. There should be a shorting jumper on JP2; there should not be any shorting jumper on JP1.

- Just to the right of the address width selection jumpers, there is a row of jumper pins. The pins marked MRDC and MWTC should have shorting jumpers installed. The pins marked IORC and IOWC should not have any jumpers installed.

- Then there is a row of jumpers marked INT7 at one end, down to INT0 at the other end; these select the interrupt level. For the Sun Workstation, select interrupt level 3 — install a shorting jumper on the fourth pair of jumper pins from the right-hand end of the row. Remove the jumpers from all other INT(n) pins.

- The switch marked ADR17/ at the bottom right-hand corner of the board should have all its keys set to OFF.

- The switch marked ADR19/ in the lower left-hand corner of the board selects the memory base address for the board. The Ethernet controller board consumes 8K bytes of Multibus memory space.

- The Sun Workstation expects to find the first Ethernet board's memory at address 0xE0000. This address is selected by setting keys 1, 2, and 3 on the switch marked ADR19/ to ON and the other switches to OFF.

- The Sun Workstation expects to find the second Ethernet board's memory at address 0xe2000. This address is selected by setting keys 1, 2, 3, and 7 on the switch marked ADR19/ to ON and the other switches to OFF.

- In all cases, keyswitch 8 on the switch marked ADR19/ should always be OFF.

- In component position 12 on the Ethernet controller board there is a PROM which contains the hardware Ethernet address for this board.

### 8.11. Sky Floating Point Board

In both the Model 100U and the Model 150U card cages, the Sky floating point processor may be placed in any available slot which does not share a P2 bus section with any Sun board which uses the P2 bus. The Sky board does IEEE format floating-point calculations much faster than they can be done in software, and so makes many compute-intensive programs run significantly faster.

To run in the Sun environment, the Sky board must be set to interrupt on level 2, and configured for Multibus I/O address 0x2000. If you purchased your board from Sky, you must set these parameters by rewiring pins at jumper block locations JP02 and JP01, as follows:

- Holding the board with the Multibus edge connectors facing down (the location numbers on the board will be right side up), locate JP02 and JP01: they are the two blocks with exposed pins near the lower left-hand corner of the board. JP02 is the leftmost block. In the following diagrams, we assume that the board is in this position.
To set the interrupt level to 2, replace the wire between pins 2 and 6 of JP02 by two wires: one between pins 1 and 6, and one between pins 3 and 6 (leave the shunt between pins 4 and 5 on):

Table 8-7: Selecting SKY Board Interrupt Level

Before: 

<table>
<thead>
<tr>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

After: 

<table>
<thead>
<tr>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

To set the address of the board to 2000 in Multibus I/O space, replace the shunt between pins 1 and 2 of JP01 with a wire between pins 1 and 11 of the same jumper block:

Table 8-8: Selecting SKY Board Base Address

Before: 

<table>
<thead>
<tr>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

After: 

<table>
<thead>
<tr>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

For more information, see the Installation Notes for SKY FFP document shipped with the board.
Appendix A

Power-up and Bootstrap

The central processor board (CPU) of the Sun Workstation has a set of ROMs which contain a program generally known as the “monitor”. The monitor controls the operation of the system before the UNIX kernel takes control.

The first, second, and third major sections in this appendix cover the startup and bootstrap functions of the monitor. Under normal circumstances, the monitor automatically bootstraps the UNIX system after initial power-on; no manual intervention is required. These sections describe how it does that, and how to “boot” manually when necessary.

The fourth section lists messages which the monitor and boot program can display. These should be useful for troubleshooting.

For information on the PROM Monitor commands, see the CPU PROM Monitor document in the Sun System Internals Manual.

A.1. Power-On Self Test Procedures

When system power is first turned on, the monitor runs a quick self-test procedure. The test can have one of these results:

- Critical errors are found. The screen remains dark. The error is reported on eight LEDs on the CPU board in the card cage.
- No video board is found. The monitor sends its output to serial port ‘A’ (labelled “RS-232 A” on Model 100U/150U backpanels, and “SIO-A” on Model 120/170 backpanels). Connect an ASCII terminal to this RS-232 connector. Configure the terminal for 9600 baud, no parity, one stop bit. Then power on the workstation again and look for messages on the terminal.
- Non-critical errors are found. These are reported to the screen, and the system begins the automatic boot process.
- No errors are found. This is reported to the screen, and the system begins the automatic boot process.

A.1.1. Critical Errors from Self Test

Severe problems are reported using the eight miniature LEDs on the CPU board. Depending on your workstation model, you get at these differently:
• With a Sun-1/100U, these LEDs can be glimpsed through the cooling slots on the left side of the monitor base. If your screen remains dark, and you see more than one LED on in the middle of some board in the card cage, you'll probably have to slide the drawer out of your Sun-1/100U to really see the error code in the LEDs.

• With a Sun-1/150U or Sun-2/170, you can see the LEDs by opening the front door of the card cage enclosure.

• With a Sun-2/120, pull off the front plastic panel of the pedestal; the row of 8 LEDs is on the front edge of a card in the middle of the card cage.

When power is first applied to the workstation, all eight LEDs light, then each lights quickly in sequence. Following this 'lamp test', the lights blink rapidly as each test is passed. The lights slow down as memory is tested; each of the two memory tests takes a few seconds per megabyte. Finally, three LEDs on the end light momentarily, then all the LEDs go off except for a middle LED which blinks about once a second. If your workstation follows this sequence, self-test has not found a critical problem. (Once UNIX or other programs have gained control of the system, they can use the LEDs in other ways. This description only applies to the power-on sequence.)

If at some point in the above sequence, the lights freeze (keep the same pattern for more than a minute), or the sequence restarts from the beginning, there is a critical hardware problem with the workstation. The appropriate thing to do in this case is to contact Sun Microsystems Field Service or your local Field Service organization. Copy down the pattern of lights (as well as you can, if it is repeating over and over); they contain important diagnostic information for Field Service.

A.1.2. Non-Critical Errors from Self Test

Non-critical errors result in a display like the following:

Self Test found a problem in something
Wrote wdata at address addr, but read rdata.
Damage found, damages
---> Give the above information to your service-person

Sun Workstation, Model model_number, type_of_keyboard.
ROM Rev N, some_number_MB memory installed
Serial #some_number, Ethernet address n:n:n:n:n:n
Auto-boot in progress . . .

something shows what part of the system was most recently found to be malfunctioning.
(If more than one error occurs, a summary of all errors is given in the damages section, and details about the last error are reported here.)

wdata is the data that was written into part of the system, or which was expected to be there if the system was functioning normally.

addr is the address where the data was read and/or written. For memory errors, this is a physical memory address; for other errors, the interpretation of the address depends on something.

rdata is the data that was read back from addr and was found to be invalid because it was not the same as wdata.
A.1.3. No Errors from Self Test

The following display results:

Self Test completed successfully.

Sun Workstation, Model model_number, type_of_keyboard.
ROM Rev N, some_number_MB memory installed
Serial #some_number, Ethernet address n:n:n:n:n:n
Auto-boot in progress . . .

The monitor then begins auto-boot.

A.2. Automatic Boot Procedure

The monitor immediately attempts to boot from a default device (it tries a Xylogics SMD Disk Controller, then a SCSI Disk Controller, and finally tries to boot over the network):

Auto-boot in progress

Boot: disk(0,0,0)vmunix
Load: disk(0,0,0)boot
Boot: disk(0,0,0)vmunix
Size: 215040+ 24576+ 30916 bytes
Sun UNIX 4.2, etc...

Disk is the device name of the "best" local or network disk the monitor could find. The file called vmunix is booted from it. This file does not have to contain a UNIX kernel; it can contain any program you like, as long as the disk is in 4.2BSD UNIX file system format. It is also possible to set up the disk to boot a small program which need not be in a UNIX file system. This discussion assumes that the disk is set up for UNIX.

A.3. Booting from Specific Devices

The Sun Workstation can be booted from:

- Any logical partition of a local disk.
- Any publicly available network disk partition on your Ethernet.
- The first file of a local tape drive.

As mentioned above, the monitor automatically attempts to boot vmunix from a default disk. If you want to boot a different program, or from a different device, you must stop the automatic boot process by aborting. The specific abort sequence depends on your keyboard type. For a Sun-1 Model 100U or 150U, the sequence is either 'SET-UP-A' (hold down the
'SET-UP' key while typing the 'A' key) or 'ERASE-EOF-A'. For a Sun-2 keyboard, the abort sequence is 'L1-A'. For a standard terminal, the BREAK key generates an abort. When you abort, the monitor displays the address where it aborted and a ">" prompt, and waits for you to type a command.

The monitor's boot command looks like:

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

where `device` is the type of hardware to boot from, `parameters` specify the address or partitioning of the device, `pathname` is the name of the actual file (in a UNIX file system on that device) to boot into memory, and `args` are optional arguments to the program.

To determine which devices your monitor ROMs are able to boot from, you can use the command:

```
> b ?
```

The devices are shown in order from "best" to "worst", as used by the automatic boot procedure to select a boot device.

Booting a program involves the cooperation of up to three different programs within your Sun Workstation. The first is the monitor, which reads in the "boot program" or "mini boot program", depending on the device. If the "mini boot" is used, it in turn reads in the "real" boot program. Once the "real" boot program is running, it finds and reads in the program you wanted to run. This process is more or less obvious depending what device you are booting from. Where there are minor differences between the commands you enter to the monitor and to the boot program, these will be noted. In general, the difference is that the monitor commands start with 'b' and the boot program's commands don't.

### A.3.1. Booting from Disk

For disk drives, the format of the boot command is:

```
> b controller/address,drive,partition/pathname args
```

- `controller` names the disk controller which runs the specific disk: ip for the Interphase 2180 disk controller, xy for the Xylogics 440 or 450 controller, or sd for a SCSI disk controller.
- `address` can either be a small number, indicating the nth standard controller board, or is the physical address of the controller on the Multibus.
- `drive` is the unit number of the disk on that specific controller.
- `partition` is a number corresponding to the logical partition on the disk where the file specified by `pathname` can be found. Zero corresponds to partition 'a', 1 to 'b', etc.
- `pathname` is the name of the file to boot.
- `args` are optional arguments.

### A.3.2. Booting from Network Disk

To boot the system from network disk, use a command like:
> b controller(address, hostnumber, partition)pathname args

controller is the device abbreviation for your Ethernet Controller: ec for a 3COM Ethernet Controller, or ie for a Sun-2 Ethernet Controller.

address can either be a small number, indicating the nth standard controller board, or is the physical address of the controller on the Multibus.

hostnumber is an arbitrary number (between 1 and 255) assigned to each machine on a local network to uniquely identify the machine. To find the host number of an existing node, check the node's /etc/hosts file. The entries in the file look something like:

<table>
<thead>
<tr>
<th>Address</th>
<th>Hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.9.1.1</td>
<td>winkin</td>
</tr>
<tr>
<td>192.9.1.2</td>
<td>blinkin</td>
</tr>
<tr>
<td>192.9.1.3</td>
<td>nod</td>
</tr>
<tr>
<td>192.9.1.24</td>
<td>henry</td>
</tr>
</tbody>
</table>

The last component of the complete internet address is the host number.

Henry's host number here is 24. Note that you must supply the hostnumber to the monitor in hexadecimal; the numbers in /etc/hosts may be in decimal, so you may have to convert.

Using zero as hostnumber is valid here, and means 'whichever host is my net disk server.'

partition is the desired public partition number on the specified server. The correspondence between this number and a real disk partition is defined in /etc/nd.local on the server machine.

pathname is the name of the file to boot.

args are optional arguments.

A.3.3. Booting from Tape

The Sun Workstation can be booted from industry-standard nine-track magnetic tape, from a 1/4-inch cartridge tape controlled by a Sun 1/4-inch tape controller, or from a 1/4-inch tape controlled by a SCSI tape controller, using the following command:

> b tape(controller, unit, filenum)

tape is the device abbreviation for your tape controller: ar for a Sun 1/4-inch tape controller, mt for nine-track tape, or st for a SCSI tape controller.

controller is a small number indicating the nth standard magnetic tape controller in the system, or is the Multibus address of the controller.

unit specifies which tape drive on the controller is to be used.

filenum specifies which file of the tape is to be booted. By convention, boot commands number the first file on the tape file #0, the second #1, and so on.

The monitor ignores the supplied value of filenum and can only boot the first file on a tape. To boot a file further down the tape, use the monitor to boot the "boot" program. Sun-supplied UNIX distribution tapes always have the "boot" program on the first file of the tape.
A.3.4. Booting Files from the Default Device

To boot any file from the default device, enter:

```
b pathname args
```

This is useful for booting standalone utility programs, once your disk or network disk is set up, or for trying new versions of the UNIX kernel.

A.4. Messages from the Monitor and Boot Program

Abort at aaaaaa
The monitor has aborted execution of the current program because you entered the "abort sequence" (upper left key held while pressing "A") from the Sun keyboard, or pressed BREAK on a serial console. aaaaaa is the address of the next instruction. You can continue the program from there by entering the "c" command.

Address Error, addr: zzzzz at aaaaaa
The current program has stopped because it made an invalid memory access. zzzzz is the (invalid) address; aaaaaa is an address near the instruction which failed (typically two to ten bytes beyond). There is no general way to recover from this error, except to debug the program.

ar: cartridge is write protected
The current program is trying to write on an Archive tape cartridge, but the "Safe" switch at the top left corner of the cartridge is set to prevent writing on the tape.

ar: zzzz error
The monitor or boot program is trying to boot from an Archive tape, and encountered an unexpected error. The status bytes zzzz can be decoded by looking under "Read Status Command" in the Archive Product Manual. This error could be caused by incorrect cables, a bad tape, or other problems.

ar: drive not responding
The monitor is trying to boot from an Archive tape, but can get no response from the tape drive. This can occur if your system contains an Archive controller board but no tape drive, or if the tape drive's cable is loose or disconnected, or if the tape drive's power is not on.

ar: invalid state zz
This message indicates that the standalone I/O system has a bug in its Archive driver.

ar: no cartridge
ar: no cartridge in drive
The monitor or boot program is trying to boot from an Archive tape, but there is no cartridge in the tape drive.

ar: no drive
The monitor or boot program is trying to boot from an Archive tape, but the specified drive does not exist. Typical Archive configurations include only drive 0.

ar: RDST gave Exception, retrying
The current program is trying to use the Archive tape drive, and encountered an error. The error is probably caused by hardware. Check the cable(s) that connect the tape drive to the system.
ar: triggered at idle zz
This message indicates that the standalone I/O system has a bug in its Archive driver.

Auto-boot in progress...
The monitor has finished its power-on sequence and is looking for a good device to boot the Unix system from.

Bad device
The current program (possibly the boot program) has tried to open a file without a device name (eg /sys/). This could mean that the boot command you typed had no device name.

Bad format
The boot program is trying to boot from a file which is not in a standard UNIX a.out(5) format. The boot program can only boot files which are in this format, which is generated by the ld(1) command.

bn negative
bn ovf dd
bn void dd
A standalone program (such as the boot program) is trying to read a file from disk or net disk, and the block number it is trying to read is invalid.

Boot:
The boot program is waiting for you to specify a device and file name to boot from. The boot program accepts the same commands that the monitor would, without the initial 'b'. See the section Booting From Specific Device, above.

Boot syntax: b [!] [dev(ctlr,unit,part)] name [options]
boot syntax: dev(ctlr,unit,part) name
You have entered an invalid boot command. This message describes the general format of the boot command to remind you. The first form is used by the monitor; the second (without the 'b') is used by the boot program. Don't type the brackets; they indicate optional parts of the command.

Boot failed
The boot program has tried to boot the device and/or file you specified, but could not. A preceding message should give more details about why.

Boot syntax: b [!] [dev(ctlr,unit,part)] name [options]

Bus Error, addr: zzzzzz at aaaaaa
The current program has stopped because it tried to make an invalid memory access. The reason for the error is shown before this message. The memory location being accessed was zzzzzz, and the instruction which made the access is near location aaaaaa. There is no way to recover from this error, in general, except to debug the program.

Can't write files yet... Sorry
The current program is trying to write to a disk or network disk file thru the standalone I/O system. Writing on files (as opposed to writing on devices) is not supported
when running standalone (i.e. before booting the Unix kernel).

Corrupt label

Corrupt label on head \( h \)

The monitor or boot program is trying to boot from a disk. The first sector of the disk appears to be a label (as it ought to be), but the checksum on the label is wrong. Try again a few times; if the problem recurs, you should probably relabel your disk. See sections "Using the Diag Utility" and "Labelling the Disk" in the chapter, *Installing UNIX for the First Time*. Before trying to relabel your disk, make sure that you know what ought to be in the label — writing the wrong label on the disk is highly likely to cause destruction of some or all files on the disk.

count=\( ddd \)

A standalone program is trying to write to a device and has specified a block size which is not a multiple of 512. The write proceeds anyway, but may cause incorrect results.

Damage found, *damage*...

As part of the power-on self test procedure, the monitor has found damage in one or more parts of the system. This message should be reported to your local service representative or Sun Microsystems Field Service. *Damage* is a list of subsystem names, such as "memory" or "timer".

Exception *ee* at \( aaaaaa \)

The current program has stopped because it got an interrupt. The interrupt could have been caused either by hardware or software. *ee* is the hexadecimal address of the interrupt vector used; you can look it up on a Motorola 68010 or 68000 reference card or CPU manual to see what kind of interrupt has occurred. \( aaaaaa \) is the address of the instruction where the interrupt occurred.

Extra chars in command

Your previous 'u' command had extra, unrecognized characters on the end.

FCn space

The address space being accessed by the monitor's memory reference commands is defined by Function Code number \( n \). See the Motorola 68010 CPU manual for more information. This message is printed by the 's' command.

For phys part \( p \), No label found.

The boot program is trying to boot from a nonzero "physical partition" on a disk, and can't find a label. Physical partitions are used for disk drives part of which are fixed and part of which are removable.

\( \rightarrow \) Give the above information to your service-person.

The monitor has found a hardware problem while executing its power-on self test procedure. The preceding messages describe the error in more detail. You should report the problem to your local service staff, or to Sun Microsystems Field Service.

Giving up...

See "Waiting for disk to spin up...". The monitor has given up on waiting for the disk to become ready.

ie: cannot initialize

The monitor or boot program is trying to boot from a Sun-2 Ethernet controller, and something serious has gone wrong with the board. Call your local Sun Microsystems Field Service person.

ip: error \( xx \)

The monitor or boot program is trying to boot from an Interphase disk controller, and...
encountered an unexpected error. The error number 'zz' can be decoded by looking in appendix B of the Interphase SMD 2180 Storage Module Controller/Formatter User's Guide. This problem is usually caused by loose or unplugged disk drive cables.

**ID PROM INVALID**
The monitor cannot find a valid ID PROM on the CPU board. The ID PROM contains the machine's serial number and other information specific to your system. If you have recently changed CPU boards, it is possible that you installed the ID PROM incorrectly. You should attempt to locate the correct ID PROM and install it in your CPU board.

**Invalid Page Bus Error ...**
See "Bus Error...". The attempted access was invalid because the virtual page containing the addressed data has been designated as invalid. It usually means that your program is using the wrong address.

**Invalid selection**
Your last 'u' command was not correct.

**Keyboard error detected**
The microprocessor on the keyboard has reported an error. This probably means that your keyboard hardware is broken and should be replaced.

**Load: dest(cfr,unit,part)boot**
The monitor has loaded in the "mini" boot program from a disk drive or network disk. The mini boot is now reading in the "real" boot program from the disk. The "real" boot program will then read in the program you requested.

**Lower Byte Parity Bus Error ...**
See "Bus Error..." and "Parity...". The preceding access was to a word in memory with a parity error in its lower byte.

**Misplaced label on head n**
The monitor or boot program is trying to boot from a disk. It has found a label which seems to identify itself as belonging to a different read/write head from the one where the label is written. See "Corrupt label" above.

**mt: controller does not initialize**
The monitor is trying to boot from nine-track tape, and could not get the tape controller to complete its initialization sequence. This might indicate a possible defect in the controller, or incorrect configuration of the controller board.

**mt: error Oxx**
The monitor is trying to boot from nine-track tape, and encountered an unexpected error. The error number 'zz' can be decoded by looking in appendix C of the Tapemaster Product Specification, which is supplied with your tape drive.

**mt: unit not ready**
The monitor is trying to boot from nine-track tape, but the tape drive is not ready. Check to see that the drive is on-line.

**nd: no file server, giving up.**
The monitor or boot program is trying to boot from a network disk server over the Ethernet. It has been retrying for a long time and there is no response from the server. Check the Ethernet address in the boot command; if it is zero, make sure your machine's Ethernet address is recorded in the server's /etc/nd.local file. If that's OK, check your Ethernet cable connection, see whether the server is running correctly, and/or see whether other machines on the network can communicate.
No controller at mbio zzzz
The monitor is trying to boot, but it can’t find a device controller where you asked it to look. You should try another boot command, or make sure that your controller board is plugged in and has all its jumpers and switches set properly.

No default boot devices
The monitor is trying to boot but it can’t find a disk or Ethernet interface to boot from. To boot from a tape, you must specify the device name explicitly, as in ‘b ar()’.

No label found.
No label found -- attempting boot anyway.
The monitor or boot program is trying to boot from a disk, and can’t find a valid label on the disk. This is best fixed by booting a copy of diag(8S) from a different device (for example, network disk or tape) and using the ‘verify label’ and ‘label’ commands. See the warning under “Corrupt label” above. This error might also be caused by missing or bad disk cables.

No more file slots
The current program is using the standalone I/O library and has opened too many devices or files.

Not a directory
The current program (possibly the boot program) has tried to open a disk or network disk file with a pathname, but one of the names in the path is not a directory.

name not found
The boot program has searched for the requested file, but cannot find it. You can retry your boot command, using “*” instead of name, to get a list of the names that exist in that directory.

null path
The current program (possibly the boot program) has tried to open a file whose name is empty.

PageMap aaaaa [se]: zzzzzzzz!
The monitor is displaying or modifying a page map entry because you entered a ‘p’ command. aaaaaa is the virtual memory address whose map entry is being examined. se is the segment map entry which is being used to map this page map entry and page. zzzzzzzz is the page map entry itself. You can enter a space and type ‘RETURN’ to get back to command mode.

Parity Bus Error ...
See “Bus Error...”. The attempted access was probably valid, but was cancelled because the preceding access was to memory with bad parity. (Parity errors are reported on the memory cycle after the failing cycle.) If neither “Upper Byte” nor “Lower Byte” is reported, the parity on both bytes was invalid. The access address printed in the Bus Error message is probably not relevant to the parity error. There is no general way to recover from this error; a good starting point, though, is to boot pars can(8S), which will search all of memory for parity errors.

Please clear keyboard to begin
The monitor is trying to listen for your typing on the keyboard, but cannot tell which shift keys are down until you release all the locking keys (Caps Lock and Shift Lock). Once it has seen all the keys released, it can then track the movements of the keys and typing will work.
Please start it, if necessary, -OR- press any key to quit.
See "Waiting for disk to spin up...".

Possible boot devices:
You have asked for a list of boot devices with the 'b ?' command.

Protection Bus Error ...
See "Bus Error...". The attempted access was invalid because your program is not permitted to access the addressed data in this way; for example, writing to that page is disallowed.

ROM Rev z, mm memory installed
The monitor is identifying its revision level and the system configuration as part of the power-on sequence. z is a letter or phrase indicating which particular version of the monitor is installed; mm shows how much memory was found during system configuration at power-on. If an even number of megabytes is installed, mm is displayed as "nnMB"; otherwise as "nnnnKB".

Retensing...
The monitor is attempting to boot from an Archive tape. Its first attempt failed, so it is retensing the tape (winding all the tape from one reel to the other), which makes it much more likely to succeed.

Seek not from beginning of file
The current program is using the standalone I/O library and has tried to do an unsupported seek operation.

SegMap asaaaa: xx?
The monitor is examining or changing the segment map in response to your recent 'm' command. You can enter a space and type 'RETURN' to get back to command mode.

Self Test completed successfully
The monitor has completed its power-on self test without finding any hardware problems.

Self Test found a problem in something
The monitor has completed its power-on self test and found a problem in some subsystem. Something describes the general location of the error. Further messages give more details; see "Wrote ..." and "Damage found...".

Serial #some_number, Ethernet address n:n:n:n:n:n
The monitor is identifying your machine's serial number and hardware Ethernet address as part of the power-on sequence. The hardware Ethernet address is taken from the ID PROM on the Sun-2 CPU board, and is given as a 6-byte hexadecimal value with a colon between each byte. A typical Ethernet address might be "8:0:20:1:1:A3".

Short read
The boot program is trying to boot a program from disk or net disk. It has located the program, but encountered an error while reading it into memory.

Size: text + data + bss bytes
The boot program is loading in the program you requested. Text, data, and bss are the sizes of the three sections of the program; they are printed as each is read into memory. After finishing display of this message, the boot program begins execution of your program; further messages can come from it instead of from the boot program or monitor.

Sun Workstation, Model Sun-1/100U or Sun-1/150U, keyb keyboard
The Model 100U or 150U workstation has just been powered on, or you entered a 'kb'

Revision H of 12 March 1984
command, and the monitor is identifying its configuration. *Keyb* is either VT100 or Two-tone, depending which keyboard your monitor ROMs support.

**Sun Workstation, Model Sun-2/120 or Sun-2/170, Sun-2 keyboard**

The Model 120 or 170 workstation has just been powered on, or you entered a 'kb' command, and the monitor is identifying its configuration.

**Timeout Bus Error ...**

See "Bus Error...". The attempted access was invalid because no device responded at the addressed location. This most often happens for Multibus references. The program was probably trying to access a device or section of memory which does not exist, or which has gotten into a hung state. If this occurs in response to a boot command, the device you are trying to boot from is not installed in your system.

```
tm: error nn during config of ctlr cc
```

```
tm hard err nn
```

```
tm: no response from ctlr cc
```

A standalone program (possibly the boot program) is trying to use the Tapemaster nine-track tape drive, and has encountered an error. This could be caused by a bad or missing tape, loose or misplugged cables, incorrect jumpers on the Tapemaster controller board, or hardware errors. *Nn* can be decoded by looking in the Tapemaster *Product Specification*.

**Unknown device**

The current program (possibly the boot program) has tried to use a device which is unknown to the standalone I/O system.

**Upper Byte Parity Bus Error ...**

See "Bus Error..." and "Parity...". The preceding access was to a word in memory with a parity error in its upper byte.

```
u i, uoo, uaaabaud, ubbbaud, uaaaaaa, uecho
```

The monitor is describing its console and serial port configuration in response to a 'u' command. *I* is the input device (k for keyboard, or a or b for a serial port); o is the output device (s for screen, or a or b); abaud and bbaud are the baud rates on the serial ports; aaaaaa is the address of the Zilog 8530 chip which implements the serial ports, and *echo* is 'e' if input echoing is enabled ("full duplex") or 'ne' if disabled ("half duplex").

**Using RS232 A input.**

The monitor did not find the Sun keyboard, so it is taking input from one of the serial ports on the back of the Workstation, marked "RS232 A". If this is unexpected, make sure that the keyboard is plugged into the correct socket on the workstation. The keyboard must be plugged in before system power is turned on. If you connect a Sun keyboard after this message appears, you can let the monitor know about the keyboard by entering the Abort sequence (hold down the upper left key on the Sun keyboard, and press 'A'). The monitor will switch to using the Sun keyboard since that's where the Abort was typed. Then type 'c' to continue whatever program was running when you aborted.

If you don't want to use a Sun keyboard, connect a normal ASCII terminal to the "RS232 A" connector on the back panel. Configure the terminal for 9600 baud, no parity, one stop bit. Things that you type on the terminal will be displayed on the Sun video screen, if you have one, or on the terminal's screen.
Waiting for disk to spin up...

The monitor is trying to boot from a disk. The disk is not ready, so the monitor is waiting in the hope that the disk is just starting to spin and will become ready soon. If you get this message when the power has been on for a while, your disk cables are probably loose or disconnected.

Watchdog reset!

The current program has stopped executing with a "double bus fault". This is explained in detail in the Motorola 68010 manual; the two most common causes are that low memory (interrupt vectors) has been overwritten, or the system stack pointer is pointing to an invalid address. There is a serious bug in your program if this occurs.

What?

You typed a command that the monitor does not recognize. Try again.

Wrote wdata at address addr, but read rdata.

The monitor has completed its power-on self test and found a problem in some subsystem. The preceding "Self Test found a problem..." message describes which part of the system was in error. This message gives more details about the error. Wdata is the data that was written into part of the system, or which was expected to be there if the system was functioning normally. Addr is the address where the data was read and/or written. For memory errors, this is a physical memory address; for other errors, the interpretation of this field depends on what subsystem was being tested. Rdata is the data that was read back from addr and was found to be invalid because it was not the same as wdata. This information should be written down and reported to your local Field Service organization, or to Sun Microsystems Field Service. See the section Non-Critical Errors From Self Test above.

xy: error nn cmd zz

xy: error nn bno bbbbb

xy: init error zz

The monitor is trying to boot from the Xylogics disk and has encountered an error. The command being executed at the time is defined by the hexadecimal value zz (if present); the block number is bbbbb (if present), and the particular error is encoded as nn. The error and command can be decoded by looking in the Xylogics manual.

xy: no bad block info

The boot program is trying to read from the Xylogics disk, but can't find the information about bad blocks on the disk. It continues, but if the program attempts to read any bad blocks (which have been remapped to elsewhere on the disk), the attempt will fail.

zero length directory

A standalone program (possibly the boot program) is trying to read a file from disk, but one of the directories in the path name has no files in it. The file system should be checked and fixed by using fsck(8).
NAME
intro - introduction to system maintenance and operation commands

DESCRIPTION
This section contains information related to system bootstrapping, operation and maintenance
and describes all the server processes and daemons which run on the system.

Disk formatting and labelling is done by diag(8S) which participates in most system bootstraps. Bootstrapping of the system is described in reboot(8). The standard set of commands run by the system when it boots is described in rc(8S). Related commands include ones to check the consistency of file systems jsck(8), mount and unmount file systems mount(8) and umount(8), add swap devices swapon(8), cause all outstanding disk I/O to complete sync(8), shutdown or reboot a running system shutdown(8S), halt(8), and reboot(8), set the time on a machine from the time on another machine rdate(8).

Creation of file systems is discussed in mafs(8) and newfs(8). File system performance parameters are adjustable with tunefs(8). File system saves and restores are described in dump(8) and restore(8).

Procedures for adding new users to a system are described in adduser(8) using vipw(8).

Other programs useful when the system crashes or hardware is broken include gztst(8S) which tests the frame buffer on a workstation, imemtst(8S) which tests the memory, crash(8S) which describes what happens when the system crashes, savecore(8) and analyze(8) which can be used to analyze system crash dumps. Occasionally useful as adjuncts to the jsck(8) file system repair program are clri(8), dcheck(8), icheck(8), and ncheck(8).

Configuring a new version of the UNIX kernel requires using the program config(8); major system bootstraps often require the use of mkproto(8). New devices are made in the /dev directory when device drivers are added to the system by using the makedev(8) and mknod(8) commands. If you have source, you will use the install(8) command to reinstall freshly compiled programs, and catman(8) to reformat the pre-formatted version of the manual.

Resource accounting is enabled by the accton(8) command, and summarized by sa(8). Login time accounting is performed by ac(8).

A number of service and daemon processes are described here. The cron(8) daemon forces delayed disk I/O to occur and runs periodic events (such as removing temporary files from the disk periodically). The dmesg(8) process is invoked by cron and keeps the system error log. The init(8) process is the initial process created when UNIX boots and manages the reboot process and creates the initial login prompts on the various system terminals through the services of getty(8).

The Internet super-server inetd(8C) invokes all other internet servers as needed. These servers include the remote shell servers rshd(8C) and reccd(8C) the remote login server rlogind(8C) the FTP and TELNET daemons ftpd(8C) and telnetd(8C), the TFTP daemon tftpd(8C) and the mail arrival notification daemon cometd(8C). Other network daemons include the 'load average/who is logged in' daemon rwhod(8C), the routing daemon routed(8C), and the mail daemon sendmail(8).

If network protocols are being debugged, then the protocol debugging trace program trpt(8C) is often useful. Remote magnetic tape access is provided by reh and rmt(8C). Remote line printer access is provided by lpd(8) and control over the various print queues is had through lpc(8). Printer cost accounting is done through pacc(8).

Network host tables may be gotten from the ARPA NIC using gettable(8C) and converted to UNIX usable format using htable(8).

Sun Release 1.1 Last change: 9 March 1984 1
**LIST OF PROGRAMS**

<table>
<thead>
<tr>
<th>Program</th>
<th>Appears on Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ac</td>
<td>ac.8</td>
<td>login accounting</td>
</tr>
<tr>
<td>accnt</td>
<td>sa.8</td>
<td>system accounting</td>
</tr>
<tr>
<td>adbgen</td>
<td>adbgen.8</td>
<td>generate adb script</td>
</tr>
<tr>
<td>adduser</td>
<td>adduser.8</td>
<td>procedure for adding new users</td>
</tr>
<tr>
<td>analyze</td>
<td>analyze.8</td>
<td>Virtual UNIX postmortem crash analyzer</td>
</tr>
<tr>
<td>arp</td>
<td>arp.8c</td>
<td>address resolution display and control</td>
</tr>
<tr>
<td>catman</td>
<td>catman.8</td>
<td>create the cat files for the manual</td>
</tr>
<tr>
<td>chown</td>
<td>chown.8</td>
<td>change owner</td>
</tr>
<tr>
<td>chri</td>
<td>chri.8</td>
<td>clear i-node</td>
</tr>
<tr>
<td>comsat</td>
<td>comsat.8c</td>
<td>biff server</td>
</tr>
<tr>
<td>config</td>
<td>config.8</td>
<td>build system configuration files</td>
</tr>
<tr>
<td>crash</td>
<td>crash.8s</td>
<td>what happens when the system crashes</td>
</tr>
<tr>
<td>cron</td>
<td>cron.8</td>
<td>clock daemon</td>
</tr>
<tr>
<td>dcheck</td>
<td>dcheck.8</td>
<td>file system directory consistency check</td>
</tr>
<tr>
<td>diag</td>
<td>diag.8s</td>
<td>General-purpose stand-alone utility package</td>
</tr>
<tr>
<td>dmesg</td>
<td>dmesg.8</td>
<td>collect system diagnostic messages to form error log</td>
</tr>
<tr>
<td>dump</td>
<td>dump.8</td>
<td>incremental file system dump</td>
</tr>
<tr>
<td>dumpfs</td>
<td>dumpfs.8</td>
<td>dump file system information</td>
</tr>
<tr>
<td>expire</td>
<td>expire.8</td>
<td>remove outdated news articles</td>
</tr>
<tr>
<td>fastboot</td>
<td>fastboot.8</td>
<td>reboot/halt the system without checking the disks</td>
</tr>
<tr>
<td>fasthalt</td>
<td>fasthalt.8</td>
<td>reboot/halt the system without checking the disks</td>
</tr>
<tr>
<td>fsck</td>
<td>fsck.8</td>
<td>file system consistency check and interactive repair</td>
</tr>
<tr>
<td>ftpd</td>
<td>ftpd.8c</td>
<td>DARPA Internet File Transfer Protocol server</td>
</tr>
<tr>
<td>gettable</td>
<td>gettable.8c</td>
<td>get NIC format host tables from a host</td>
</tr>
<tr>
<td>getty</td>
<td>getty.8</td>
<td>set terminal mode</td>
</tr>
<tr>
<td>gxtest</td>
<td>gxtest.8s</td>
<td>stand alone test for the Sun video graphics board</td>
</tr>
<tr>
<td>halt</td>
<td>halt.8</td>
<td>stop the processor</td>
</tr>
<tr>
<td>htable</td>
<td>htable.8</td>
<td>convert NIC standard format host tables</td>
</tr>
<tr>
<td>icheck</td>
<td>icheck.8</td>
<td>file system storage consistency check</td>
</tr>
<tr>
<td>ifconfig</td>
<td>ifconfig.8c</td>
<td>configure network interface parameters</td>
</tr>
<tr>
<td>imemtest</td>
<td>imemtest.8s</td>
<td>stand alone memory test</td>
</tr>
<tr>
<td>inetd</td>
<td>inetd.8c</td>
<td>internet services daemon</td>
</tr>
<tr>
<td>init</td>
<td>init.8</td>
<td>process control initialization</td>
</tr>
<tr>
<td>iostat</td>
<td>iostat.8</td>
<td>report I/O statistics</td>
</tr>
<tr>
<td>kgmon</td>
<td>kgmon.8</td>
<td>generate a dump of the operating system's profile buffers</td>
</tr>
<tr>
<td>lpc</td>
<td>lpc.8</td>
<td>line printer control program</td>
</tr>
<tr>
<td>lpd</td>
<td>lpd.8</td>
<td>line printer daemon</td>
</tr>
<tr>
<td>MAKEDEV</td>
<td>makedev.8</td>
<td>make system special files</td>
</tr>
<tr>
<td>makekey</td>
<td>makekey.8</td>
<td>generate encryption key</td>
</tr>
<tr>
<td>mkfs</td>
<td>mkfs.8</td>
<td>construct a file system</td>
</tr>
<tr>
<td>mknod</td>
<td>mknod.8</td>
<td>build special file</td>
</tr>
<tr>
<td>mkproto</td>
<td>mkproto.8</td>
<td>construct a prototype file system</td>
</tr>
<tr>
<td>mount</td>
<td>mount.8</td>
<td>mount and dismount file system</td>
</tr>
<tr>
<td>ncheck</td>
<td>ncheck.8</td>
<td>generate names from i-numbers</td>
</tr>
<tr>
<td>nd</td>
<td>nd.8c</td>
<td>network disk control</td>
</tr>
<tr>
<td>netstat</td>
<td>netstat.8</td>
<td>show network status</td>
</tr>
<tr>
<td>newaliases</td>
<td>newaliases.8</td>
<td>rebuild the data base for the mail aliases file</td>
</tr>
<tr>
<td>news</td>
<td>news.8</td>
<td>construct a new file system</td>
</tr>
<tr>
<td>pac</td>
<td>pac.8</td>
<td>printer/plotter accounting information</td>
</tr>
<tr>
<td>pstat</td>
<td>pstat.8</td>
<td>print system facts</td>
</tr>
<tr>
<td>quot</td>
<td>quot.8</td>
<td>summarize file system ownership</td>
</tr>
</tbody>
</table>

Last change: 9 March 1984
Sun Release 1.1
rc          rc.8            command script for auto-reboot and daemons
rdate       rdate.8        set system date from a remote host
reboot      reboot.8       UNIX bootstrapping procedures
recnews     recnews.8      receive unprocessed articles via mail
renice      renice.8       alter priority of running processes
restore     restore.8      incremental file system restore
reexecd     reexecd.8c     remote execution server
rlogind     rlogind.8c     remote login server
rmail       rmail.8        handle remote mail received via uucp
rmt          rmt.8c        remote magtape protocol module
route        route.8c      manually manipulate the routing tables
routed       routed.8c     network routing daemon
rshd         rshd.8c       remote shell server
rwhod        rwhod.8c      system status server
sa           sa.8          system accounting
savecore     savecore.8    save a core dump of the operating system
sendmail     sendmail.8    send mail over the internet
sendnews     sendnews.8    send news articles via mail
shutdown     shutdown.8    close down the system at a given time
sticky       sticky.8      executable files with persistent text
swapon       swapon.8      specify additional device for paging and swapping
sync         sync.8         update the super block
syslog       syslog.8      log systems messages
telnetd      telnetd.8c    DARPA TELNET protocol server
tftpd        tftpd.8c      DARPA Trivial File Transfer Protocol server
timed        timed.8c      DARPA Time server
trpt          trpt.8c     transliterate protocol trace
tunefs       tunefs.8      tune up an existing file system
umount       umount.8      mount and dismount file system
update       update.8      periodically update the super block
uuclean      uuclean.8c    uucp spool directory clean-up
uurec        uurec.8      receive processed news articles via mail
vipw          vipw.8      edit the password file
vmstat       vmstat.8      report virtual memory statistics
NAME
    ac – login accounting

SYNOPSIS
    /usr/etc/ac [-w wtmp] [-p] [-d | people] ...

DESCRIPTION
    Ac produces a printout giving connect time for each user who has logged in during the life of the current wtmp file. A total is also produced.

    The accounting file /usr/adm/wtmp is maintained by init and login. Neither of these programs creates the file, so if it does not exist no connect-time accounting is done. To start accounting, it should be created with length 0. On the other hand if the file is left undisturbed it will grow without bound, so periodically any information desired should be collected and the file truncated.

OPTIONS
    -w specifies an alternate wtmp file.
    -p prints individual totals; without this option, only totals are printed.
    -d printout for each midnight to midnight period. Any people will limit the printout to only the specified login names. If no wtmp file is given, /usr/adm/wtmp is used.

FILES
    /usr/adm/wtmp

SEE ALSO
    init(8), sa(8), login(1), utmp(5).
NAME
adbgen - generate adb script

SYNOPSIS
/usr/lib/adb/adblen file.adb ...

DESCRIPTION
Adbgen makes it possible to write adb(1) scripts that do not contain hard-coded dependencies on structure member offsets. The input to adbgen is a file named file.adb which contains adbgen header information, then a null line, then the name of a structure, and finally an adb script. Adbgen only deals with one structure per file; all member names are assumed to be in this structure. The output of adbgen is an adb script in file. Adbgen operates by generating a C program which determines structure member offsets and sizes, which in turn generates the adb script.

The header lines, up to the null line, are copied verbatim into the generated C program. Typically these include C #include statements to include the header files containing the relevant structure declarations.

The adb script part may contain any valid adb commands (see adb(1)), and may also contain adbgen requests, each enclosed in {}s. Request types are:

1. Print a structure member. The request form is {member,format}. Member is a member name of the structure given earlier, and format is any valid adb format request. For example, to print the p_pid field of the proc structure as a decimal number, you would write {p_pid,d}.

2. Reference a structure member. The request form is {*member,base}. Member is the member name whose value is desired, and base is an adb register name which contains the base address of the structure. For example, to get the p_pid field of the proc structure, you would get the proc structure address in an adb register, say <f, and write {*p_pid,<f}.

3. Tell adbgen that the offset is ok. The request form is {OFFSETOK}. This is useful after invoking another adb script which moves the adb dot.

4. Get the size of the structure. The request form is {SIZEOF}. Adbgen replaces this request with the size of the structure. This is useful in incrementing a pointer to step through an array of structures.

5. Get the offset to the end of the structure. The request form is {END}. This is useful at the end of the structure to get adb to align the dot for printing the next structure member.

Adbgen keeps track of the movement of the adb dot and emits adb code to move forward or backward as necessary before printing any structure member in a script. Adbgen's model of the behavior of adb's dot is simple: it is assumed that the first line of the script is of the form struct_address/adb text and that subsequent lines are of the form +/adb text. This causes the adb dot to move in a sane fashion. Adbgen does not check the script to ensure that these limitations are met. Adbgen also checks the size of the structure member against the size of the adb format code and warns you if they are not equal.

EXAMPLE
If there were an include file x.h which contained:

```c
struct x {
  char *x_cp;
  char x_c;
  int x_j;
};
```
Then an `adbgen` file (call it `script.adb`) to print it would be:

```c
#include "x.h"

x
./"x_cp"16t"x_c"8t"x_i"n{x_cp,X}{x_c,C}{x_i,D}
```

After running `adbgen` the output file `script` would contain:

```c
./"x_cp"16t"x_c"8t"x_i"nXC+D
```

To invoke the script you would type:

```
x$<script
```

**DIAGNOSTICS**

Warnings about structure member sizes not equal to `adb` format items and complaints about badly formatted requests. The C compiler complains if you reference a structure member that does not exist. It also complains about `&` before array names; these complaints may be ignored.

**FILES**

```
/usr/lib/adb/* adb scripts for debugging the kernel
```

**SEE ALSO**

`adb(1)`, Using ADB to Debug the UNIX Kernel

**BUGS**

`Adb` syntax is ugly; there should be a higher level interface for generating scripts.

Structure members which are bit fields cannot be handled because C will not give the address of a bit field. The address is needed to determine the offset.
NAME
adduser – procedure for adding new users

DESCRIPTION
A new user must choose a login name, which must not already appear in /etc/passwd. An account can be added by editing a line into the passwd file; this must be done with the password file locked, for example, by using vipw(8).

A new user is given a group and user id. User id's should be distinct across a system, since they are used to control access to files. Typically, users working on similar projects will be put in the same group. System staff is group '10' for historical reasons, and the super-user is in this group.

A skeletal account for a new user 'esmerelda' would look like:

```
esmerelda:235:20: & Featherstonehaugh:/usr/esmerelda:/bin/csh
```

Fields in the password file have the following meanings:

1. Login name ('esmerelda').
2. Encrypted password. Set the user's password with passwd(1).
3. User ID.
4. Group ID.
5. This field is called the 'GCOS' field (from earlier implementation of UNIX) and is traditionally used to hold the user's full name. Some installations have other information encoded in this field. From this information we can tell that esmerelda's real name is 'Esmerelda Featherstonehaugh'. The & here is a shorthand for the user's login name.
7. Initial shell which this user will see on login. If this field is empty, sh(1) is used as the initial shell.

It is useful to give new users some help in getting started, supplying them with a few skeletal files such as .profile if they use '/bin/sh', or .cshrc and .login if they use '/bin/csh'. New users should be given copies of these files which, for instance, arrange to use tset(1) automatically at each login.

FILES
/etc/passwd password file

SEE ALSO
passwd(1), chsh(1), passwd(5), vipw(8)
NAME
analyze - Virtual UNIX postmortem crash analyzer

SYNOPSIS

DESCRIPTION
Analyze is the post-mortem analyzer for the state of the paging system. In order to use analyze, you must arrange to get a image of the memory (and possibly the paging area) of the system after it crashes (see crash(8S)).

The analyze program reads the relevant system data structures from the core image file and indexing information from /vmunix (or the specified file) to determine the state of the paging subsystem at the point of crash. It looks at each process in the system, and the resources each is using in an attempt to determine inconsistencies in the paging system state. Normally, the output consists of a sequence of lines showing each active process, its state (whether swapped in or not), its pdir, and the number and location of its page table pages. Any pages which are locked while raw i/o is in progress, or which are locked because they are intransit are also printed. (Intransit text pages often diagnose as duplicated; you will have to weed these out by hand.)

The program checks that any pages in core which are marked as not modified are, in fact, identical to the swap space copies. It also checks for non-overlap of the swap space, and that the core map entries correspond to the page tables. The state of the free list is also checked.

Options to analyze:
- D causes the diskmap for each process to be printed.
- d causes the (sorted) paging area usage to be printed.
- f which causes the free list to be dumped.
- m causes the entire coremap state to be dumped.
- v (long unused) which causes a hugely verbose output format to be used.

In general, the output from this program can be confused by processes which were forking, swapping, or exiting or happened to be in unusual states when the crash occurred. You should examine the flags fields of relevant processes in the output of a pstat(8) to weed out such processes.

It is possible to look at the core dump with adb if you do
adb -k /vmunix /vmcore

FILES
/vmunix default system namelist

SEE ALSO
adb(1S), ps(1), crash(8S), pstat(8)

AUTHORS
Ozalp Babaoglu and William Joy

DIAGNOSTICS
Various diagnostics about overlaps in swap mappings, missing swap mappings, page table entries inconsistent with the core map, incore pages which are marked clean but differ from disk-image copies, pages which are locked or intransit, and inconsistencies in the free list.

It would be nice if this program analyzed the system in general, rather than just the paging system in particular.

8 Last change: 4 February 1983 Sun Release 1.1
NAME
arp – address resolution display and control

SYNOPSIS
arp hostname
arp -a [ vmunix ] [ kmem ]
arp -d hostname
arp -s hostname ether_addr [ temp ] [ pub ]
arp -f filename

DESCRIPTION
The arp program displays and modifies the Internet-to-Ethernet address translation tables used by
the address resolution protocol (arp(4p)).

With no flags, the program displays the current ARP entry for hostname. With the -a flag, the
program displays all of the current ARP entries by reading the table from the file kmem (default
/dev/kmem) based on the kernel file vmunix (default /vmunix).

With the -d flag, a super-user may delete an entry for the host called hostname.

The -s flag is given to create an ARP entry for the host called hostname with the Ethernet
address ether_addr. The Ethernet address is given as six hex bytes separated by colons. The
entry will be permanent unless the word temp is given in the command. If the word pub is
given, the entry will be "published", e.g., this system will respond to ARP requests for hostname
even though the hostname is not its own.

The -f flag causes the file filename to be read and multiple entries to be set in the ARP tables.
Entries in the file should be of the form

hostname ether_addr [ temp ] [ pub ]

with argument meanings as given above.

SEE ALSO
arp(4p), ifconfig(8c)
NAME
   catman – create the cat files for the manual
SYNOPSIS
   /usr/etc/catman [-p] [-n] [-w] [sections]
DESCRIPTION
   Catman creates the preformatted versions of the on-line manual from the nroff input files. Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, catman recreates the /usr/lib/whatis database.
   If there is one parameter not starting with a ‘-’, it is take to be a list of manual sections to look in. For example
   catman 123
   only updates manual sections 1, 2, and 3.
OPTIONS
   -n    Do not create /usr/lib/whatis.
   -p    Print what would be done instead of doing it.
   -w    Only create the /usr/lib/whatis database. No manual reformatting is done.
FILES
   /usr/man/man?/*.*      raw (nroff input) manual sections
   /usr/man/cat?/*.*      preformatted manual pages
   /usr/lib/makewhatis     commands to make whatis database
SEE ALSO
   man(1)

Last change: 21 April 1983
Sun Release 1.1
NAME
chown – change owner

SYNOPSIS
/etc/chown -f owner file ...

DESCRIPTION
Chown changes the owner of the files to owner. The owner may be either a decimal UID or a login name found in the password file.

Only the super-user can change owner, in order to simplify as yet unimplemented accounting procedures.

OPTIONS
-f Do not report errors.

FILES
/etc/passwd

SEE ALSO
chgrp(1), chown(2), passwd(5), group(5)
NAME
cr - clear i-node
SYNOPSIS
/etc/cr filesystem i-number ...
DESCRIPTION
N.B.: Cr is obsoleted for normal file system repair work by fack(8).
Cr writes zeros on the i-nodes with the decimal i-numbers on the filesystem. After cr, any
blocks in the affected file will show up as 'missing' in an icheck(8) of the filesystem.
Read and write permission is required on the specified file system device. The i-node becomes
allocatable.
The primary purpose of this routine is to remove a file which for some reason appears in no direc-
tory. If it is used to zap an i-node which does appear in a directory, care should be taken to track
down the entry and remove it. Otherwise, when the i-node is reallocated to some new file, the old
entry will still point to that file. At that point removing the old entry will destroy the new file.
The new entry will again point to an unallocated i-node, so the whole cycle is likely to be
repeated again and again.
SEE ALSO
icheck(8)
BUGS
If the file is open, cr is likely to be ineffective.
NAME
comsat - biff server

SYNOPSIS
/usr/etc/in.comsat

DESCRIPTION
Comsat is the server process which listens for reports of incoming mail and notifies users who have requested to be told when mail arrives. It is invoked as needed by inetd(8C), and times out if inactive for a few minutes.

Comsat listens on a datagram port associated with the "biff" service specification (see services(5)) for one line messages of the form

user@mailbox-offset

If the user specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a "biff y"), the offset is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the "From", "To", "Date", or "Subject" lines are not printed when displaying the message.

FILES
/etc/utmp to find out who's logged on and on what terminals

SEE ALSO
biff(1)

BUGS
The message header filtering is prone to error.

Users should be notified of mail which arrives on other machines than the one they are currently logged in to.

The notification should appear in a separate window so it does not mess up the screen.
NAME
config - build system configuration files

SYNOPSIS
/usr/etc/config [-p] config_file

DESCRIPTION
Config builds a set of system configuration files from a short file which describes the sort of system that is being configured. It also takes as input a file which tells config what files are needed to generate a system. This can be augmented by a configuration-specific set of files that give alternate files for a specific machine. (see the FILES section below) Specifying the -p option to config configures a system for profiling (see kgy_mon(8) and gprof(1)).

Run config from the conf subdirectory of the system source (in a Sun environment, from /usr/conf). Config assumes that there is already a directory ../config_file created and it places all its output files in there. The output of config consists of several files: loconf, which contains a description of I/O devices attached to the system; a makefile, which is a file used by make(1) in building the system; a set of header files which contain the number of various devices that will be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for swapping, the root file system, argument processing, and system dumps.

After running config, you must then change directory to the directory in which the new makefile was created, and use make to create the dependency tree for the new system:

    # ed ../System_Name
    # make depend

Config reminds you of this when it completes.

If you get any other error messages from config, you should fix the problems in your configuration file and try again. If you try to compile a system that had configuration errors, you will probably not succeed.

CONFIG FILE FORMAT

In the following descriptions, a number can be a decimal integer, a whole octal number or a whole hexadecimal number. Hex and octal are specified to config in the same way they are specified to the C compiler, a number starting with "0x" is a hex number and a number starting with just a "0" is an octal number. When specifying the timezone, you may also use floating point numbers.

Comments are specified in a config file with the character "#". All characters from a "#" to the end of a line are ignored.

Lines beginning with tabs are considered continuations of the previous line.

Lines of the configuration file can be one of two basic types. First, there are lines which describe general things about your system:

    machine type
    This is system is to run on the machine type specified. Only one machine type can appear in the config file. The legal type for a Sun system is sun.

    cpu "type"
    This system is to run on the cpu type specified. For a Sun system, "type" is "SUN2" (note that the double quotes are part of the designation).

    ident name
    Gives the system identifier — a name for the machine or machines that run this kernel.
    name must be enclosed in double quotes if it contains both letters and numbers.

    timezone number [ dst ]
    Specifies the timezone you are in. This is measured in the number of hours west of GMT you are. 5 is EST, 8 is PST. Negative numbers indicate hours east of GMT. If you specify dst, the system will operate under daylight savings time.
**maxusers number**

The maximum expected number of simultaneously active user on this system is `number`. This number is used to size several system data structures.

**options optlist**

Compile the listed options into the system. Options in this list are separated by commas. There is a list of options that you may specify in the generic makefile. A line of the form "options FUNNY,HAHA" yields `-DFUNNY -DAHA` to the C compiler. An option may be given a value, by following its name with "==" then the value enclosed in (double) quotes. None of the standard options use such a value.

**config syname config_clauses...**

Generate a system with name `syname` and configuration as specified in `config_clauses`. The `syname` is used to name the resultant binary image and per-system swap configuration files. The `config_clauses` indicate the location for the root file system, one or more disk partitions for swapping and paging, a disk partition to which system dumps should be made, and a disk partition on which `execve` argument lists should be constructed. All but the root device specification may be omitted, `config` will fill them in with default values as described below.

**root**

A root device specification is of the form `root on xy0d`. If a specific partition is omitted — for example, if only `root on xy0` is specified — the "a" partition is assumed. When a generic system is being built, no root specification should be given; the root device will be defined at boot time by prompting the console.

**swap**

Swap device specifications have two possible forms. If a generic swap configuration is required, the clause `swap generic` should be specified. Otherwise, if a single partition is to be used for swapping, one may specify `swap on xy0b`. If multiple partitions are to be interleaved one should specify something of the form `swap on xy0 and xy1 and xy1g`. If no swap specification is given, `config` assumes swapping should be done on the "b" partition of the root device. Swapping areas may be almost any size and multiple swap partitions of varying size may be interleaved. Partitions used for swapping are sized at boot time by the system; to override dynamic sizing of a swap area the number of sectors in the swap area can be specified in the `config` file. For example, `swap on xy0b size 99999` would configure a swap partition with 99999 sectors.

**args**

The partition to be used for argument list processing may be specified with a clause of the form `args on xy1b`. If no argument device is specified, the first swap partition specified is used. If a device is specified without a particular partition, the "b" partition is assumed. If a generic system is being built, no argument device should be specified; the argument device will be assigned to the swap device dynamically configured at boot time.

**dumps**

The location to which system dumps are sent may be specified with a clause of the form `dumps on xy1`. If no dump device is specified, the first swap partition specified is used. If a device is specified without a particular partition, the "b" partition is assumed. If a generic configuration is to be built, no dump device should be specified; the dump device will be assigned to the swap device dynamically configured at boot time.

Dumps are placed at the end of the partition specified. Their size and location is recorded in global kernel variables `dumsize` and `dumplo`, respectively, for use by `savecore(8)`.

Device names specified in configuration clauses are mapped to block device major numbers with a file `/sys/conf/devices.machine`, where `machine` is the machine type previously specified in the configuration file. If a device name to block device major number mapping must be overridden, a device specification may be given in the form `major x minor y`.

Sun Release 1.1

Last change: 6 March 1984

15
The second group of lines in the configuration file describe which devices your system has and what they are connected to (for example, I have a Xylogics xy450 on the Multibus). These lines have the following format:

```
dev_type dev_name at con_dev more_info
```

*Dev_type* is either *tape*, *disk*, *controller*, *device*, or *pseudo-device*. These types have the following meanings:

- **controller**: is a Multibus disk controller or a Multibus tape controller.
- **disk** or **tape**: are devices connected to a controller.
- **device**: is something that plugs into the Multibus, like a cartridge tape interface.
- **pseudo-device**: is something which should be conditionally loaded, but is not really a device. Current examples are the pseudo-tty driver and various network subsystems. (For pseudo-devices, *more_info* may be specified as an integer, that gives the value of the symbol defined in the header file created for that device, and is generally used to indicate the number of instances of the pseudo-device to create. If you load a subsystem you will probably find it convenient to enable conditional code using an *options* specification.

*dev_name* is the name of the device you are specifying. If it is not a pseudo-device, you must give a number afterwards (for example, *xyco* for a Xylogics controller, or *ar0* for an Archive quarter-inch tape controller).

*Con_dev* is what the device you are specifying is connected to. For example, disk *xy1* is connected to controller *xyco*.

*more_info* is a sequence of the following:

- **csr addr**: Specifies the csr (command and status registers) for a device. Must be specified for all Multibus tape and disk controllers and all devices connected to the Multibus. Make certain that you put a leading zero on the address so that it will be interpreted as an octal number.

- **drive number**: For a disk or tape, specifies which drive this is.

- **flags number**: These flags are passed to the device driver at system initialization time.

- **priority level**: For devices which interrupt on the Multibus, specifies the interrupt level at which the device operates.

The easiest way to understand config files is to look at a working one and modify it to suit your system. A good sample is provided in the *Installing UNIX for the First Time* chapter of the *System Installation and Maintenance Guide* at the beginning of this manual. In the example, all lines are explained.

A ? may be substituted for a number in two places and the system will figure out what to fill in for the ? when it boots. You can put question marks on a *con_dev* (e.g. at mba?), or on a drive number (e.g. drive ?) This allows redundancy, as a single system can be built which will reboot on different hardware configurations.
FILES
/sys/conf/Generic generic config file for Sun systems
/sys/conf/README file describing how to make a new kernel
/sys/conf/makefile.sun generic makefile for Sun systems
/sys/conf/files list of common files system is built from
/sys/conf/files.sun list of Sun-specific files
/sys/conf/devices.sun name to major device mapping file for Sun systems

SEE ALSO
The SYNOPSIS portion of each device entry in the section 4 pages of the System Interface Manual.

In the System Manager's Manual:
Kernel Configuration in the Installing UNIX for the First Time chapter.
Building Sun Workstation Kernels in the Tutorials section

BUGS
The line numbers reported in error messages are usually off by one.
The 'make depend' currently generates a few spurious error messages, for example: '..sun/locore.s: no such file or directory'. Ignore these.
NAME
crash - what happens when the system crashes

DESCRIPTION
This section explains what happens when the system crashes and how you can analyze crash
dumps.

When the system crashes voluntarily it prints a message of the form
    panic: why i gave up the ghost
on the console, takes a dump on a mass storage peripheral, and then invokes an automatic reboot
procedure as described in reboot(8). Unless some unexpected inconsistency is encountered in the
state of the file systems due to hardware or software failure the system will then resume multi-
user operations.

The system has a large number of internal consistency checks; if one of these fails, then it will
panic with a very short message indicating which one failed.

The most common cause of system failures is hardware failure, which can reflect itself in different
ways. Here are the messages which you are likely to encounter, with some hints as to causes.
Left unstated in all cases is the possibility that hardware or software error produced the message
in some unexpected way.

IO err in push
hard IO err in swap

The system encountered an error trying to write to the paging device or an error in reading
critical information from a disk drive. You should fix your disk if it is broken or unreliable.

timeout table overflow
This really shouldn't be a panic, but until we fix up the data structure involved, running
out of entries causes a crash. If this happens, you should make the timeout table bigger.

trap type %d, code=%d, pc=%x
A unexpected trap has occurred within the system; the trap types are:

0 reserved addressing fault
1 privileged instruction fault
2 reserved operand fault
3 bpt instruction fault
4 xfc instruction fault
5 system call trap
6 arithmetic trap
7 ast delivery trap
8 segmentation fault
9 protection fault
10 trace trap
11 compatibility mode fault
12 page fault
13 page table fault

The favorite trap types in system crashes are trap types 8 and 9, indicating a wild reference.
The code is the referenced address, and the pc at the time of the fault is printed.
These problems tend to be easy to track down if they are kernel bugs since the processor
stops cold, but random flakiness seems to cause this sometimes.

init dled
The system initialization process has exited. This is bad news, as no new users will then
be able to log in. Rebooting is the only fix, so the system just does it right away.
That completes the list of panic types you are likely to see.

When the system crashes it writes (or at least attempts to write) an image of memory into the back end of the primary swap area. After the system is rebooted, the program `savecore(8)` runs and preserves a copy of this core image and the current system in a specified directory for later perusal. See `savecore(8)` for details.

To analyze a dump you should begin by running `adb(1S)` with the `-k` flag on the core dump. Normally the command `"*(intstack-4)$c"` will provide a stack trace from the point of the crash and this will provide a clue as to what went wrong. A more complete discussion of system debugging is impossible here. See, however, "Using ADB to Debug the UNIX Kernel".

SEE ALSO
adb(1S), analyze(8), reboot(8)

Using ADB to Debug the UNIX Kernel
NAME
cron - clock daemon

SYNOPSIS
/etc/cron

DESCRIPTION
  Cron executes commands at specified dates and times according to the instructions in the file
  /usr/lib/crontab. Since cron never exits, it should only be executed once. This is best done by
  running cron from the initialization process through the file /etc/rc; see init(8).

  /usr/lib/crontab consists of lines of six fields each. The fields are separated by spaces or tabs.
  The first five are integer patterns to specify the minute (0-59), hour (0-23), day of the month (1-31),
  month of the year (1-12), and day of the week (1-7 with 1=Monday). Each of these patterns
  may contain a number in the range above; two numbers separated by a dash meaning a range
  inclusive; a list of numbers separated by commas meaning any of the numbers; or an asterisk
  meaning all legal values. The sixth field is a string that is executed by the shell at the specified
  times. A percent character in this field is translated to a new-line character.

  Only the first line (up to a % or end of line) of the command field is executed by the shell. The other lines are
  made available to the command as standard input.

Here are a few example lines from /usr/lib/crontab, to give you a better sense of the file’s format:

  0 0 ** * calendar -
  15 0 * * * /usr/etc/sa -s >/dev/null
  0,30 * * * /usr/etc/dmesg - > > /usr/adm/messages
  15 4 * * * find /usr/preserve -mtime +7 -a -exec rm -f {}
  10 4 * * * egrep 'SYSERR|refused|unreachable' /usr/spool/log/syslog.O |mail Postmaster

  Cron examines /usr/lib/crontab under the following conditions:
  • At least once per hour (on the hour).
  • When the next command is to be run — cron looks ahead until the next command and sleeps
    until then.
  • When cron’s process is sent a SIGHUP. This means that someone who changes /usr/lib/crontab
    can get cron to look at it right away.

You can also create the /usr/adm/cronlog file, if you wish. If the file exists, cron logs to it each
  time it executes an instruction from /usr/lib/crontab. You can thus use the cronlog file to make
  sure cron is running properly. The lines in the file consist of a timestamp, and process statement.
  Lines look something like this:

  Thu Mar  8 10:20:00 1984: /etc/dmesg - > > /usr/adm/messages
  Thu Mar  8 10:29:59 1984: /etc/atrun
  Thu Mar  8 10:30:00 1984: /etc/dmesg - > > /usr/adm/messages
  Thu Mar  8 10:39:59 1984: /etc/dmesg - > > /usr/adm/messages

FILES
  /usr/lib/crontab            Instruction file
  /usr/adm/cronlog            Log file

SEE ALSO
  crontab(5)

20 Last change: 8 March 1984 Sun Release 1.1
NAME
dcheck – file system directory consistency check

SYNOPSIS
/usr/ete/dcheck [-l numbers] [ filesystem ]

DESCRIPTION
N.B. Dcheck is obsoleted for normal consistency checking by fseq(8).

Dcheck reads the directories in a file system and compares the link-count in each i-node with the
number of directory entries by which it is referenced. If the file system is not specified, dcheck
checks a set of default file systems.

Dcheck is fastest if the raw version of the special file is used, since the i-list is read in large
chunks.

OPTIONS
-l numbers

Numbers is a list of i-numbers; when one of those i-numbers turns up in a directory, the
number, the i-number of the directory, and the name of the entry are reported.

FILES
Default file systems vary with installation.

SEE ALSO
fseq(8), icheck(8), fs(5), elli(8), ncheck(8)

DIAGNOSTICS
When a file turns up for which the link-count and the number of directory entries disagree, the
relevant facts are reported. Allocated files which have 0 link-count and no entries are also listed.
The only dangerous situation occurs when there are more entries than links; if entries are
removed, so the link-count drops to 0, the remaining entries point to thin air. They should be
removed. When there are more links than entries, or there is an allocated file with neither links
nor entries, some disk space may be lost but the situation will not degenerate.

BUGS
Since dcheck is inherently two-pass in nature, extraneous diagnostics may be produced if applied
to active file systems.

Dcheck is obsoleted by fseq and remains for historical reasons.
NAME
diag - General-purpose stand-alone utility package

SYNOPSIS
b stand/diag

DESCRIPTION
Diag is a general-purpose stand-alone utility package containing a grab-bag of the kinds of tools needed for disk initialization, testing, and file transfer. Diag supports the various SMD and ST-506 disk controllers.

Note: that diag can only be called from the Sun Workstation PROM monitor — diag is not a system utility.

The most common use of diag is formatting and labelling a disk — see the format, label, and partition commands.

Diag is interactive — it prompts for options and arguments. There are two phases to using diag: in the first phase, diag prompts for information about the type of disk drive and controller that it is working with, and essentially 'configures' itself to work with that disk and controller. At the end of this phase, diag tries to access the disk controller you have defined. If the attempt succeeds, diag gives you a status report on the disk and gives you a 'diag>' prompt; this signals the beginning of diag's second phase. If the attempt to access the controller fails (if the controller is mis-defined or non-existent, for example), you get a bus error message, and return to the PROM monitor.

During diag's second phase, you can use the commands listed below in response to the 'diag>' prompt. A specific diag command is called up by typing enough characters to uniquely identify the command. The commands that diag currently recognizes are:

clear Sends a restore command to a disk. This is needed to manually reset disk errors.
diag Re-initializes the diag program itself — goes back to phase one of the initialization process described above.
errors Toggles an option to report all errors as they occur.
fix Formats a single track (Interphase 2180 controller) or sector (Xylogics SMD controller) of a disk. This sub-command is not available for use with Adaptec disk controllers (ST-506 interface). If the track/sector is already formatted, you are asked for confirmation in order to avoid destroying the data on the track/sector.
format Formats a disk.
help or ? Displays a list of the available commands.
info Toggles an option to report all disk activity as it completes.
label Labels the disk.
map Explicitly maps one track or sector to a different track or sector. Usually required for bad track mapping. The format command usually does this automatically. Note that, like the the fix command above, map is disk controller-dependent: you can map tracks with an Interphase controller, and sectors with Xylogics controllers. map is not available for use with Adaptec disk controllers (ST-506 interface).
partition Creates, assigns, or modifies logical partition tables for a disk. The UNIX operating system requires logical partitions. The label command writes the partition map to the disk. There are standard partition tables for each type of disk that diag knows about.
position Tests the disk by reading sectors from random positions on the disk. This command runs until the user aborts it by typing the letter 'a'.
quit Quits from diag and returns to whoever called it up.
**read**  Reads specified blocks from the disk. The `read` command prompts for the starting block number, number of blocks, and the block increment. The `read` command doesn't report the data it reads — it is intended for verifying that blocks are readable.

**seek**  Performs a seek test on the disk such that a seek is made to every cylinder and a seek is made to every possible cylinder distance.

**status**  Reports the ready status of each drive on the current controller.

**test**  Does a general disk test by writing random data to random positions on the disk and then verifying that the correct data can be read back. The `test` command destroys data on the disk. It runs until the user aborts the process by typing the letter 'a'.

**time**  Toggles an option to turn timing on and off. When timing is on, `diag` reports on how long things take — `diag` is less verbose in this state so it doesn't waste time displaying messages.

**verify**  Reads and displays the label from the disk. Shows the logical partition assignments. This is usually done automatically when the `format` command has labelled the disk.

**write**  Writes garbage data to specified blocks on the disk. The `write` command prompts for the starting block number, number of blocks, and the block increment. The `write` is intended for verifying that blocks are writeable.

**+**  Adds two numbers and reports the result in decimal, hexadecimal, and as a disk address.

**-**  Subtracts two numbers and reports the result in decimal, hexadecimal, and as a disk address.

Block numbers may be entered either as an absolute decimal block number, or as a disk address of the form cylinder/head/sector.

Any `diag` command may be aborted by typing a `^C` (control-C).
NAME
dmesg – collect system diagnostic messages to form error log

SYNOPSIS
/usr/etc/dmesg [- ]

DESCRIPTION

Dmesg looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed by the system when device (hardware) errors occur and (occasionally) when system tables overflow non-fatally. If the – flag is given, then dmesg computes (incrementally) the new messages since the last time it was run and places these on the standard output. This is typically used with cron(8) to produce the error log /usr/adm/messages by running the command
/etc/dmesg – > > /usr/adm/messages
every 10 minutes.

FILES
/usr/adm/messages error log (conventional location)
/usr/adm/msgbuf scratch file for memory of – option

BUGS

The system error message buffer is of small finite size. As dmesg is run only every few minutes, not all error messages are guaranteed to be logged. This can be construed as a blessing rather than a curse.

Error diagnostics generated immediately before a system crash will never get logged.
NAME
dump – incremental file system dump

SYNOPSIS
/etc/dump [ key [ argument ... ] filesystem ]

DESCRIPTION
Dump copies to magnetic tape all files changed after a certain date in the filesystem. The key specifies the date and other options about the dump. Key consists of characters from the set 0123456789.fusWnec.

0–9 This number is the 'dump level'. All files modified since the last date stored in the file /etc/dumpdates for the same filesystem at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed; thus the entire filesystem is dumped if the 0 option is used.

f Place the dump on the next argument file instead of the tape. If file is of the form machine:device, dump to a remote machine.

u If the dump completes successfully, write the date of the beginning of the dump on file /etc/dumpdates. This file records a separate date for each filesystem and each dump level. The format of /etc/dumpdates is readable by people, consisting of one free format record per line: filesystem name, increment level and ctime(9) format dump date. /etc/dumpdates may be edited to change any of the fields, if necessary.

s The size of the dump tape is specified in feet. The number of feet is taken from the next argument. When the specified size is reached, dump waits for reels to be changed. The default tape size is 2300 feet.

d The density of the tape, expressed in BPI, is taken from the next argument. This is used in calculating the amount of tape used per reel. The default is 1600.

W Dump tells the operator what file systems need to be dumped. This information is gleaned from the files /etc/dumpdates and /etc/fstab. The W option causes dump to print out, for each file system in /etc/dumpdates the most recent dump date and level, and highlights those file systems that should be dumped. If the W option is set, all other options are ignored, and dump exits immediately.

w Is like W, but prints only those filesystems which need to be dumped.

n Whenever dump requires operator attention, notify by means similar to a wall(1) all of the operators in the group "operator".

c Dump to cartridge tape instead of standard half-inch magnetic tape.

b Specifies the blocking factor for the dump. The blocking factor is taken from the next argument on the command line. The default blocking factor is 10.

If no arguments are given, the key is assumed to be 0u and a default file system is dumped to the default tape.

Dump requires operator intervention on these conditions: end of tape, end of dump, tape write error, tape open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the n key, dump interacts with the operator on dump's control terminal at times when dump can no longer proceed, or if something is grossly wrong. All questions dump poses must be answered by typing "yes" or "no", appropriately.

Since making a dump involves a lot of time and effort for full dumps, dump checkpoints itself at the start of each tape volume. If writing that volume fails for some reason, dump will, with operator permission, restart itself from the checkpoint after the old tape has been rewound and removed, and a new tape has been mounted.
Dump tells the operator what is going on at periodic intervals, including usually low estimates of
the number of blocks to write, the number of tapes it will take, the time to completion, and the
time to the tape change. The output is verbose, so that others know that the terminal controlling
dump is busy, and will be for some time.

Now a short suggestion on how to perform dumps. Start with a full level 0 dump

dump 0

Next, dumps of active file systems are taken on a daily basis, using a modified Tower of Hanoi
algorithm, with this sequence of dump levels:

3 2 5 4 7 6 9 8 9 9 ...

For the daily dumps, a set of 10 tapes per dumped file system is used on a cyclical basis. Each
week, a level 1 dump is taken, and the daily Hanoi sequence repeats with 3. For weekly dumps, a
set of 5 tapes per dumped file system is used, also on a cyclical basis. Each month, a level 0
dump is taken on a set of fresh tapes that is saved forever.

FILES
/dev/rrplg  default filesystem to dump from
/dev/rmt8   default tape unit to dump to
/etc/dumpdates new format dump date record
/etc/fstab   dump table: file systems and frequency
/etc/group   to find group operator

SEE ALSO
restore(8), dump(5), fstab(5)

DIAGNOSTICS
Many, and verbose.

BUGS
Sizes are based on 1600 BPI blocked tape; the raw magtape device has to be used to approach
these densities. Fewer than 32 read errors on the filesystem are ignored. Each reel requires a new
process, so parent processes for reels already written just hang around until the entire tape is
written.

It would be nice if dump knew about the dump sequence, kept track of the tapes scribbled on,
told the operator which tape to mount when, and provided more assistance for the operator run-
nning restore.
NAME
dumpfs – dump file system information

SYNOPSIS
/usr/etc/dumpfs device

DESCRIPTION
Dumpfs prints out the super block and cylinder group information for the file system or special
device specified. The listing is very long and detailed. This command is useful mostly for finding
out certain file system information such as the file system block size and minimum free space per-
ceptage.

SEE ALSO
fs(5), tunefs(8), newfs(8), fsck(8)
NAME
expire - remove outdated news articles

SYNOPSIS

DESCRIPTION
Expire is normally started up by cron(8) every night to remove all expired news. If no newsgroups are specified, the default is to expire all.

Articles whose specified expiration date has already passed are considered expirable.

OPTIONS
-n newsgroups
List of newsgroups to expire. Elements in the list of newsgroups must be separated by commas and/or spaces.

-a
archive articles in /usr/spool/oldnews. Articles are unlinked in the absence of the -a option.

-v[level]
be more verbose. A verbosity level (default 1) can follow the -v flag, as in -v3 for even more output. This is useful if articles aren’t being expired and you want to know why.

-e days
give the number of days to use for a default expiration date. If not given, an installation dependent default (often 2 weeks) is used. Note that you must use a space between -e and days.

-l
-I
ignore any expiration date explicitly given on articles. This can be used when disk space is really tight. -I always ignores expiration dates, while -l only ignores the date if ignoring it would expire the article sooner. WARNING: If you have articles archived by giving them expiration dates far into the future, these options might remove these files anyway.

-r
rebuild the history file without removing any files. In the process, expire formats the dbm(3) format files associated with the history file.

-h
expire articles without using the history file. Both the -r and -h flags use the active file for newsgroup information rather than the history file.

SEE ALSO
checknews(1), inews(1), readnews(1), recnews(8), sendnews(8), uurec(8)
NAME
fastboot, fasthalt - reboot/halt the system without checking the disks

SYNOPSIS
/etc/fastboot [ boot-options ]
/etc/fasthalt [ halt-options ]

DESCRIPTION
Fastboot and fasthalt are shell scripts which reboot and halt the system without checking the file systems. This is done by creating a file /fastboot, then invoking the reboot program. The system startup script, /etc/rc, looks for this file and, if present, skips the normal invocation of fsck(8).

SEE ALSO
halt(8), reboot(8), rc(8)
NAME

fsck - file system consistency check and interactive repair

SYNOPSIS

/etc/fsck -p [ filesystem ... ]
/etc/fsck [ -b block# ] [ -y ] [ -n ] [ filesystem ] ...

DESCRIPTION

The first form of fsck preens a standard set of file systems or the specified file systems. It is normally used in the script /etc/rc during automatic reboot. In this case fsck reads the table /etc/fstab to determine which file systems to check. It uses the information there to inspect groups of disks in parallel taking maximum advantage of i/o overlap to check the file systems as quickly as possible. Normally, the root file system will be checked on pass 1, other "root" ("a" partition) file systems on pass 2, other small file systems on separate passes (e.g. the "d" file systems on pass 3 and the "e" file systems on pass 4), and finally the large user file systems on the last pass, e.g. pass 5. A pass number of 0 in fstab causes a disk to not be checked; similarly partitions which are not shown as to be mounted "rw" or "ro" are not checked.

The system takes care that only a restricted class of innocuous inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

- Unreferenced inodes
- Link counts in inodes too large
- Missing blocks in the free list
- Blocks in the free list also in files
- Counts in the super-block wrong

These are the only inconsistencies which fsck with the -p option will correct; if it encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines will be printed identifying the file system on which the correction will take place, and the nature or the correction. After successfully correcting a file system, fsck will print the number of files on that file system and the number of used and free blocks.

Without the -p option, fsck audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that a number of the corrective actions which are not fixable under the -p option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond yes or no. If the operator does not have write permission fsck will default to a -n action.

Fsck has more consistency checks than its predecessors check, dcheck, fcheck, and icheck combined.

The following flags are interpreted by fsck.

- Use the block specified immediately after the flag as the super block for the file system. Block 32 is always an alternate super block.
- Assume a yes response to all questions asked by fsck; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.
- Assume a no response to all questions asked by fsck; do not open the file system for writing.

If no file systems are given to fsck then a default list of file systems is read from the file /etc/fstab.

Last change: 4 February 1983

Sun Release 1.1
Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   - Directory size not of proper format.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   - File pointing to unallocated inode.
   - Inode number out of range.
8. Super Block checks:
   - More blocks for inodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator’s concurrence, reconnected by placing them in the lost+found directory. The name assigned is the inode number. The only restriction is that the directory lost+found must preexist in the root of the filesystem being checked and must have empty slots in which entries can be made. This is accomplished by making lost+found, copying a number of files to the directory, and then removing them (before fsck is executed).

Checking the raw device is almost always faster.

FILES
/etc/fstab contains default list of file systems to check.

DIAGNOSTICS
The diagnostics produced by fsck are intended to be self-explanatory.

SEE ALSO
fstab(5), fs(5), newfs(8), mkfs(8), crash(8S), reboot(8)

BUGS
Inode numbers for . and .. in each directory should be checked for validity.
There should be some way to start a fsck -p at pass n.
NAME
ftpd - DARPA Internet File Transfer Protocol server

SYNOPSIS
/usr/etc/in.ftpd host.socket

DESCRIPTION
Ftpd is the DARPA Internet File Transfer Protocol server process. The server is invoked by
the Internet daemon inetd(8C) each time a connection to the ftp service (see services(5)) is made,
with the connection available as descriptor 0 and the host and socket the connection originated
from (in hex and decimal respectively) as argument.

Inactive connections are timed out after 60 seconds.

The ftp server currently supports the following ftp requests; case is not distinguished.

<table>
<thead>
<tr>
<th>Request</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT</td>
<td>specify account (ignored)</td>
</tr>
<tr>
<td>ALLO</td>
<td>allocate storage (vacuously)</td>
</tr>
<tr>
<td>APPE</td>
<td>append to a file</td>
</tr>
<tr>
<td>CWD</td>
<td>change working directory</td>
</tr>
<tr>
<td>DELE</td>
<td>delete a file</td>
</tr>
<tr>
<td>HELP</td>
<td>give help information</td>
</tr>
<tr>
<td>LIST</td>
<td>give list files in a directory (&quot;ls -lg&quot;)</td>
</tr>
<tr>
<td>MODE</td>
<td>specify data transfer mode</td>
</tr>
<tr>
<td>NLST</td>
<td>give name list of files in directory (&quot;ls&quot;)</td>
</tr>
<tr>
<td>NOOP</td>
<td>do nothing</td>
</tr>
<tr>
<td>PASS</td>
<td>specify password</td>
</tr>
<tr>
<td>PORT</td>
<td>specify data connection port</td>
</tr>
<tr>
<td>QUIT</td>
<td>terminate session</td>
</tr>
<tr>
<td>RETR</td>
<td>retrieve a file</td>
</tr>
<tr>
<td>RNFR</td>
<td>specify rename-from file name</td>
</tr>
<tr>
<td>RNTO</td>
<td>specify rename-to file name</td>
</tr>
<tr>
<td>STOR</td>
<td>store a file</td>
</tr>
<tr>
<td>STRU</td>
<td>specify data transfer structure</td>
</tr>
<tr>
<td>TYPE</td>
<td>specify data transfer type</td>
</tr>
<tr>
<td>USER</td>
<td>specify user name</td>
</tr>
<tr>
<td>XCUP</td>
<td>change to parent of current working directory</td>
</tr>
<tr>
<td>XCWD</td>
<td>change working directory</td>
</tr>
<tr>
<td>XMKD</td>
<td>make a directory</td>
</tr>
<tr>
<td>XPWD</td>
<td>print the current working directory</td>
</tr>
<tr>
<td>XRMD</td>
<td>remove a directory</td>
</tr>
</tbody>
</table>

The remaining ftp requests specified in Internet RFC 765 are recognized, but not implemented.

Ftpd interprets file names according to the "globbing" conventions used by cdh(1). This allows
users to utilize the metacharacters "[*?[]"]".

Ftpd authenticates users according to three rules.

1) The user name must be in the password data base, /etc/passwd, and not have a null pass-
word. In this case a password must be provided by the client before any file operations
may be performed.

2) The user name must not appear in the file /etc/ftpusers.

3) If the user name is "anonymous" or "ftp", an anonymous ftp account must be present in
the password file (user "ftp"). In this case the user is allowed to log in by specifying any
password (by convention this is given as the client host's name).
In the last case, *ftpd* takes special measures to restrict the client's access privileges. The server performs a `chroot(2)` command to the home directory of the "ftp" user. In order that system security is not breached, it is recommended that the "ftp" subtree be constructed with care; the following rules are recommended.

- **ftp** Make the home directory owned by "ftp" and unwritable by anyone.
- **ftp/bin**
  Make this directory owned by the super-user and unwritable by anyone. The program `ls(1)` must be present to support the list commands. This program should have mode 111.
- **ftp/etc**
  Make this directory owned by the super-user and unwritable by anyone. The files `passwd(5)` and `group(5)` must be present for the `ls` command to work properly. These files should be mode 444.
- **ftp/pub**
  Make this directory mode 777 and owned by "ftp". Users should then place files which are to be accessible via the anonymous account in this directory.

**SEE ALSO**

`ftp(1C)`

**BUGS**

There is no support for aborting commands.

The anonymous account is inherently dangerous and should avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.
NAME
gettable - get NIC format host tables from a host

SYNOPSIS
/etc/gettable host

DESCRIPTION
Gettable is a simple program used to obtain the NIC standard host tables from a "nicname" server. The indicated host is queried for the tables. The tables, if retrieved, are placed in the file hosts.txt.

Gettable operates by opening a TCP connection to the port indicated in the service specification for "nicname". A request is then made for "ALL" names and the resultant information is placed in the output file.

Gettable is best used in conjunction with the htable(8) program which converts the NIC standard file format to that used by the network library lookup routines.

SEE ALSO
intro(3N), htable(8)

BUGS
Should allow requests for only part of the database.
NAME
  getty - set terminal mode

SYNOPSIS
  /etc/getty [ char ]

DESCRIPTION
  Getty is invoked by init(8) immediately after a terminal is opened, following the making of a connection. While reading the name getty attempts to adapt the system to the speed and type of terminal being used.

  Init calls getty with an argument specified by the ttys file entry for the terminal line. Arguments other than '0' can be used to make getty treat the line specially. Refer to ttys(5) and gettytab(5) for descriptions of how getty uses the data in gettytab to set up the terminal. Normally, it sets the speed of the interface to 300 baud, specifies that raw mode is to be used (break on every character), that echo is to be suppressed, and either parity allowed. It types a banner identifying the system (from gethostname(2)) and the 'login:' message. Then the user's name is read, a character at a time. If a null character is received, it is assumed to be the result of the user pushing the 'break' ('interrupt') key. The speed is then changed to 1200 baud and the 'login:' is typed again; a second 'break' changes the speed to 150 baud and the 'login:' is typed again. Successive 'break' characters cycle through the speeds 300, 1200, and 150 baud.

  The user's name is terminated by a new-line or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see tty(4)).

  The user's name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is nonempty, the system is told to map any future upper-case characters into the corresponding lower-case characters.

  Finally, login is called with the user's name as argument.

SEE ALSO
  init(8), login(1), ioctl(2), tty(4), ttys(5), gettytab(5)
NAME
gxtest – stand alone test for the Sun video graphics board

SYNOPSIS
   b /stand/gxtest

DESCRIPTION
Gxtest runs stand alone, not under control of the UNIX operating system. With the PROM
resident monitor in control of the system, type the command:

   > b /stand/gxtest

and the monitor boots the video test program into memory. Gxtest is completely self-explanatory
and runs under its own steam. It reports any errors it finds on the screen.
NAME
halt - stop the processor

SYNOPSIS
/etc/halt [-n] [-q] [-y]

DESCRIPTION
Halt writes out sandbagged information to the disks and then stops the processor.

OPTIONS
- n  Prevents the sync before stopping.
- q  Do a quick halt, no graceful shutdown is attempted.
- y  Needed if you are trying to halt the system from a dialup.

SEE ALSO
reboot(8), shutdown(8)
NAME
htable - convert NIC standard format host tables

SYNOPSIS
/usr/etc/htable file

DESCRIPTION
Htable is used to convert host files in the format specified in Internet RFC 810 to the format used by the network library routines. Three files are created as a result of running htable: hosts, networks, and gateways. The hosts file is used by the gethostent(3N) routines in mapping host names to addresses. The networks file is used by the getnetent(3N) routines in mapping network names to numbers. The gateways file is used by the routing daemon in identifying "passive" Internet gateways; see routed(8C) for an explanation.

If any of the files localhosts, localnetworks, or localgateways are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

Htable is best used in conjunction with the gettable(8C) program which retrieves the NIC database from a host.

SEE ALSO
intro(3N), gettable(8C)

BUGS
Does not properly calculate the gateways file.
NAME
icheck - file system storage consistency check

SYNOPSIS
/usr/etc/icheck [-s] [-b numbers] [filesystem]

DESCRIPTION
N.B.: Icheck is obsoleted for normal consistency checking by fsck(8).
Icheck examines a file system, builds a bit map of used blocks, and compares this bit map against
the free list maintained on the file system. If the file system is not specified, a set of default file
systems is checked. The normal output of icheck includes a report of

- The total number of files and the numbers of regular, directory, block special and character special files.
- The total number of blocks in use and the numbers of single-, double-, and triple-indirect blocks and directory blocks.
- The number of free blocks.
- The number of blocks missing; i.e. not in any file nor in the free list.

The -s option causes icheck to ignore the actual free list and reconstruct a new one by rewriting
the super-block of the file system. The file system should be dismounted while this is done; if this
is not possible (for example if the root file system has to be salvaged) care should be taken that
the system is quiescent and that it is rebooted immediately afterwards so that the old, bad in-core
copy of the super-block will not continue to be used. Notice also that the words in the super-
block which indicate the size of the free list and of the i-list are believed. If the super-block has
been curdled these words will have to be patched. The -s option causes the normal output
reports to be suppressed.

Following the -b option is a list of block numbers; whenever any of the named blocks turns up in
a file, a diagnostic is produced.

Icheck is faster if the raw version of the special file is used, since it reads the i-list many blocks at
a time.

FILES
Default file systems vary with installation.

SEE ALSO
fsck(8), dcheck(8), ncheck(8), fs(5), clri(8)

DIAGNOSTICS
For duplicate blocks and bad blocks (which lie outside the file system) icheck announces the
difficulty, the i-number, and the kind of block involved. If a read error is encountered, the block
number of the bad block is printed and icheck considers it to contain 0. 'Bad freeblock' means
that a block number outside the available space was encountered in the free list. 'n dups in free'
means that n blocks were found in the free list which duplicate blocks either in some file or in the
earlier part of the free list.

BUGS
Since icheck is inherently two-pass in nature, extraneous diagnostics may be produced if applied
to active file systems.
It believes even preposterous super-blocks and consequently can get core images.
The system should be fixed so that the reboot after fixing the root file system is not necessary.
NAME
ifconfig - configure network interface parameters

SYNOPSIS
/etc/ifconfig interface [ Ethernet_address ] [ hostname ] [ parameters ]

DESCRIPTION
Ifconfig is used to assign an address to a network interface and/or to configure network interface parameters. Ifconfig must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address. Used without options, ifconfig displays the current configuration for a network interface. Only the super-user may modify the configuration of a network interface.

The Interface parameter is a string of the form "name unit", for example "ie0".

OPTIONS
Ethernet_address is the hardware Ethernet address of a given machine. The address is a 6-byte hexadecimal value; each byte of the address is separated by a colon. A typical Ethernet address is '8:0:20:1:1:A3'. This address is contained in the ID PROM on the Sun-2 CPU Board, and is reported at boot time as one of the PROM monitor's sign-on messages. The Ethernet_address option is normally not used — the hardware supplies the default; the option is used only when trying to talk to a device which does not support ARP (like a VAX on a 4.1c network).

hostname may be either the hostname of a given machine (present in the hostname database, hosts(5)), or the complete Internet address consisting of your system's network number and the machine's unique host number. A typical Internet address might be "192.9.200.44", where "192.9.200" is the network number and "44" is the machine's hostnumber. To find a machine's Internet address, consult the local /etc/hosts file.

The following parameters may be set with ifconfig:

up Mark an interface "up".
down Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface.
trailers Enable the use of a "trailer" link level encapsulation when sending (default). If a network interface supports trailers, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver.
-trailers Disable the use of a "trailer" link level encapsulation.
arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10Mb/s Ethernet addresses.
-arp Disable the use of the Address Resolution Protocol.

DIAGNOSTICS
Messages indicating the specified interface does not exist, the requested address is unknown, the user is not privileged and tried to alter an interface's configuration.

SEE ALSO
rc(8), intro(4N), netstat(1)
NAME

imemtest - stand alone memory test

SYNOPSIS

b /stand/imemtest

DESCRIPTION

Imemtest runs stand alone, not under control of the UNIX operating system. With the PROM resident monitor in control of the system, type the command:

> b /stand/imemtest

and the monitor boots the memory test program into memory. Imemtest is completely self-explanatory. It prompts for all start and end addresses, and after that it runs under its own steam. It reports any errors it finds on the screen.
NAME
inetd - internet services daemon

SYNOPSIS
/etc/inetd [-d]

DESCRIPTION
Inetd is the internet super-server which invokes all internet server processes as needed. Connection-oriented services are invoked each time a connection is made, by creating a process. This process is passed the connection as file descriptor 0 and an argument of the form "sourcehost.ssourceport" where sourcehost is hex and sourceport is decimal.

Datagram oriented services are invoked when a datagram arrives; a process is created and passed the connection as file descriptor 0. Inetd will look at the socket where datagrams arrive again only after this process completes. The paradigms for such processes are either to read off the incoming datagram and then fork and exit, or to process the arriving datagram and then time out.

Inetd consults servers(5) when it is invoked, and supports whatever services are in that file.

OPTIONS
-d Specifies that debugging traces are to be turned on for connection-oriented (TCP) services.

FILES
/etc/servers list of internet server processes

SEE ALSO
servers(5), comsat(8C), ftpd(8C), rexec(8C), rlogin(8C), syslog(8), rsh(8C), talkd(8C), telnetd(8C), tftp(8C)

BUGS
There is no provision for selectively invoking TCP debugging packet tracing per-service. Should reread the servers(5) file on receipt of a SIGHUP signal.
NAME
  init – process control initialization

SYNOPSIS
  /etc/init

DESCRIPTION
  Init is invoked inside UNIX as the last step in the boot procedure. It normally then runs the
  automatic reboot sequence as described in reboot(8), and if this succeeds, begins multi-user opera-
  tion. If the reboot fails, it commences single user operation by giving the super-user a shell on the
  console. It is possible to pass parameters from the boot program to init so that single user opera-
  tion is commenced immediately. When such single-user operation is terminated by killing the
  single-user shell (i.e. by hitting 'D), init runs /etc/rc without the reboot parameter. This com-
  mand file performs housekeeping operations such as removing temporary files, mounting file sys-
  tems, and starting daemons.

In multi-user operation, init’s role is to create a process for each terminal port on which a user
may log in. To begin such operations, it reads the file /etc/ttys and forks several times to create
a process for each terminal specified in the file. Each of these processes opens the appropriate ter-
minal for reading and writing. These channels thus receive file descriptors 0, 1 and 2, the stan-
dard input and output and the diagnostic output. Opening the terminal will usually involve a
delay, since the open is not completed until someone is dialed up and carrier established on the
channel. If a terminal exists but an error occurs when trying to open the terminal init complains
by writing a message to the system console; the message is repeated every 10 minutes for each
such terminal until the terminal is shut off in /etc/ttys and init notified (by a hangup, as
described below), or the terminal becomes accessible (init checks again every minute). After an
open succeeds, /etc/getty is called with argument as specified by the second character of the ttys
file line. Getty reads the user’s name and invokes login to log in the user and execute the Shell.

Ultimately the Shell will terminate because of an end-of-file either typed explicitly or generated as
a result of hanging up. The main path of init, which has been waiting for such an event, wakes up
and removes the appropriate entry from the file utmp, which records current users, and makes
an entry in /usr/adm/wtmp, which maintains a history of logins and logouts. The wtmp entry is
made only if a user logged in successfully on the line. Then the appropriate terminal is reopened
and getty is reinvoked.

Init catches the hangup signal (signal SIGHUP) and interprets it to mean that the file /etc/ttys
should be read again. The Shell process on each line which used to be active in ttys but is no
longer there is terminated; a new process is created for each added line; lines unchanged in the file
are undisturbed. Thus it is possible to drop or add phone lines without rebooting the system by
changing the ttys file and sending a hangup signal to the init process: use ‘kill –HUP 1.’

Init will terminate multi-user operations and resume single-user mode if sent a terminate (TERM)
signal, i.e. “kill –TERM 1”. If there are processes outstanding which are deadlocked (due to
hardware or software failure), init will not wait for them all to die (which might take forever), but
will time out after 30 seconds and print a warning message.

Init will cease creating new getty’s and allow the system to slowly die away, if it is sent a terminal
stop (TSTOP) signal, i.e. “kill –TSTOP 1”. A later hangup will resume full multi-user operations, or
a terminate will initiate a single user shell. This hook is used by reboot(8) and halt(8).

Init’s role is so critical that if it dies, the system will reboot itself automatically. If, at bootstrap
time, the init process cannot be located, the system will loop in user mode at location 0x13.

DIAGNOSTICS
  init: ttys cannot open. A terminal which is turned on in the rc file cannot be opened, likely
  because the requisite lines are either not configured into the system or the associated device was
  not attached during boot-time system configuration.
WARNING: Something is hung (wont die); ps axl advised. A process is hung and could not be killed when the system was shutting down. This is usually caused by a process which is stuck in a device driver due to a persistent device error condition.

FILES
/dev/console, /dev/tty*, /etc/utmp, /usr/adm/wtmp, /etc/ttys, /etc/rc

SEE ALSO
login(1), kill(1), sht(5), getty(8), rc(8), reboot(8), halt(8), shutdown(8)
NAME
iostat - report I/O statistics

SYNOPSIS
iostat [ interval [ count ] ]

DESCRIPTION
iostat iteratively reports the number of characters read and written to terminals, and, for each
disk, the number of seeks and transfers per second, and the milliseconds per average seek. It also
gives the percentage of time the system has spent in user mode, in user mode running low priority
(niced) processes, in system mode, and idling.

To compute this information, for each disk, seeks and data transfer completions and number of
words transferred are counted; for terminals collectively, the number of input and output charac-
ters are counted. Also, each sixtieth of a second, the state of each disk is examined and a tally is
made if the disk is active. From these numbers and given the transfer rates of the devices it is
possible to determine average seek times for each device.

The optional interval argument causes iostat to report once each interval seconds. The first report
is for all time since a reboot and each subsequent report is for the last interval only.

The optional count argument restricts the number of reports.

FILES
/dev/kmem
/vmunix

SEE ALSO
vmstat(8)
NAME
kgmon - generate a dump of the operating system's profile buffers

SYNOPSIS
/usr/etc/kgmon [-b] [-h] [-r] [-p] [system] [memory]

DESCRIPTION
Kgmon is a tool used when profiling the operating system. When no arguments are supplied, kgmon indicates the state of operating system profiling as running, off, or not configured (see config(8)). If the -p flag is specified, kgmon extracts profile data from the operating system and produces a gmon.out file suitable for later analysis by gprof(1).

OPTIONS
- b Resume the collection of profile data.
- h Stop the collection of profile data.
- p Dump the contents of the profile buffers into a gmon.out file.
- r Reset all the profile buffers. If the -p flag is also specified, the gmon.out file is generated before the buffers are reset.

If neither -b nor -h is specified, the state of profiling collection remains unchanged. For example, if the -p flag is specified and profile data is being collected, profiling is momentarily suspended, the operating system profile buffers are dumped, and profiling is immediately resumed.

FILES
/vmunix - the default system
/dev/kmem - the default memory

SEE ALSO
gprof(1), config(8)

DIAGNOSTICS
Users with only read permission on /dev/kmem cannot change the state of profiling collection. They can get a gmon.out file with the warning that the data may be inconsistent if profiling is in progress.
NAME
lpcline printer control program

SYNOPSIS
/usr/etclpc [command [argument ...]]

DESCRIPTION
Lpc is used by the system administrator to control the operation of the line printer system. For each line printer configured in /etc/printcap, lpc may be used to:

- disable or enable a printer,
- disable or enable a printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status of printers, and their associated spooling queues and printer daemons.

Without any arguments, lpc will prompt for commands from the standard input. If arguments are supplied, lpc interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected so that lpc reads commands from a file. Commands may be abbreviated; the following is the list of recognized commands.

? [command ...]
help [command ...]
   Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort {all | printer ...}
   Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by lpr) for the specified printers.

clean {all | printer ...}
   Remove all files beginning with "cf", "tf", or "df" from the specified printer queue(s) on the local machine.

enable {all | printer ...}
   Enable spooling on the local queue for the listed printers. This will allow lpr to put new jobs in the spool queue.

exit
quit
   Exit from lpc.

disable {all | printer ...}
   Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by lpr.

restart {all | printer ...}
   Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. Lpq will report that there is no daemon present when this condition occurs.

start {all | printer ...}
   Enable printing and start a spooling daemon for the listed printers.

status [all | [printer ...]]
   Display the status of daemons and queues on the local machine.

stop {all | printer ...}
   Stop a spooling daemon after the current job completes and disable printing.
FILES
/etc/printcap  printer description file
/usr/spool/*  spool directories
/usr/spool/*/lock  lock file for queue control

SEE ALSO
lpd(8), lpr(1), lpq(1), lprm(1), printcap(5)

DIAGNOSTICS
?Ambiguous command  abbreviation matches more than one command
?Invalid command  no match was found
?Privileged command  command can be executed by root only
NAME
   lp - line printer daemon

SYNOPSIS
   /usr/lib/lpd [-I] [-L logfile] [ port # ]

DESCRIPTION
   Lpd is the line printer daemon (spool area handler) and is normally invoked at boot time from the
   rc(8) file. It makes a single pass through the printcap(5) file to find out about the existing
   printers and prints any files left after a crash. It then uses the system calls listen(2) and accept(2)
   to receive requests to print files in the queue, transfer files to the spooling area, display the queue,
   or remove jobs from the queue. In each case, it forks a child to handle the request so the parent
   can continue to listen for more requests. The Internet port number used to rendezvous with other
   processes is normally obtained with getservent(3N) but can be changed with the port# argument.
   The -L option changes the file used for writing error conditions from the system console to logfile.
   The -I flag causes lpd to log valid requests received from the network. This can be useful for
   debugging purposes.

   Access control is provided by two means. First, all requests must come from one of the machines
   listed in the file /etc/hosts.equiv. Second, if the "rs" capability is specified in the printcap entry
   for the printer being accessed, lpr requests will only be honored for those users with accounts on
   the machine with the printer.

   The file lock in each spool directory is used to prevent multiple daemons from becoming active
   simultaneously, and to store information about the daemon process for lpr(1), lpq(1), and lprm(1).
   After the daemon has successfully set the lock, it scans the directory for files beginning with cf.
   Lines in each cf file specify files to be printed or non-printing actions to be performed. Each such
   line begins with a key character to specify what to do with the remainder of the line.

   J Job Name. String to be used for the job name on the burst page.

   C Classification. String to be used for the classification line on the burst page.

   L Literal. The line contains identification info from the password file and causes the banner
   page to be printed.

   T Title. String to be used as the title for pr(1).

   H Host Name. Name of the machine where lpr was invoked.

   P Person. Login name of the person who invoked lpr. This is used to verify ownership by
   lprm.

   M Send mail to the specified user when the current print job completes.

   f Formatted File. Name of a file to print which is already formatted.

   l Like "f" but passes control characters and does not make page breaks.

   p Name of a file to print using pr(1) as a filter.

   t TROFF File. The file contains troff(1) output (C/A/T phototypesetter commands).

   d DVI File. The file contains TeX output (DVI format from Stanford).

   g Graph File. The file contains data produced by plot(3X).

   c Cifplot File. The file contains data produced by cifplot.

   v The file contains a raster image.

   r The file contains text data with FORTRAN carriage control characters.

   1 TROFF Font R. Name of the font file to use instead of the default.

   2 TROFF Font I. Name of the font file to use instead of the default.

   3 TROFF Font B. Name of the font file to use instead of the default.
Troff Font S. Name of the font file to use instead of the default.

Width. Changes the page width (in characters) used by pr(1) and the text filters.

Indent. The number of characters to indent the output by (in ascii).

Unlink. Name of file to remove upon completion of printing.

File name. The name of the file which is being printed, or a blank for the standard input (when lpr is invoked in a pipeline).

If a file can not be opened, a message will be placed in the log file (normally the console). Lpd will try up to 20 times to reopen a file it expects to be there, after which it will skip the file to be printed.

Lpd uses flock(2) to provide exclusive access to the lock file and to prevent multiple daemons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process id of the daemon and the second is the control file name of the current job being printed. The second line is updated to reflect the current status of lpd for the programs lpq(1) and lprm(1).

FILES
/etc/printcap printer description file
/usr/spool/* spool directories
/dev/lp* line printer devices
/etc/hosts.equiv lists machine names allowed printer access

SEE ALSO
lp(8), pac(8), lpr(1), lpq(1), lprm(1), printcap(5)
4.2BSD Line Printer Spooler Manual
NAME
MAKEDEV - make system special files

SYNOPSIS
/dev/MAKEDEV device...

DESCRIPTION
MAKEDEV is a shell script normally used to install special files. It resides in the /dev directory, as this is the normal location of special files. Arguments to MAKEDEV are usually of the form device-name? where device-name is one of the supported devices listed in section 4 of the manual and '?' is a logical unit number (0-9). A few special arguments create assorted collections of devices and are listed below.

std Create the standard devices for the system; for example, /dev/console, /dev/tty.
local Create those devices specific to the local site. This request runs the shell file /dev/MAKEDEV.local. Site specific commands, such as those used to setup dialup lines as 'ttyd?' should be included in this file.

Since all devices are created using mknod(8), this shell script is useful only to the super-user.

DIAGNOSTICS
Either self-explanatory, or generated by one of the programs called from the script. Use sh -x MAKEDEV in case of trouble.

SEE ALSO
intro(4), confg(8), mknod(8)
NAME
makekey - generate encryption key

SYNOPSIS
/usr/lib/makekey

DESCRIPTION
Makekey improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input, and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute (that is, to require a substantial fraction of a second).

The first eight input bytes (the input key) can be arbitrary ASCII characters. The last two (the salt) are best chosen from the set of digits, upper- and lower-case letters, and ' ' and '/'. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the output key.

The transformation performed is essentially the following: the salt is used to select one of 4096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but modified in 4096 different ways. Using the input key as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 useful key bits in the result.

Makekey is intended for programs that perform encryption (for instance, ed and crypt(1)). Usually makekey's input and output will be pipes.

SEE ALSO
crypt(1), ed(1)
NAME
mkfs - construct a file system

SYNOPSIS
/etc/mkfs special size [ nsect ] [ ntrack ] [ blksize ] [ fragsize ] [ ncpg ] [ minfree ] [ rps ]

DESCRIPTION
N.B.: file system are normally created with the newfs(8) command.

Mkfs constructs a file system by writing on the special file special. The numeric size specifies the number of sectors in the file system. Mkfs builds a file system with a root directory and a lost+found directory (see fsck(8)). The number of i-nodes is calculated as a function of the file system size. No boot program is initialized by mkfs (see newfs(8)).

OPTIONS
The optional arguments allow fine tune control over the parameters of the file system.

Nsect specify the number of sectors per track on the disk.

Ntrack specify the number of tracks per cylinder on the disk.

Blksize gives the primary block size for files on the file system. It must be a power of two, currently selected from 4096 or 8192.

Fragsize gives the fragment size for files on the file system. The fragsize represents the smallest amount of disk space that will be allocated to a file. It must be a power of two currently selected from the range 512 to 8192.

Ncpg specifies the number of disk cylinders per cylinder group. This number must be in the range 1 to 32.

Minfree specifies the minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only the super-user is allowed to allocate disk blocks. The default value is 10%.

rps If a disk does not revolve at 60 revolutions per second, this parameter may be specified.

Users with special demands for their file systems are referred to the paper cited below for a discussion of the tradeoffs in using different configurations.

SEE ALSO
fs(5), dir(5), fsck(8), newfs(8), tunefs(8)

McKusick, Joy, Leffler; A Fast File System for Uniz, Computer Systems Research Group, Dept of EECS, Berkeley, CA 94720;
NAME
  mknod - build special file

SYNOPSIS
  /etc/mknod name [ c ] [ b ] major minor

DESCRIPTION
  Mknod makes a special file. The first argument is the name of the entry. The second is b if the special file is block-type (disks, tape) or c if it is character-type (other devices). The last two arguments are numbers specifying the major device type and the minor device (for example, unit, drive, or line number).

  Mknod is only for use by system configuration people. Normally you should use /dev/MAKEDEV instead when making special files.

SEE ALSO
  mknod(2), makedev(8)
NAME
mkproto - construct a prototype file system

SYNOPSIS
/usr/etc/mkproto special proto

DESCRIPTION
Mkproto is used to bootstrap a new file system. First a new file system is created using newfs(8). Mkproto is then used to copy files from the old file system into the new file system according to the directions found in the prototype file proto. The prototype file contains tokens separated by spaces or new lines. The first tokens comprise the specification for the root directory. File specifications consist of tokens giving the mode, the user-id, the group id, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6 character string. The first character specifies the type of the file. (The characters bed specify regular, block special, character special and directory files respectively.) The second character of the type is either u or - to specify set-user-id mode or not. The third is g or - for the set-group-id mode. The rest of the mode is a three digit octal number giving the owner, group, and other read, write, execute permissions, see chmod(1).

Two decimal number tokens come after the mode; they specify the user and group ID's of the owner of the file.

If the file is a regular file, the next token is a pathname whence the contents and size are copied.

If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers.

If the file is a directory, mkproto makes the entries . and .. and then reads a list of names and (recursively) file specifications for the entries in the directory. The scan is terminated with the token $.

A sample prototype specification follows:

```
   d—777 3 1
   usr  d—777 3 1
       sh —755 3 1 /bin/sh
       ken d—755 6 1
              $
       b0  b—644 3 1 0 0
       c0  c—644 3 1 0 0
              $
```

SEE ALSO
fs(5), dir(5), fsck(8), newfs(8)

BUGS
There should be some way to specify links.
There should be some way to specify bad blocks.
Mkproto can only be run on virgin file systems. It should be possible to copy files into existent file systems.
NAME
mount, umount — mount and dismount file system

SYNOPSIS
/etc/mount [-a] [-r] [-f] [-v] [special] [dir]
/etc/umount [-a] [-v] [dev ...]

DESCRIPTION
Mount announces to the system that a removable file system is present on the device special. The file dir must exist already and it must be a directory (unless the root of the mounted file system is not a directory). It becomes the name of the newly mounted root. The optional argument -r indicates that the file system is to be mounted read-only.

Umount announces to the system that the removable file system previously mounted on device special should be removed.

Mount and umount maintain a table of mounted devices in /etc/mtab. If invoked without an argument, mount displays the table.

Physically write-protected and magnetic tape file systems must be mounted read-only or errors will occur when access times are updated, whether or not any explicit write is attempted.

MOUNT OPTIONS
-r Mount the specified file system read-only.
-a Attempt to mount or unmount all the file systems described in /etc/fstab. In this case, special and dir are taken from /etc/fstab. The special file name from /etc/fstab is the block special name.
-v Verbose — mount displays a message indicating the file system being mounted.
-f Fake a new /etc/mtab entry. Mount does not actually mount any file systems.

UMOUNT OPTIONS
-a Attempt to unmount all the file systems described in /etc/fstab. In this case, special is taken from /etc/fstab.
-v Verbose — umount displays a message indicating the file system being unmouted.

FILES
/etc/mtab mount table
/etc/fstab file system table

SEE ALSO
mount(2), mtab(5), fstab(5)

BUGS
Mounting file systems full of garbage will crash the system.
Mounting a root directory on a non-directory makes some apparently good pathnames invalid.
NAME
nccheck - generate names from i-numbers

SYNOPSIS
/usr/etc/nccheck [-l numbers] [-a] [-s] [filesystem]

DESCRIPTION
N.B.: For most normal file system maintenance, the function of nccheck is subsumed by fsck(8).

Ncheck with no argument generates a pathname versus i-number list of all files on a set of default file systems. Names of directory files are followed by '/.'.

A file system may be specified by the optional filesystem argument.

The report is in no useful order, and probably should be sorted.

OPTIONS
-1 numbers
  Report only those files whose i-numbers follow.

-a
  Print the names '.' and '..', which are ordinarily suppressed.

-s
  Report only special files and files with set-user-ID mode. This is intended to discover concealed violations of security policy.

SEE ALSO
sort(1), dcheck(8), fsck(8), icheck(8)

DIAGNOSTICS
When the filesystem structure is improper, '??' denotes the 'parent' of a parentless file and a pathname beginning with '...' denotes a loop.
NAME
nd - network disk control

SYNOPSIS
/etc/nd [ command ]

DESCRIPTION
The nd command controls the network disk service of the kernel as described in nd(4p). A single command may be given on the command line; if none is given then the standard input is read for a list of commands. Typically, the file /etc/nd.local is used for input. Lines beginning with "#" are considered to be comments.

The available commands are:

```
user hostname hisunit mydev myoff mysize mylunit
```

For the client hostname transform incoming requests for hisunit into server device mydev at offset myoff and size mysize sectors. /dev/nd/ml/mylunit provides a local name for this disk "subpartition". If mysize is "-1", then this user unit is equivalent to the entire filesystem partition mydev (no "subpartitioning" is done.) If mylunit is "-1" then no local name is needed for this user unit; this is usually the case with a swap unit, or a unit represented by an entire filesystem. If hostname is a numeric zero, hisunit refers to a public unit.

```
ether hostname ether_addr [ maxpacks ]
```

The ether command associates an Ethernet address ether_addr with the client hostname. This information is required in order for a client to learn his Internet address when it boots — nd provides the Internet address for hostname based on the Ethernet address received in a boot request. The Ethernet address is given as six hex bytes separated by colons, e.g., 8:0:20:1:1:a3. At least one previous user command must have been issued for the client. The maxpacks option may be given to set the maximum number of packets that the server will send to the client before requesting an acknowledge. The default is 6; it should suffice unless the client Ethernet interface is very slow.

```
version versionnumber
```

The version command gives the level of configuration of the server. Occasionally the need arises to reorganize or reload the diskless partitions. Since the clients will rewrite locally cached blocks, they must be kept from writing their filesystems until they reboot. Before such a reorganization occurs, the system manager should warn diskless users to save files and halt their machines. Modification of the partitions should occur with the disk server off. After modification is complete, versionnumber should be incremented to force users to reboot.

```
son
```

Starts the network disk server. This command should be issued after all user, ether, and version commands.

```
soff
```

Stops the disk server until a subsequent son command.

```
clear
```

Stops the disk server and clears all user and ether information.

```
serverat hostname
```

Systems with disks may use the serverat command to specify a disk server if they wish to use a network disk in addition to their locally attached disk. Even then, this command is only necessary if they wish to use a public network disk or if they wish to change network disk servers.

Last change: 11 January 1984

Sun Release 1.1
FILES

/etc/nd.local

SEE ALSO

nd(4p)

BUGS

No sanity checking of disk partitions is done.
NAME
netstat - show network status

SYNOPSIS
netstat [-Aahnrs] [-a] [-i] [-l] [-n] [-m]

DESCRIPTION
Netstat symbolically displays the contents of various network-related data structures.

OPTIONS
-A show the address of any associated protocol control blocks; used for debugging.
-a show the state of all sockets; normally sockets used by server processes are not shown
-h show the state of the IMP host table
-l show the state of interfaces which have been auto-configured (interfaces statically configured into a system, but not located at boot time are not shown)
-m show statistics recorded by the memory management routines (the network manages a "private share" of memory)
-n show network addresses as numbers (normally netstat interprets addresses and attempts to display them symbolically). proto;
-s show per-protocol statistics
-r show the routing tables

The arguments, system and core allow substitutes for the defaults "/vmunix" and "/dev/kmem".

If an interval is specified, netstat will continuously display the information regarding packet traffic on the configured network interfaces, pausing interval seconds before refreshing the screen.

There are a number of display formats, depending on the information presented. The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and, optionally, the internal state of the protocol.

Address formats are of the form "host.port" or "network.port" if a socket's address specifies a network but no specific host address. When known the host and network addresses are displayed symbolically according to the data bases /etc/hosts and /etc/networks, respectively. If a symbolic name for an address is unknown, or if the -n option is specified, the address is printed in the Internet "dot format"; refer to inet(3N) for more information regarding this format. Unspecified, or "wildcard", addresses and ports appear as "*".

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network address (currently Internet specific) of the interface and the maximum transmission unit ("mtu") are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows the state of the route ("U" if "up"), and whether the route is to a gateway ("G"). Direct routes are created for each interface attached to the local host. The refcnt field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route then discard it. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface utilized for the route.

When netstat is invoked with an interval argument, it displays a running count of statistics related to network interfaces. This display consists of a column summarizing information for all interfaces, and a column for the interface with the most traffic since the system was last rebooted. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

Last change: 3 January 1984
Sun Release 1.1
SEE ALSO
iostat(8), vmstat(8), hosts(5), networks(5), protocols(5), services(5), trpt(8C)

BUGS
The notion of errors is ill-defined. Collisions mean something else for the IMP.
NAME
newaliases – rebuild the database for the mail aliases file

SYNOPSIS
newaliases

DESCRIPTION
Newaliases rebuilds the random access database for the mail aliases file /usr/lib/aliases. It must be run each time /usr/lib/aliases is changed in order for the change to take effect.

SEE ALSO
aliases(5), sendmail(8)
NAME
newfs - construct a new file system

SYNOPSIS
/etc/newfs [-v] [ -n ] [ mkfs-options ] special [ disk-type ]

DESCRIPTION
newfs is a "friendly" front-end to the mkfs(8) program. On the VAX, newfs will look up the type of disk a file system is being created on in the disk description file /etc/disktab; on the Sun the disk type is determined by reading the disk label. newfs then calculate the appropriate parameters to use in calling mkfs, then build the file system by forking mkfs and, if the file system is a root partition, install the necessary bootstrap programs in the initial 16 sectors of the device.

OPTIONS
- n Do not install the bootstrap programs.
- v newfs prints out its actions, including the parameters passed to mkfs.

Options which may be used to override default parameters passed to mkfs are:
- s size The size of the file system in sectors.
- b block-size The block size of the file system in bytes.
- f frag-size The fragment size of the file system in bytes.
- t #tracks/cylinder
- c #cylinders/group
  The number of cylinders per cylinder group in a file system. The default value used is 16.
- m free space %
  The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.
- r revolutions/minute
  The speed of the disk in revolutions per minute (normally 3600).

FILES
/etc/disktab for disk geometry and file partition information (VAX only)
/etc/mkfs to actually build the file system
/usr/mdec for boot strapping programs

SEE ALSO
fs(5), fsck(8), tunefs(8)
NAME
   pac  - printer/plotter accounting information

SYNOPSIS
   /usr/etc/pac [ -P printer ] [ -pprice ] [ -s ] [ -r ] [ -e ] [ name ... ]

DESCRIPTION
   Pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case)
or feet (for raster devices) of paper consumed by each user, and printing out how much each user
consumed in pages or feet and dollars. If any names are specified, then statistics are only printed
for those users; usually, statistics are printed for every user who has used any paper.

OPTIONS
   -P printer
       Do accounting for the named printer. Normally, accounting is done for the default
       printer (site dependent) or the value of the PRINTER environment variable is used.

   -pprice
       Use the value price for the cost in dollars instead of the default value of 0.02.

   -e
       Sorted the output by cost; usually the output is sorted alphabetically by name.

   -r
       Reverse the sorting order.

   -s
       Summarize the accounting information on the summary accounting file; this summary is
       necessary since on a busy system, the accounting file can grow by several lines per day.

FILES
   /usr/adm/?.acc  raw accounting files
   /usr/adm/?.sum  summary accounting files

BUGS
   The relationship between the computed price and reality is as yet unknown.
**NAME**

`pstat` - print system facts

**SYNOPSIS**

```
/etc/pstat -alxptutr [ suboptions ] [ system ] [ corefile ]
```

**DESCRIPTION**

`pstat` interprets the contents of certain system tables. If `corefile` is given, the tables are sought there, otherwise in `/dev/kmem`. The required namelist is taken from `/vmunix` unless `system` is specified.

**OPTIONS**

- `-a` Under `-p`, describe all process slots rather than just active ones.

- `-l` Print the inode table with the these headings:
  
  |- LOC  | The core location of this table entry. |
  |- FLAGS | Miscellaneous state variables encoded thus: |
  |       | `L` locked |
  |       | `U` update time ($(fa(5)$) must be corrected |
  |       | `A` access time must be corrected |
  |       | `M`file system is mounted here |
  |       | `W` wanted by another process (L flag is on) |
  |       | `T` contains a text file |
  |       | `C` changed time must be corrected |
  |       | `S` shared lock applied |
  |       | `E` exclusive lock applied |
  |       | `Z` someone waiting for an exclusive lock |
  |       | `CNT` Number of open file table entries for this inode. |
  |       | `DEV` Major and minor device number of file system in which this inode resides. |
  |       | `RDC` Reference count of shared locks on the inode. |
  |       | `WRC` Reference count of exclusive locks on the inode (this may be $> 1$ if, for example, a file descriptor is inherited across a fork). |
  |       | `INO` I-number within the device. |
  |       | `MODE` Mode bits, see `chmod(2)`. |
  |       | `NLK` Number of links to this inode. |
  |       | `UID` User ID of owner. |
  |       | `SIZ/DEV` Number of bytes in an ordinary file, or major and minor device of special file. |

- `-x` Print the text table with these headings:
  
  |- LOC  | The core location of this table entry. |
  |- FLAGS | Miscellaneous state variables encoded thus: |
  |       | `T` `ptrace(2)` in effect |
  |       | `W` text not yet written on swap device |
  |       | `L` loading in progress |
  |       | `K` locked |
  |       | `w` wanted (L flag is on) |
  |       | `P` resulted from demand-page-from-inode exec format (see `execve(2)`)
  |       | `DADDR` Disk address in swap, measured in multiples of 512 bytes. |
  |       | `CADDR` Head of a linked list of loaded processes using this text segment. |
  |       | `SIZE` Size of text segment, measured in multiples of 512 bytes. |
  |       | `IPTR` Core location of corresponding inode. |
  |       | `CNT` Number of processes using this text segment. |
CCNT  Number of processes in core using this text segment.

-p  Print process table for active processes with these headings:

LOC  The core location of this table entry.
S   Run state encoded thus:
    0  no process
    1  (unused)
    3  runnable
    4  being created
    5  being terminated
    6  stopped under trace

F  Miscellaneous state variables, or-ed together (hexadecimal):
    0000001  loaded
    0000002  a system process (scheduler or page-out daemon)
    0000004  locked for swap out
    0000008  swapped out during process creation
    0000010  traced
    0000020  used in tracing
    0000040  (unused)
    0000080  in page-wait
    0000100  prevented from swapping during fork(2)
    0000200  restore old sigmask after taking signal
    0000400  exiting
    0001000  process resulted from a ufork(2) which is not yet complete
    0002000  another flag for ufork(2)
    0004000  process has no virtual memory, as it is a parent in the context of ufork(2)
    0008000  process is demand paging data pages from its text inode.
    0010000  process has advised of sequential behavior with vadvice(2).
    0020000  process has advised of anomalous behavior with vadvice(2).
    0040000  process is in a sleep which will timeout.
    0080000  (unused)
    0100000  using old signal mechanism
    0200000  process is owed a profiling tick.
    0400000  process is setting up a select(2) call
    080000  Process is a login process
    100000  Page tables for this process have changed (tlb is dirty)

POIP  number of pages currently being pushed out from this process.
PRI  Scheduling priority, see setpriority(2).
SIGNAL  Signals received (signals 1-32 coded in bits 0-31),
UID  Real user ID.
SLP  Amount of time process has been blocked.
TIM  Time resident in seconds; times over 127 coded as 127.
CPU  Weighted integral of CPU time, for scheduler.
NI  Nice level, see setpriority(2).
PGRP  Process number of root of process group (the opener of the controlling terminal).
PID  The process ID number.
PPID  The process ID of parent process.
ADDR  If in core, the page frame number of the first page of the ‘u-area’ of the process. If swapped out, the position in the swap area measured in multiples of 512 bytes.
RSS  Resident set size – the number of physical page frames allocated to this process.
SRSS  RSS at last swap (0 if never swapped).
SIZE  Virtual size of process image (data+ stack) in multiples of 512 bytes.
WCHAN Wait channel number of a waiting process.
LINK Link pointer in list of runnable processes.
TEXTP If text is pure, pointer to location of text table entry.
CLKT Countdown for real interval timer, `setitimer' measured in clock ticks (10 milliseconds).

-t Print table for terminals with these headings:
  RAW Number of characters in raw input queue.
  CAN Number of characters in canonicalized input queue.
  OUT Number of characters in output queue.
  ADDR Physical device address.
  DEL Number of delimiters (newlines) in canonicalized input queue.
  COL Calculated column position of terminal.
  STATE Miscellaneous state variables encoded thus:
    W waiting for open to complete
    O open
    S has special (output) start routine
    C carrier is on
    B busy doing output
    A process is awaiting output
    X open for exclusive use
    H hangup on close
  PGRP Process group for which this is controlling terminal.
  DISC Line discipline; blank is old tty OTTYDISC or “new tty” for NTTYDISC or “net” for NETLDISC (see `blt(4)).

-u print information about a user process; the next argument is its address as given by `p8'().
The process must be in main memory, or the file used can be a core image and the address 0.

-f Print the open file table with these headings:
  LOC The core location of this table entry.
  TYPE The type of object the file table entry points to.
  FLG Miscellaneous state variables encoded thus:
    R open for reading
    W open for writing
    A open for appending
  CNT Number of processes that know this open file.
  INO The location of the inode table entry for this file.
  OFFS/ SOCK The file offset (see `lseek'()), or the core address of the associated socket structure.

-s print information about swap space usage: the number of (1k byte) pages used and free is
given as well as the number of used pages which belong to text images.

-T prints the number of used and free slots in the several system tables and is useful for check-
ing to see how full system tables have become if the system is under heavy load.
FILES
   /vmunix namelist
   /dev/kmem default source of tables

SEE ALSO
   ps(1), stat(2), fs(5)
   K. Thompson, UNIX Implementation

BUGS
   It would be very useful if the system recorded "maximum occupancy" on the tables reported by
   -T; even more useful if these tables were dynamically allocated.
NAME
  quot — summarize file system ownership

SYNOPSIS
  /etc/quot [ option ] ... [ filesystem ]

DESCRIPTION
  Quot prints the number of blocks in the named filesystem currently owned by each user. If no
  filesystem is named, a default name is assumed.

OPTIONS
  -n  Run the pipeline nccheck filesystem | sort -0n | quot -n filesystem to produce a list
       of all files and their owners.
  -c  Print three columns giving file size in blocks, number of files of that size, and cumulative
       total of blocks in that size or smaller file.
  -f  Print count of number of files as well as space owned by each user.

FILES
  Default file system varies with system.
  /etc/passwd to get user names

SEE ALSO
  ls(1), du(1)
NAME
rc – command script for auto-reboot and daemons

SYNOPSIS
/etc/rc
/etc/rc.local

DESCRIPTION
Rc is the command script which controls the automatic reboot and rc.local is the script holding commands which are pertinent only to a specific site.

When an automatic reboot is in progress, rc is invoked with the argument autoboot and runs a fsck with option -p to “preen” all the disks of minor inconsistencies resulting from the last system shutdown and to check for serious inconsistencies caused by hardware or software failure. If this auto-check and repair succeeds, then the second part of rc is run.

The second part of rc, which is run after a auto-reboot succeeds and also if rc is invoked when a single user shell terminates (see init(8)), starts all the daemons on the system, preserves editor files and clears the scratch directory /tmp. Rc.local is executed immediately before any other commands after a successful fsck. Normally, the first commands placed in the rc.local file define the machine’s name, using hostname(1), and save any possible core image that might have been generated as a result of a system crash, savecore(8). The latter command is included in the rc.local file because the directory in which core dumps are saved is usually site specific.

SEE ALSO
init(8), reboot(8), savecore(8)
NAME
   rdate – set system date from a remote host

SYNOPSIS
   /usr/ucb/rdate hostname

DESCRIPTION
   Rdate sets the local date and time from the hostname given as argument. You must be super-user on the local system. Typically rdate can be inserted as part of your /etc/rc.local startup script.

SEE ALSO
   timed(8C)

BUGS
   Could be modified to accept a list of hostnames and try each until a valid date returned. Better yet would be to write a real date server that accepted broadcast requests.

   Version 1.1 of rdate is incompatible with version 1.0. rdate's issued on 1.1 systems will not get the time from systems running 1.0.
NAME
reboot - UNIX bootstrapping procedures

SYNOPSIS
/etc/reboot [-n] [-q]

DESCRIPTION
The UNIX operating system is started by placing it in memory at location zero and transferring to zero. Since the system is not reentrantable, it is necessary to read it in from disk or tape each time it is to be bootstrapped.

Rebooting a running system. When a UNIX system is running and a reboot is desired, shutdown(8) is normally used. If there are no users then /etc/reboot can be used. Reboot performs a sync operation on the disks, and then a multi-user reboot (as described below) is initiated. This causes a system to be booted and an automatic disk check to be performed. If all this succeeds without incident, the system is then brought up for many users.

OPTIONS
-n option avoids the sync. It can be used if a disk or the processor is on fire.
-q reboots quickly and ungracefully, without first shutting down running processes.

Power fail and crash recovery. Normally, the system will reboot itself at power-up or after crashes. Provided the auto-restart is enabled on the machine front panel, an automatic consistency check of the file systems will be performed then and unless this fails the system will resume multi-user operations.

On both processors, the boot program finds the corresponding file on the given device, loads that file into memory location zero, and starts the program at the entry address specified in the program header (after clearing off the high bit of the specified entry address.) Normal line editing characters can be used in specifying the pathname.

For tapes, the minor device number gives a file offset.

FILES
/vmunix system code
/boot system bootstrap

SEE ALSO
crash(8S), fsck(8), init(8), rc(8), shutdown(8), halt(8), newfs(8)
NAME
recnews — receive unprocessed articles via mail

SYNOPSIS
/usr/lib/news/recnews [ newsgroup [ sender ] ]

DESCRIPTION
Recnews reads a letter from the standard input; determines the article title, sender, and newsgroup; and gives the body to inews with the right arguments for insertion.

If newsgroup is omitted, the to line of the letter is used. If sender is omitted, the sender is determined from the from line of the letter. The title is determined from the subject line.

SEE ALSO
inews(1), uurec(8), sendnews(8), readnews(1), checknews(1)
NAME
renice - alter priority of running processes

SYNOPSIS
/etc/renice [ -g ] [ -u ] priority who ...

DESCRIPTION
Renice can be used to alter the scheduling priority of one or more running processes. By default, the processes to be affected are specified by their process id's. If the -g option is specified, the who parameters are interpreted as process groups and all the processes in the specified process groups have their scheduling priority altered. If the -u option is indicated, the who parameters are interpreted as user names and all process owned by the user are affected.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their “nice value” within the range 0 to PRIO_MIN (20). (This prevents overriding administrative fiats.) The super-user may alter the priority to any value in the range PRIO_MAX (-20) to PRIO_MIN. Useful priorities are: 19 (the affected processes will run only when nothing else in the system wants to), 0 (the “base” scheduling priority), anything negative (to make things go very fast).

If no who parameter is specified, the current process (alternatively, process group or user) is used.

FILES
/etc/passwd to map user names to user id's

SEE ALSO
getpriority(2)

BUGS
If you make the priority very negative, then the process cannot be interrupted. To regain control you make the priority greater than zero. Non super-users can not increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.
NAME
   restore – incremental file system restore

SYNOPSIS
   /etc/restore key [ name ... ]

DESCRIPTION
   Restore restores files from tapes previously created via the dump(8) command. Restore's actions are controlled by the key argument. The key is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to restore are file or directory names specifying the files that are to be restored. Unless the h key is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

FUNCTION LETTERS
   The function portion of the key is specified by one of the following letters:

   r   The tape is read and loaded into the current directory. This should not be done lightly; the r key should only be used to restore a complete dump tape onto a clear file system or to restore an incremental dump tape after a full level zero restore. Thus:

       tutorial% /etc/newfs /dev/xy0g eagle
       tutorial% /etc/mount /dev/xy0g /mnt
       tutorial% cd /mnt
       tutorial% restore r

   is a typical sequence to restore a complete dump. Another restore can be done to get an incremental dump in on top of this. A dump(8) followed by a newfs(8) and a restore can be used to change the size of a file system.

   R   Restore requests a particular tape or a multi volume set on which to restart a full restore (see the r key above). This allows restore to be interrupted and then restarted.

   x   Extract the named files from the tape. If the named file matches a directory whose contents had been written onto the tape, and the h key is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, the root directory is extracted, which results in the entire content of the tape being extracted, unless the h key has been specified.

   t   List the names of the specified files if they occur on the tape. If no file argument is given, the root directory is listed, which results in the entire content of the tape being listed, unless the h key has been specified. Note that the t key replaces the function of the old dumpdir program.

   l   Restore files interactively from a dump tape. After reading in the directory information from the tape, restore provides a shell like interface within which the user can move around the directory tree selecting files to be extracted. The available commands that this 'shell' provides are given below. For those commands that require an argument, the default is the current directory.

       ls [arg] List the current or specified directory. Entries that are directories are appended with a '/'. Entries that have been marked for extraction are prepended with a '*'. If the verbose key is set the inode number of each entry is also listed.

       cd arg Change the current working directory to the specified argument.

       pwd Print the full pathname of the current working directory.

       add [arg] Add the current directory or specified argument to the list of files to be extracted. If a directory is specified, add that directory and all its to the extraction list (unless the h key is specified on the command line). Files that are on
the extraction list are prepended with a '*' when they are listed by ls.

**delete** [arg]
Delete the current directory or specified argument from the list of files to be extracted. If a directory is specified, delete that directory and all its descendents from the extraction list (unless the h key is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

**extract**
Extract from the dump tape all the files that are on the extraction list. *Restore* asks which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

**verbose**
Toggle the sense of the v key. When the verbose key is set, the *ls* command lists the inode numbers of all entries, and *restore* also displays information about each file as it is extracted.

**help**
List a summary of the available commands.

**quit**
*Restore* immediately exits, even if the extraction list is not empty.

**FUNCTION MODIFIERS**
The function modifiers which may be used in addition to the function letters are as follows. A few of these modifiers — namely b, f, and s— take an argument from the command line. If you use more than one of b, f, and s, nest your arguments as you do your modifiers; if you use b first on the command line, place its argument first on the command line, and so on.

**b**
Specifies the blocking factor for the *restore*. The blocking factor is taken from the next argument on the command line. This corresponds to the b key in *dump*(8).

**d**
Turns on debugging output.

**v**
This is the verbose option. It means display the name of each file it treats preceded by its file type. *Restore* normally works silently.

**f**
Use the next argument to *restore* as the name of the archive instead of /dev/rmt?. If the name of the file is '-', *restore* reads from standard input. If the name of the file is 'machine:device' the restore is done from the specified machine through the internet using *rmt*(8C). Thus, *dump*(8) and *restore* can be used in a pipeline to dump and restore a file system with the command

```
tutorial% dump of /usr | (cd /mnt; restore xf -)
```

**s**
The next argument to *restore* is the number of files to skip in the case where there are multiple dump files on the dump tape. For example, a command like

```
tutorial% restore xfs /dev/nrare0 5
```
would position you at the sixth file on the tape.

**y**
Do not ask whether to abort the restore in the event of tape errors. *Restore* always tries to skip over the bad tape block(s) and continue as best it can.

**m**
Extract by inode numbers rather than by file name. This is useful if only a few files are being extracted, and one wants to avoid regenerating the complete pathname to the file.

**h**
Extract the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

**DIAGNOSTICS**
Complaints about bad key characters.
Complaints if it gets a read error. If y has been specified, or the user responds 'y', *restore* will attempt to continue the restore.
If the dump extends over more than one tape, `restore` asks the user to change tapes. If the x or l key has been specified, `restore` also asks which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

There are numerous consistency checks that can be listed by `restore`. Most checks are self-explanatory or can 'never happen'. Common errors are given below.

Converting to new file system format.
   A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

<filename>: not found on tape
   The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file <inumber>, got <inumber>
   A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

Incremental tape too low
   When doing incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

Incremental tape too high
   When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or that has too high an incremental level has been loaded.

Tape read error while restoring <filename>

Tape read error while skipping over inode <inumber>

Tape read error while trying to resynchronize
   A tape read error has occurred. If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

resync restore, skipped <num> blocks
   After a tape read error, `restore` may have to resynchronize itself. This message lists the number of blocks that were skipped over.

FILES
   /dev/rmt? the default tape drive
   /tmp/rstdir* file containing directories on the tape.
   /tmp/rstmode* owner, mode, and time stamps for directories.
   ./restoresymtab information passed between incremental restores.

SEE ALSO
   dump(8), newfs(8), mount(8), mkfs(8), rmt(8C)

BUGS
   `Restore` can get confused when doing incremental restores from dump tapes that were made on active file systems.

A level zero dump must be done after a full restore. Because `restore` runs in user mode, it has no control over inode allocation; this means that `restore` re-positions the files, although it does change their contents. Thus, a full dump must be done to get a new set of directories reflecting the new file positions, so that later incremental dumps will be correct.
NAME
rexecd - remote execution server

SYNOPSIS
/usr/etc/in.rexecd host.port

DESCRIPTION
Re:zecd is the server for the rexec(3N) routine. The server provides remote execution facilities with authentication based on user names and encrypted passwords. It is invoked automatically as needed by inetd(8C), and then executes the following protocol:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the stddie. A second connection is then created to the specified port on the client's machine.

3) A null terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null terminated, encrypted, password of at most 16 characters is retrieved on the initial socket.

5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) Re:zecd then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the connection associated with the stddie and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by re:zecd.

DIAGNOSTICS
All diagnostic messages are returned on the connection associated with the stddie, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

"username too long"
The name is longer than 16 characters.

"password too long"
The password is longer than 16 characters.

"command too long"
The command line passed exceeds the size of the argument list (as configured into the system).

"Login incorrect."
No password file entry for the user name existed.

"Password Incorrect."
The wrong password was supplied.

"No remote directory."
The chdir command to the home directory failed.

"Try again."
A fork by the server failed.
"/bin/sh ..."
The user's login shell could not be started.

BUGS
Indicating "Login incorrect" as opposed to "Password incorrect" is a security breach which allows people to probe a system for users with null passwords.
A facility to allow all data exchanges to be encrypted should be present.

SEE ALSO
inetd(8C)
NAME
rlogind - remote login server

SYNOPSIS
/etc/in.rlogind host.port

DESCRIPTION
Rlogind is the server for the rlogin(1C) program. The server provides a remote login facility with authentication based on privileged port numbers.

Rlogind is invoked by inetd(8C) when a remote login connection is established, and executes the following protocol:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection. The client's address and port number are passed as arguments to rlogind by inetd in the form "host.port" with host in hex and port in decimal.

2) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see hosts(5)), the server aborts the connection.

Once the source port and address have been checked, rlogind allocates a pseudo terminal (see pty(4)), and manipulates file descriptors so that the slave half of the pseudo terminal becomes the stdln, stdout, and stderr for a login process. The login process is an instance of the login(1) program, invoked with the -r option. The login process then proceeds with the authentication process as described in rshd(8C), but if automatic authentication fails, it reprompts the user to login as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the rlogin program. In normal operation, the packet protocol described in pty(4) is invoked to provide 'S/'Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, "TERM"; see environ(5).

DIAGNOSTICS
All diagnostic messages are returned on the connection associated with the stderr, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

"Hostname for your address unknown."
No entry in the host name data base existed for the client's machine.

"Try again!"
A fork by the server failed.

"/bin/sh ..."
The user's login shell could not be started.

BUGS
The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

SEE ALSO
inetd(8C)
NAME
rmail - handle remote mail received via uucp

SYNOPSIS
rmail user ...

DESCRIPTION
Rmail interprets incoming mail received via uucp(1C), collapsing "From" lines in the form generated by binmail(1) into a single line of the form "return-path!sender", and passing the processed mail on to sendmail(8).

Rmail is explicitly designed for use with uucp and sendmail.

SEE ALSO
binmail(1), uucp(1C), sendmail(8)

BUGS
Rmail should not reside in /bin.
NAME
rmt - remote magtape protocol module

SYNOPSIS
/etc/rmt

DESCRIPTION
Rmt is a program used by the remote dump and restor programs in manipulating a magnetic tape
drive through an interprocess communication connection. Rmt is normally started up with an
resec(3N) or rcmd(3N) call.

The rmt program accepts requests specific to the manipulation of magnetic tapes, performs the
commands, then responds with a status indication. All responses are in ASCII and in one of two
forms. Successful commands have responses of

Anumber

where number is an ASCII representation of a decimal number. Unsuccessful commands are
responded to with

Error-number\error-message\n,

where error-number is one of the possible error numbers described in intro(2) and error-message is
the corresponding error string as printed from a call to perror(3). The protocol is comprised of
the following commands (a space is present between each token).

O device mode
Open the specified device using the indicated mode. Device is a full pathname
and mode is an ASCII representation of a decimal number suitable for passing to
open(2). If a device had already been opened, it is closed before a new open is
performed.

C device
Close the currently open device. The device specified is ignored.

L whence offset
Perform an lseek(2) operation using the specified parameters. The response
value is that returned from the lseek call.

W count
Write data onto the open device. Rmt reads count bytes from the connection,
aborting if a premature end-of-file is encountered. The response value is that
returned from the write(2) call.

R count
Read count bytes of data from the open device. If count exceeds the size of
the data buffer (10 kilobytes), it is truncated to the data buffer size. Rmt then per-
forms the requested read(2) and responds with Acount-read\ when the read was
successful; otherwise an error in the standard format is returned. If the read was
successful, the data read is then sent.

I operation count
Perform a MTIOCOP ioctl(2) command using the specified parameters. The
parameters are interpreted as the ASCII representations of the decimal values to
place in the mt_op and mt_count fields of the structure used in the ioctl call.
The return value is the count parameter when the operation is successful.

S
Return the status of the open device, as obtained with a MTIOCGET ioctl call.
If the operation was successful, an "ack" is sent with the size of the status
buffer, then the status buffer is sent (in binary).

Any other command causes rmt to exit.

DIAGNOSTICS
All responses are of the form described above.
SEE ALSO
   rcmd(3N), rexec(3N), mtio(4), dump(8), restore(8)

BUGS
   People tempted to use this for a remote file access protocol are discouraged.
ROUTE(8C) MAINTENANCE COMMANDS ROUTE(8C)

NAME
route – manually manipulate the routing tables

SYNOPSIS
/usr/etc/route [-f] [ command args ]

DESCRIPTION
Route is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, routed(8C), should tend to this task.

Route accepts three commands: add, to add a route; delete, to delete a route; and change, to modify an existing route.

All commands have the following syntax:

/usr/etc/route command destination gateway [ metric ]

where destination is a host or network for which the route is “to”, gateway is the gateway to which packets should be addressed, and metric is an optional count indicating the number of hops to the destination. If no metric is specified, route assumes a value of 0. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with destination. If the destination has a “local address part” of INADDR_ANY, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected via a gateway, the metric should be greater than 0. All symbolic names specified for a destination or gateway are looked up first in the host name database, hosts(5). If this lookup fails, the name is then looked for in the network name database, networks(5).

Route uses a raw socket and the SIOCADDRT and SIOCDELRT ioctl's to do its work. As such, only the super-user may modify the routing tables.

If the -f option is specified, route will “flush” the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, the tables are flushed prior to the command's application.

DIAGNOSTICS
“add %s gateway %s flags %x”
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the ioctl call.

“delete %s gateway %s flags %x”
As above, but when deleting an entry.

“%s %s done”
When the -f flag is specified, each routing table entry deleted is indicated with a message of this form.

“not in table”
A delete operation was attempted for an entry which wasn't present in the tables.

“routing table overflow”
An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO
routing(4N), routed(8C)

BUGS
The change operation is not implemented, one should add the new route, then delete the old one.

84 Last change: 25 October 1983 Sun Release 1.1
NAME
routed - network routing daemon

SYNOPSIS
/usr/etc/in.routed [-s] [-q] [-t] [logfile]

DESCRIPTION
Routed is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries.

In normal operation routed listens on udp(4P) socket 520 (decimal) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When routed is started, it uses the SIOCGIFCONF ioctl to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed the host will forward packets between networks. Routed then transmits a request packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for request and response packets from other hosts.

When a request packet is received, routed formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric relative to the sender.

Request packets received by routed are used to update the routing tables if one of the following conditions is satisfied:

1. No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (that is, the hop count is not infinite).
2. The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
3. The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.
4. The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, routed records the change in its internal tables and generates a response packet to all directly connected hosts and networks. Routed waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables to allow possible unstable situations to settle.

In addition to processing incoming packets, routed also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

Supplying the -s option forces routed to supply routing information whether it is acting as an internetwork router or not. The -q option is the opposite of the -s option. If the -t option is specified, all packets sent or received are printed on the standard output. In addition, routed will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process. Any other argument supplied is interpreted as the name of file in which routed's actions should be logged. This log contains information about any changes to the routing tables and a
history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, routed supports the notion of "distant" passive and active gateways. When routed is started up, it reads the file /etc/gateways to find gateways which may not be identified using the SIOGIFCONF ioctl. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a routed process running on the machine). Passive gateways are maintained in the routing tables forever and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted.

The /etc/gateways is comprised of a series of lines, each in the following format:

```
<net | host > name1 gateway name2 metric value < passive | active>
```

The net or host keyword indicates if the route is to a network or specific host.

*Name1* is the name of the destination network or host. This may be a symbolic name located in /etc/networks or /etc/hosts, or an Internet address specified in “dot” notation; see *inet*(3N).

*Name2* is the name or address of the gateway to which messages should be forwarded.

*Value* is a metric indicating the hop count to the destination host or network.

The keyword passive or active indicates if the gateway should be treated as passive or active (as described above).

**FILES**

/etc/gateways for distant gateways

**SEE ALSO**

Internet Transport Protocols, XSIS 028112, Xerox System Integration Standard. (Sun 800-1066-01)

udp(4P)

**BUGS**

The kernel's routing tables may not correspond to those of routed for short periods of time while processes utilizing existing routes exit; the only remedy for this is to place the routing process in the kernel.

*Routed* should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information.
NAME
rshd - remote shell server

SYNOPSIS
/etc/ln.rshd host.port

DESCRIPTION
Rshd is the server for the remd(3N) routine and, consequently, for the rsh(1C) program. The server provides remote execution facilities with authentication based on privileged port numbers.

Rshd is invoked by inetd(8C) each time a shell service is requested, and executes the following protocol:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection. The clients host address (in hex) and port number (in decimal) are the argument passed to rshd.

2) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the stderr. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see host(5)), the server aborts the connection.

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the server's machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the client's machine.

7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) Rshd then validates the user according to the following steps. The remote user name is looked up in the password file and a chdir is performed to the user's home directory. If the lookup or fails, the connection is terminated. If the chdir fails, it does a chdir to / (root). If the user is not the super-user, (user id 0), the file /etc/host.equiv is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file .rhosts in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated.

9) A null byte is returned on the connection associated with the stderr and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by rshd.

DIAGNOSTICS
All diagnostic messages are returned on the connection associated with the stderr, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the command execution).

"locuser too long"
The name of the user on the client's machine is longer than 16 characters.
"remuser too long"
The name of the user on the remote machine is longer than 16 characters.

"command too long"
The command line passed exceeds the size of the argument list (as configured into the system).

"Hostname for your address unknown."
No entry in the host name database existed for the client's machine.

"Login incorrect."
No password file entry for the user name existed.

"Permission denied."
The authentication procedure described above failed.

"Can't make pipe."
The pipe needed for the stderr, wasn't created.

"Try again."
A fork by the server failed.

"/bin/sh: ...
The user's login shell could not be started.

SEE ALSO
rsh(1C), rcmd(3N)

BUGS
The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.
NAME
rwhod — system status server

SYNOPSIS
/etc/rwhod

DESCRIPTION
Rwhod is the server which maintains the database used by the ruho(1C) and runtime(1C) programs. Its operation is predicated on the ability to broadcast messages on a network.

Rwhod operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network. As a consumer of information, it listens for other rwhod servers' status messages, validating them, then recording them in a collection of files located in the directory /usr/spool/rwho.

The ruho server transmits and receives messages at the port indicated in the "ruho" service specification, see service(5). The messages sent and received, are of the form:

```c
struct outmp {
    char out_line[8]; /*tty name*/
    char out_name[8]; /*user id*/
    long out_time; /*time on*/
};

struct whod {
    char wd_vers;
    char wd_type;
    char wd_fill[2];
    int wd_sendtime;
    int wd_recvtime;
    char wd_hostname[32];
    int wd_loadav[3];
    int wd_boottime;
    struct whoent {
        struct outmp we_utmp;
        int we_idle;
    } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are calculated by the w(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the gethostname(2) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the utmp(5) entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the ruho server are discarded unless they originated at a ruho server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by rwhod are placed in files named whod.hostname in the directory /usr/spool/rwho. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 60 seconds. Rwhod performs an nlist(3) on /vmunix every 10 minutes to guard against the possibility that this file is not the system image currently operating.
SEE ALSO
rwho(1C), ruptime(1C)

BUGS
Should relay status information between networks. People often interpret the server dying as a machine going down.
NAME
sa, accton - system accounting

SYNOPSIS

```
/usr/etc/sa [-abcdfjkklmnstuv] [ file ]
/usr/etc/accton [ file ]
```

DESCRIPTION

With an argument naming an existing file, accton causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

Sa reports on, cleans up, and generally maintains accounting files.

Sa is able to condense the information in /usr/adm/acct into a summary file /usr/adm/savacct which contains a count of the number of times each command was called and the time resources consumed. This condensation is desirable because on a large system /usr/adm/acct can grow by 500K bytes per day. The summary file is normally read before the accounting file, so the reports include all available information.

If a file name is given as the last argument, that file will be treated as the accounting file; /usr/adm/acct is the default.

Output fields are labelled: 'cpu' for the sum of user+ system time (in minutes), 're' for real time (also in minutes), 'k' for cpu-time averaged core usage (in 1k units), 'avio' for average number of I/O operations per execution. With options fields labelled 'tio' for total I/O operations, 'k*sec' for cpu storage integral (kilo-core seconds), 'u' and 's' for user and system cpu time alone (both in minutes) will sometimes appear.

OPTIONS

a Place all command names containing unprintable characters and those used only once under the name '***other.'
b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.
c Besides total user, system, and real time for each command print percentage of total time over all commands.
d Sort by average number of disk I/O operations.
D Print and sort by total number of disk I/O operations.
f Force no interactive threshold compression with -v flag.
i Don't read in summary file.
j Instead of total minutes time for each category, give seconds per call.
k Sort by cpu-time average memory usage.
K Print and sort by cpu-storage integral.
l Separate system and user time; normally they are combined.
m Print number of processes and number of CPU minutes for each user.
n Sort by number of calls.
r Reverse order of sort.
s Merge accounting file into summary file /usr/adm/savacct when done.
t For each command report ratio of real time to the sum of user and system times.
u Superseding all other flags, print for each record in the accounting file the user ID and command name.
Followed by a number \( n \), types the name of each command used \( n \) times or fewer. Await a reply from the terminal; if it begins with 'y', add the command to the category '**junk**'. This is used to strip out garbage.

**FILES**

\[ /usr/adm/acct \] raw accounting

**SEE ALSO**

ac(8), acct(2), acct(5), savacct(5), usracct(5)
NAME
savecore - save a core dump of the operating system

SYNOPSIS
/usr/etc/savecore dirname [ system ]

DESCRIPTION
Savecore is meant to be called near the end of the /etc/rc file after the system boots. Savecore's function is to save the core dump of the system (assuming one was made) and to write a reboot message in the shutdown log.

Savecore checks the core dump to be certain it corresponds with the current running version of the operating system. If it does, savecore saves the core image in the file dirname/vmcore.n and its brother, the namelist, dirname/vmunix.n The trailing .n in the pathnames is replaced by a number which grows every time savecore is run in that directory.

Before savecore writes out a core image, it reads a number from the file dirname/minfree. If there less free space on the filesystem which contains dirname than the number obtained from the minfree file, the core dump is not done. If the minfree file does not exist, savecore always writes out the core file (assuming that a core dump was taken).

Savecore also writes a reboot message in the shutdown log. If the system crashed as a result of a panic, savecore records the panic string in the shutdown log too.

If the core dump was from a system other than /vmunix, the name of that system must be supplied as $name$.

FILES
/usr/adm/shutdownlog shutdown log
/usr/crash/bounds number to assign to the core images
/vmunix current UNIX

BUGS
Can be fooled into thinking a core dump is the wrong size.
NAME
sendmail – send mail over the internet

SYNOPSIS

DESCRIPTION
Sendmail sends a message to one or more people, routing the message over whatever networks are necessary. Sendmail does internetwork forwarding as necessary to deliver the message to the correct place.

Sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; sendmail is used only to deliver pre-formatted messages.

With no flags, sendmail reads its standard input up to a control-D or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions, for example, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

OPTIONS
-ba Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.
-bd Run as a daemon. This requires Berkeley IPC.
-bl Initialize the alias database.
-bm Deliver mail in the usual way (default).
-bp Print a summary of the mail queue.
-bs Use the SMTP protocol as described in RFC821. This flag implies all the operations of the -ba flag that are compatible with SMTP.
-bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
-bv Verify names only – do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.
-bs Create the configuration freeze file.
-Cfile Use alternate configuration file.
-dX Set debugging value to X.
-Ffullname Set the full name of the sender.
-fname Sets the name of the "from" person (that is, the sender of the mail). -f can only be used by the special users root, daemon, and network, or if the person you are trying to become is the same as the person you are.
-hN Set the hop count to N. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.
-n Don't do aliasing.
-oz value Set option x to the specified value. Options are described below.
-q[ time]  Processed saved messages in the queue at given intervals. If time is omitted, process the queue once. Time is given as a tagged number, with 's' being seconds, 'm' being minutes, 'h' being hours, 'd' being days, and 'w' being weeks. For example, "-qlh30m" or "-q90m" would both set the timeout to one hour thirty minutes.

-rname  An alternate and obsolete form of the -f flag.

-t  Read message for recipients. To, Cc, and Bcc: lines will be scanned for people to send to. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed.

-v  Go into verbose mode. Alias expansions will be announced, etc.

PROCESSING OPTIONS
There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the -o flag or in the configuration file. These are described in detail in the Installation and Operation Guide. The options are:

Afile  Use alternate alias file.

C  On mailers that are considered "expensive" to connect to, don't initiate immediate connection. This requires queueing.

dz  Set the delivery mode to z. Delivery modes are 'i' for interactive (synchronous) delivery, 'b' for background (asynchronous) delivery, and 'q' for queue only — that is, actual delivery is done the next time the queue is run.

D  Try to automatically rebuild the alias database if necessary.

ez  Set error processing to mode z. Valid modes are 'm' to mail back the error message, 'w' to "write" back the error message (or mail it back if the sender is not logged in), 'p' to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes 'm' or 'w' and if the sender is local to this machine, a copy of the message is appended to the file "dead.letter" in the sender's home directory.

F mode  The mode to use when creating temporary files.

f  Save UNIX-style From lines at the front of messages.

gN  The default group id to use when calling mailers.

Hfile  The SMTP help file.

i  Do not take dots on a line by themselves as a message terminator.

Ln  The log level.

m  Send to "me" (the sender) also if I am in an alias expansion.

o  If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

Queue dir  Select the directory in which to queue messages.

rtimeout  The timeout on reads; if none is set, sendmail will wait forever for a mailer.

Sfile  Save statistics in the named file.

s  Always instantiate the queue file, even under circumstances where it is not strictly necessary.
**Time**
Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.

**tz, dtz**
Set the name of the time zone.

**uN**
Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep sendmail from suppressing the blanks from between arguments.

Sendmail returns an exit status describing what it did. The codes are defined in `<sysexits.h>

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX_OK</td>
<td>Successful completion on all addresses.</td>
</tr>
<tr>
<td>EX_NOUSER</td>
<td>User name not recognized.</td>
</tr>
<tr>
<td>EX_UNAVAILABLE</td>
<td>Catchall meaning necessary resources were not available.</td>
</tr>
<tr>
<td>EX_SYNTAX</td>
<td>Syntax error in address.</td>
</tr>
<tr>
<td>EX_SOFTWARE</td>
<td>Internal software error, including bad arguments.</td>
</tr>
<tr>
<td>EX_OSERR</td>
<td>Temporary operating system error, such as &quot;cannot fork&quot;.</td>
</tr>
<tr>
<td>EX_NOHOST</td>
<td>Host name not recognized.</td>
</tr>
<tr>
<td>EX_TEMPFAIL</td>
<td>Message could not be sent immediately, but was queued.</td>
</tr>
</tbody>
</table>

If invoked as `newaliases`, sendmail rebuilds the alias database. If invoked as `mailq`, sendmail prints the contents of the mail queue.

**FILES**
Except for `/usr/lib/sendmail.cf`, these pathnames are all specified in `/usr/lib/sendmail.cf`. Thus, these values are only approximations.

- `/usr/lib/aliases`
  - raw data for alias names
- `/usr/lib/aliases.pag`
  - data base of alias names
- `/usr/lib/aliases.dir`
  - configuration file
- `/usr/lib/sendmail.fc`
  - frozen configuration
- `/usr/lib/sendmail.hf`
  - help file
- `/usr/lib/sendmail.st`
  - collected statistics
- `/usr/bin/uux`
  - to deliver uucp mail
- `/bin/mail`
  - to deliver local mail
- `/usr/spool/mqueue/*`
  - temp files and queued mail

**SEE ALSO**
`biff(1)`, `binmail(1)`, `mail(1)`, `aliases(5)`,
`DARPA Internet Request For Comments RFC819, RFC821, RFC822`,

In the *System Manager’s Manual:*

*Setting Up the Mail System* in the *System Set-up and Operation* chapter.

*Sendmail – An Internetwork Mail Router*, in the Tutorials section

*Sendmail Installation and Operation Guide*, in the Tutorials section

**BUGS**
Sendmail converts blanks in addresses to dots. This is incorrect according to the old ARPANET mail protocol RFC733 (NIC 41952), but is consistent with the new protocols (RFC822).
NAME
sendnews – send news articles via mail

SYNOPSIS
sendnews [ -o ] [ -a ] [ -b ] [ -n newsgroups ] destination

DESCRIPTION
sendnews reads an article from its standard input, performs a set of changes to it, and gives it to
the mail program to mail it to destination.
An ‘N’ is prepended to each line for decoding by uurec(1).

OPTIONS
-0 handle old format articles.
-a used for sending articles via the ARPANET. It maps the article's path from uucphost!zzz
to zzz@arpahost.
-b used for sending articles via the Berknet. It maps the article's path from uucphost!zzz
to berkhost!zzz.
-n change the article's newsgroup to the specified newsgroup.

SEE ALSO
inews(1), uurec(8), recnews(8), readnews(1), checknews(1)
NAME
shutdown – close down the system at a given time

SYNOPSIS
/etc/shutdown [-k] [-r] [-h] time [warning-message ...]

DESCRIPTION
Shutdown provides an automated shutdown procedure which a super-user can use to notify users
nicely when the system is shutting down, saving them from system administrators, hackers, and
gurus, who would otherwise not bother with niceties.
Time is the time at which shutdown will bring the system down and may be the word now (indicating
an immediate shutdown) or specify a future time in one of two formats: +number and
hour:min. The first form brings the system down in number minutes and the second brings the
system down at the time of day indicated (as a 24-hour clock).
At intervals which get closer together as apocalypse approaches, warning messages are displayed
at the terminals of all users on the system. Five minutes before shutdown, or immediately if
shutdown is in less than 5 minutes, logins are disabled by creating /etc/nologin and writing a
message there. If this file exists when a user attempts to log in, login(1) prints its contents and
exits. The file is removed just before shutdown exits.
At shutdown time a message is written in the file /usr/adm/shutdownlog, containing the time of
shutdown, who ran shutdown and the reason. Then a terminate signal is sent at init to bring the
system down to single-user state.
The time of the shutdown and the warning message are placed in /etc/nologin and should be used
to inform the users about when the system will be back up and why it is going down (or anything
else).

OPTIONS
As an alternative to the above procedure, these options can be specified:
- r    Execute reboot(8).
- h    Execute halt(8).
- k    If it isn't obvious, -k is to make people think the system is going down!

FILES
/etc/nologin tells login not to let anyone log in
/usr/adm/shutdownlog log file for successful shutdowns.

SEE ALSO
login(1), reboot(8)

BUGS
Only allows you to kill the system between now and 23:59 if you use the absolute time for shut-
down.
STICKY(8) MAINTENANCE COMMANDS STICKY(8)

NAME
sticky - executable files with persistent text

DESCRIPTION
While the 'sticky bit', mode 01000 (see chmod(2)), is set on a sharable executable file, the text of that file will not be removed from the system swap area. Thus the file does not have to be fetched from the file system upon each execution. As long as a copy remains in the swap area, the original text cannot be overwritten in the file system, nor can the file be deleted. Directory entries can be removed so long as one link remains.

Sharable files are made by the -z option of id(1).

To replace a sticky file that has been used do: (1) Clear the sticky bit with chmod(1). (2) Execute the old program to flush the swapped copy. This can be done safely even if others are using it. (3) Overwrite the sticky file. If the file is being executed by any process, writing will be prevented; it suffices to simply remove the file and then rewrite it, being careful to reset the owner and mode with chmod and chown(2). (4) Set the sticky bit again.

Only the super-user can set the sticky bit.
NAME
swapon - specify additional device for paging and swapping

SYNOPSIS
/usr/etc/swapon -a
/usr/etc/swapon name ...

DESCRIPTION
Swapon is used to specify additional devices on which paging and swapping are to take place. The system begins by swapping and paging on only a single device so that only one disk is required at bootstrap time. Calls to swapon normally occur in the system multi-user initialization file /etc/rc making all swap devices available, so that the paging and swapping activity is interleaved across several devices.

Normally, the -a argument is given, causing all devices marked as "sw" swap devices in /etc/fstab to be made available.

The second form gives individual block devices as given in the system swap configuration table. The call makes only this space available to the system for swap allocation.

SEE ALSO
swapon(2), init(8)

FILES
/dev/[ru][pk]?b normal paging devices

BUGS
There is no way to stop paging and swapping on a device. It is therefore not possible to make use of devices which may be dismounted during system operation.
NAME
   sync – update the super block

SYNOPSIS
   sync

DESCRIPTION
   Sync executes the sync system primitive. Sync can be called to insure all disk writes have been completed before the processor is halted in a way not suitably done by reboot(8) or halt(8).

   See sync(2) for details on the system primitive.

SEE ALSO
   sync(2), fsync(2), halt(8), reboot(8)
NAME
syslog - log systems messages

SYNOPSIS
/usr/etc/in.syslog [-mN] [-f name] [-d]

DESCRIPTION
Syslog reads a datagram socket and logs each message it reads into a set of files described by the
configuration file /etc/syslog.conf. Syslog configures when it starts up and whenever it receives a
hangup signal. Syslog logs to the host specified by 'loghost' in the /etc/hosts file. For details on
running syslog in a Sun network environment, see the section, "System Log Configuration" in the
System Set-up and Operation chapter of the System Installation and Maintenance Guide.

Each message logged consists of one line. A message can contain a priority code, marked by a
digit in angle braces at the beginning of the line. Priorities are defined in <syslog.h>, as defined
in the list below. LOG_ALERT is priority 1 (the highest priority) while LOG_DEBUG is priority
9 (the lowest priority).

LOG_ALERT this priority should essentially never be used. It applies only to messages that
are so important that every user should be aware of them, for example, a
serious hardware failure.

LOG_SALERT messages of this priority should be issued only when immediate attention is
needed by a qualified system person, for example, when some valuable system
resource disappears. They get sent to a list of system people.

LOG_EMERG Emergency messages are not sent to users, but represent major conditions.
An example might be hard disk failures. These could be logged in a separate
file so that critical conditions could be easily scanned.

LOG_ERR these represent error conditions, such as soft disk failures, etc.

LOG_CRIT such messages contain critical information, but which cannot be classed as
errors, for example, 'su' attempts. Messages of this priority and higher are
typically logged on the system console.

LOG_WARNING issued when an abnormal condition has been detected, but recovery can take
place.

LOG_NOTICE something that falls in the class of "important information;" this class is
informational but important enough that you don't want to throw items in it
away casually. Messages without any priority assigned to them are typically
mapped into this priority.

LOG_INFO information level messages. These messages could be thrown away without
problems, but should be included if you want to keep a close watch on your
system.

LOG_DEBUG it may be useful to log certain debugging information. Normally this will be
thrown away.

It is expected that the kernel will not log anything below LOG_ERR priority.
The syslog configuration file, etc/syslog.conf, consists of two sections separated by a blank line.
The first section defines files that syslog will log into. Each line contains a single digit which
defines the lowest priority (highest numbered priority) that this file will receive, an optional asterisk
which guarantees that something gets output at least every 20 minutes, and a pathname. The
second part of the file contains a list of users that will be informed on SALER level messages.
For example, the configuration file:

5*/dev/tty8
8/usr/spool/adm/syslog
3/usr/adm/critical

eric
kridle
kalash

logs all messages of priority 5 or higher onto the system console, including timing marks every 20 minutes; all messages of priority 8 or higher into the file /usr/spool/adm/syslog; and all messages of priority 3 or higher into /usr/adm/critical. The users ‘eric’, ‘kridle’, and ‘kalash’ will be informed on any subalert messages.

OPTIONS

-m N Set the mark interval to N (default 20 minutes).
-f name Specify an alternate configuration file.
-d Turn on debugging (if compiled in).
-p port Port number where syslog listens for incoming datagrams. The default port is defined in the 'syslog/udp' entry in the /etc/services file.

To bring syslog down, it should be sent a terminate signal. It logs that it is going down and then waits approximately 10 seconds for any additional messages to come in.

There are some special messages that cause control functions. ‘<*>N’ sets the default message priority to N. ‘<$$>’ causes syslog to reconfigure (equivalent to a hangup signal). This can be used in a shell file run automatically early in the morning to truncate the log.

Syslog creates the file /etc/syslog.pid if possible containing a single line with its process id. This can be used to kill or reconfigure syslog.

FILES

/etc/hosts – the hosts file
/etc/syslog.conf – the configuration file
/etc/syslog.pid – the process id
/etc/services – to find the syslog server’s port number.

BUGS

LOG_ALERT and LOG_SALERT messages should only be allowed to privileged programs.

Actually, syslog is not clever enough to deal with kernel error messages in the current implementation.

SEE ALSO

syslog(3), kill(2)
NAME
telnetd – DARPA TELNET protocol server

SYNOPSIS
/usr/etc/in.telnetd host.port

DESCRIPTION
Telnetd is a server which supports the DARPA standard TELNET virtual terminal protocol. The TELNET server is invoked by inetd(8C) each time there is a connection to the telnet service; see services(5).

Telnetd operates by allocating a pseudo-terminal device (see pty(4)) for a client, then creating a login process which has the slave side of the pseudo-terminal as stdin, stdout, and stderr. Telnetd manipulates the master side of the pseudo terminal, implementing the TELNET protocol and passing characters between the client and login process.

When a TELNET session is started up, telnetd sends a TELNET option to the client side indicating a willingness to do "remote echo" of characters. The pseudo terminal allocated to the client is configured to operate in "cooked" mode, and with XTABS and CRMOD enabled (see tty(4)). Aside from this initial setup, the only mode changes telnetd will carry out are those required for echoing characters at the client side of the connection.

Telnetd supports binary mode, and most of the common TELNET options, but does not, for instance, support timing marks.

SEE ALSO
telnet(1C)

BUGS
A complete list of the options supported should be given here.
NAME
tftpd - DARPA Trivial File Transfer Protocol server

SYNOPSIS
/usr/etc/tftpd [-d] [port]

DESCRIPTION
Tftpd is a server which supports the DARPA Trivial File Transfer Protocol. The TFTP server
operates at the port indicated in the "tftp" service description; see services(5), and is invoked
each time a datagram reaches this port by the internet server inetd(8C).

Due to the lack of authentication information, tftpd will allow only publicly readable files to be
accessed.

SEE ALSO
tftp(1C)

BUGS
This server is known only to be self consistent (i.e. it operates with the user TFTP program,
tftp(1C)).
NAME
    timed – DARPA Time server

SYNOPSIS
    /usr/etc/in.timed

DESCRIPTION
    Timed is a server which supports the DARPA Time Server Protocol. The time server operates at
the port indicated in the “time” service description; see services(5), and is invoked by inetd(8C)
each time there is a connection to the time server.

SEE ALSO
    services(5), rdate(8), inetd(8)

BUGS
    A more sophisticated facility that can accept broadcasts and synchronize clocks over an internet
is needed.
NAME
trpt - transliterate protocol trace

SYNOPSIS
/usr/etc/trpt [-a] [-s] [-t] [-j] [-phex-address] [ system [ core ]]

DESCRIPTION
Trpt interrogates the buffer of TCP trace records created when a socket is marked for 'debugging' (see setsockopt(2)), and prints a readable description of these records. When no options are supplied, trpt prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB).

OPTIONS
-s  Print a detailed description of the packet sequencing information, in addition to the normal output.
-t  Print the values for all timers at each point in the trace, in addition to the normal output.
-j  Just give a list of the protocol control block addresses for which there are trace records.
-phex-address  Show only trace records associated with the protocol control block who's address follows.
-a  in addition to the normal output, print the values of the source and destination addresses for each packet recorded.

The recommended use of trpt is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the -A option to netstat(8). Then run trpt with the -p option, supplying the associated protocol control block addresses. If there are many sockets using the debugging option, the -J option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

FILES
/vmunix
/dev/kmem

SEE ALSO
setsockopt(2), netstat(8)

DIAGNOSTICS
'no namelist' when the system image doesn't contain the proper symbols to find the trace buffer; others which should be self explanatory.

BUGS
Should also print the data for each input or output, but this is not saved in the race record.

The output format is inscrutable, and should be described here.
NAME
tunefs – tune up an existing file system

SYNOPSIS
/etc/tunefs tuneup-options special|filesys

DESCRIPTION
Tunefs is designed to change the dynamic parameters of a file system which affect the layout policies. The parameters which are to be changed are indicated by the flags given below:

-a maxcontig
This specifies the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see -d below). The default value is one, since most device drivers require an interrupt per disk transfer. Device drivers that can chain several buffers together in a single transfer should set this to the maximum chain length.

-d rotdelay
This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

-e maxbpg
This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

-m minfree
This value specifies the percentage of space held back from normal users; the minimum free space threshold. The default value used is 10%. This value can be set to zero, however up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. Note that if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

SEE ALSO
fs(5), newfs(8), mkfs(8)

BUGS
This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the program will only take effect if it is run on dismounted file systems. (if run on the root file system, the system must be rebooted)
NAME
update – periodically update the super block

SYNOPSIS
/etc/update

DESCRIPTION
Update is a program that executes the sync(2) primitive every 30 seconds. This insures that the file system is fairly up to date in case of a crash. This command should not be executed directly, but should be executed out of the initialization shell command file.

SEE ALSO
sync(2), sync(8), init(8)
NAME
uuclean – uucp spool directory clean-up

SYNOPSIS
/usr/lib/uucp/uuclean [-ppre] [-ntime] [-m]

DESCRIPTION
Uuclean scans the spool directory for files with the specified prefix and deletes all those which are older than the specified number of hours.

OPTIONS
-ppre Scan for files with pre as the file prefix. Up to 10 -p arguments may be specified. A -p without any pre following deletes all files older than the specified time.

-n time Files whose age is more than time hours are deleted if the prefix test is satisfied (default time is 72 hours).

-m Send mail to the owner of the file when it is deleted.

Uuclean will typically be started by cron(8).

FILES
/usr/lib/uucp directory with commands used by uuclean internally
/usr/lib/uucp/spool spool directory

SEE ALSO
uucp(1C), uux(1C)
NAME
  uurec — receive processed news articles via mail

SYNOPSIS
  uurec

DESCRIPTION
  uurec reads news articles on the standard input sent by sendnews(8), decodes them, and gives
  them to inews(1) for insertion.

SEE ALSO
  inews(1), readnews(1), recnews(8), sendnews(8), newscheck(1)
NAME
vipw - edit the password file

SYNOPSIS
/etc/vlpw

DESCRIPTION
Vipw edits the password file while setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already being edited, then you will be told to try again later. The vi editor will be used unless the environment variable EDITOR indicates an alternate editor. Vipw performs a number of consistency checks on the password entry for root, and will not allow a password file with a "mangled" root entry to be installed.

SEE ALSO
chsh(1), passwd(1), passwd(5), adduser(8)

FILES
/etc/ptmp
NAME

vmstat - report virtual memory statistics

SYNOPSIS

vmstat [ -fs ] [ interval [ count ] ]

DESCRIPTION

Vmstat delves into the system and normally reports certain statistics kept about process, virtual memory, disk, trap and cpu activity.

Without options, vmstat displays a one-line summary of the virtual memory activity since the system has been booted. If interval is specified, vmstat summarizes activity over the last interval seconds. If a count is given, the statistics are repeated count times.

For example, the following command displays a summary of what the system is doing every five seconds. This is a good choice of printing interval since this is how often some of the statistics are sampled in the system.

    hal% vmstat 5

The fields of vmstat's display are:

procs Reports the number of processes in each of the three following states:
    r in run queue
    b blocked for resources (i/o, paging, etc.)
    w runnable or short sleeper (< 20 secs) but swapped

memory Reports on usage of virtual and real memory. Virtual memory is considered active if it belongs to processes which are running or have run in the last 20 seconds.
    avm number of active virtual Kbytes
    fre size of the free list in Kbytes

page Reports information about page faults and paging activity. The information on each of the following activities is averaged each five seconds, and given in units per second.
    re page reclaims (simulating reference bits) — but see the -S option for how this field is modified.
    at number of attaches — but see the -S option for how this field is modified.
    pi pages paged in
    po pages paged out
    fr pages freed per second
    de anticipated short term memory shortfall
    sr pages scanned by clock algorithm, per-second

disk Reports number of disk operations per second (this field is system dependent). For Sun systems, four slots are available for up to four drives: “x0” (or “s0” for SCSI disks), “x1”, “x2”, and “x3”.

faults Reports trap/interrupt rate averages per second over last 5 seconds.
    in (non clock) device interrupts per second
    sy system calls per second
    cs cpu context switch rate (switches/sec)
**cpu**  Gives a breakdown of percentage usage of CPU time.

**us**  user time for normal and low priority processes

**sy**  system time

**id**  cpu idle

**OPTIONS**

**-f**  Report on the number of *forks* and *uforks* since system startup and the number of pages of virtual memory involved in each kind of fork.

**-s**  Display the contents of the *sum* structure, giving the total number of several kinds of paging-related events which have occurred since boot.

**-S**  Report on swapping rather than paging activity. This option will change two fields in *vmstat*’s “paging” display: rather than the “re” and “at” fields, *vmstat* will report “si” (swap-ins), and “so” (swap-outs).

**FILES**

*/dev/kmem*

*/vmunix*
# Table of Contents

1. Kernel Building Process ................................................................. 1  
  1.1. Quick Summary ........................................................................ 1  
  1.2. Creating a Configuration File .................................................. 2  
  1.3. Constructing Source Code Dependencies .................................. 3  
  1.4. Building the Kernel ................................................................. 3  
  1.5. Sharing Object Modules ........................................................... 4  
  1.6. Building Profiled Kernels .......................................................... 4  
  1.7. Configuring Systems without Source ......................................... 5  
  1.8. Adding New Device Drivers ..................................................... 5  

2. Configuration File Syntax .............................................................. 6  
  2.1. Global Configuration Parameters ............................................ 6  
  2.2. Kernel Image Parameters ........................................................ 7  
  2.3. Device Specifications ............................................................... 8  
  2.4. Pseudo-devices ........................................................................ 9  

A. Configuration File Grammar ......................................................... 10  

B. Rules for Defaulting System Devices .......................................... 12  
  B.1. Multiple Swap/Paging Areas .................................................. 12  
  B.2. System Dumps ......................................................................... 12  

C. Sample Configuration Files .......................................................... 13  

D. Data Structure Sizing Rules ......................................................... 21
Building Sun Workstation Kernels

This document describes the use of the `config(8)` program to configure and create bootable UNIX kernels. It discusses the structure of kernel configuration files, and how to configure kernels with non-standard hardware configurations. An appendix contains a summary of the rules used by the kernel in calculating the size of kernel data structures, and also indicates some of the standard kernel size limitations (and how to change them).

`Config` is a tool used in building new Sun UNIX† kernel images. It takes a file describing a kernel's tunable parameters and hardware support, and generates a collection of files which are then used to build a copy of UNIX appropriate to that configuration. `Config` simplifies kernel maintenance by isolating configuration dependencies in a single, easy to understand, file.

The first division of this document describes building kernels, and the second describes the syntax of the configuration file.

Appendix A gives the grammar for config files. Appendix B describes defaulting rules used during the bootstrap process. Appendix C give sample configuration files. A final appendix gives the default rules used in calculating sizes for the most important kernel data structures. It also lists kernel data structure size limitations and indicates how you can modify these limits.

If you are planning to write a device-driver for UNIX, consult the `Device Driver Manual` in the Sun System Internals Manual for more information.

1. Kernel Building Process

1.1. Quick Summary

Assuming the kernel source is located in the `/sys` directory, there are seven steps in building, installing and running a new kernel:

1) Choose a name for your configuration of the system; for example GAIA. Note that — by convention — the name should be in all uppercase letters.

2) In the `/sys/conf` directory, create the config file for the system and the directory to contain the kernel image:

```
  # cp GENERIC GAIA
  # chmod +w GAIA
  # mkdir ../GAIA
```

† UNIX is a trademark of Bell Laboratories.
Kernel Building Process

3) Edit GAIA to reflect your system, e.g. change the timezone from 8 to 5 (Pacific to Eastern time), delete devices and options which you don't need from the description, and add devices which you have which are not in the configuration.

4) Run config:

```
# /etc/config GAIA
```

5) Build the new kernel:

```
# cd ../GAIA
# make depend
# make
... lots of output ...
```

6) Install the new kernel and try it out

```
# cp vmunix /newvmunix
# /etc/halt >b newvmunix -s
```

7) If the new kernel appears to work, save the old kernel and install the new one in /vmunix.

```
# cd /
# mv vmunix ovmunix
# mv newvmunix vmunix
# /etc/reboot
```

Steps 1 and 2 are usually done only once. When a kernel configuration changes, it usually suffices to just run `config` on the modified configuration file, rebuild the source code dependencies, and remake the kernel.†

### 1.2. Creating a Configuration File

Configuration files normally reside in the subdirectory `/sys/conf`. A configuration file is most easily constructed by copying an existing configuration file and modifying it. This distribution contains a GENERIC configuration file which you can edit to suit your particular system configuration.

The configuration file must have the same name as the directory in which the configured kernel is to be built. Further, `config` assumes this directory is located in the parent directory of the directory in which it is run. For example, the generic kernel has a configuration file `/sys/conf/GENERIC` and an accompanying directory named `/sys/GENERIC`. In general it is unwise to move your configuration directories out of `/sys`, as most of the kernel code and the files created by `config` use pathnames of the form `"../"`. If you are running out of space on the file system where the configuration directories are located, there is a mechanism (for source distributions only) for sharing relocatable object files between kernels. This is described later.

When building your configuration file, be sure to include the items described in section 3. In particular, the machine type, cpu type, timezone, system identifier, maximum users, and root device must be specified. The specification of the hardware present may take a bit of work,

† In rare cases invisible configuration dependencies may exist, requiring the kernel to be rebuilt from scratch by removing the kernel building directory and starting again from step 2. This will be discussed later.
particularly if your hardware is configured at non-standard places (for example, device registers
located at funny places or devices not supported by the kernel). If the devices to be configured
are not already described in one of the existing configuration files, check the section 4 manual
pages in the Sun System Interface Manual. For each supported device, the manual page
synopsis entry gives a sample configuration line.

Once the configuration file is complete, run it through config and look for any errors. Never use
a kernel which config has complained about; the results are unpredictable. For the most part,
config's error diagnostics are self-explanatory. It may be the case that the line numbers given
with the error messages are off by one.

A successful run of config on your configuration file will generate a number of files in the
configuration directory. These files are:

- A file to be used by make(1) in compiling and loading the kernel.
- One file for each possible kernel image for your machine which describes where the root and
  swap devices are located.
- A collection of header files, one per possible device the kernel supports, which define the
  hardware configured.
- A file containing the i/o configuration tables used by the kernel during its autoconfiguration
  phase.

Unless you suspect that there is a bug in config, or are curious about how the kernel's
autoconfiguration scheme works, you should never have to look at any of these files.

1.3. Constructing Source Code Dependencies

When config is done generating the files needed to compile and link your kernel it will terminate
with a message of the form "Don't forget to run make depend". This is a reminder that you
should change your working directory to the configuration directory for the kernel just
configured, and type "make depend" to build the rules used by make to recognize interdepen-
dencies in the kernel source code. This will insure that any changes to a piece of the kernel
source code will result in the proper modules being compiled the next time make is run.

This step is particularly important if your site makes changes to the kernel include files. The
rules generated specify which source code files are dependent on which include files. Without
these rules, make will not recognize when it must rebuild modules due to a kernel header file
being modified. Note that dependency rules created by this step only reflect directly included
files. That is, if file "a" includes another file "b", which includes yet another, say "c", and then
"c" is modified, make will not recognize that "a" should be recompiled. It is best to keep
include file dependencies only one level deep.

1.4. Building the Kernel

The makefile constructed by config should allow a new kernel to be rebuilt by simply typing
"make image-name". For example, if you have named your bootable kernel image "vmunix",
then "make vmunix" will generate a bootable image named "vmunix". Alternate kernel image
names are used when the root file system location and/or swapping configuration is done in
more than one way. The makefile which config creates has entry points for each kernel image
defined in the configuration file. Thus, if you have configured "vmunix" to be a kernel with the
root file system on an "xy" device and "ipvmunix" to be a kernel with the root file system on
an "ip" device, then "make vmunix ipvmunix" will generate binary images for both.

7 January 1984
Building Sun Workstation Kernels

Kernel Building Process

Note that the name of a bootable image is different from the system identifier. All bootable images are configured for the same devices; only the information about the root file system and paging devices differ.

The last step in the kernel building process is to rearrange certain commonly used symbols in the symbol table of the kernel image; the makefile generated by `config` does this automatically for you. This is advantageous for programs such as `ps(1)` and `vmstat(8)`, which run much faster when the symbols they need are located at the front of the symbol table. Remember also that programs such as `ps` and `vmstat` expect the currently executing kernel to be named “/vmunix". If you install a new kernel and name it something other than “/vmunix”, these programs are likely to produce incorrect results.

1.5. Sharing Object Modules

This is only effective if you have a source distribution.

If you have many kernels which are all built on a single machine there are at least two approaches to saving time in building kernel images. The best way is to have a single kernel image which is run on all machines. This is attractive since it minimizes disk space used and time required to rebuild kernels after making changes. However, it is often the case that one or more systems will require a separately configured kernel image. This may be due to limited memory (building a kernel with many unused device drivers wastes core), or to configuration requirements (one machine may be a development machine where disk quotas are not needed, while another is a production machine where they are), etc. In these cases it is possible for kernels to share relocatable object modules which are not configuration dependent; most of the modules in the directory `/sys/...` are of this sort.

To share object modules, first build a GENERIC kernel. Then, for each kernel, configure as before, but before recompiling and linking, type “make links”. This will cause the kernel source to be searched for source modules which are safe to share between kernels and generate symbolic links in the current directory to the appropriate object modules in the directory `/GENERIC`. A shell script, “makelinks” is generated with this request and may be checked for correctness. The file `/sys/conf/defines` contains a list of symbols which we believe are safe to ignore when checking the source code for modules which may be shared. Note that this list includes the definitions used to conditionally compile in the virtual memory tracing facilities, and the trace point support used only rarely. It may be necessary to modify this list to reflect local needs. Also, as described previously, interdependencies which are not directly visible in the source code are not caught. Thus if you place per-system dependencies in an include file, they will not be recognized.

1.6. Building Profiled Kernels

This is only effective if you have a source distribution.

It is simple to configure a kernel which will automatically collect profiling information as it operates. The profiling data may be collected with `kgmon(8)` and processed with `gprof(1)` to obtain information regarding the kernel's operation. Profiled kernels maintain histograms of the program counter as well as the number of invocations of each routine. The `gprof(1)` command will generate a dynamic call graph of the executing kernel and propagate time spent in each routine along the arcs of the call graph.
To configure a profiled kernel, use the \texttt{-p} option with \texttt{config}. A profiled kernel is about 5-10\% larger in its text space due to the calls to count the subroutine invocations. When the kernel executes, the profiling data is stored in a buffer which is 1.2 times the size of the text space. The overhead for running a profiled kernel varies; under normal load we see anywhere from 5-25\% of the kernel time spent in the profiling code.

Note that kernels configured for profiling should not be shared as described above unless all the other shared kernels are also to be profiled.

\section*{1.7. Configuring Systems without Source}

Object only releases have binaries for standard system modules in the directory \texttt{/sys/OBJ}. Using these binaries you can create new configurations and add new device drivers to the kernel. The following lines from the GENERIC config file must be in every config file for object-only distributions:

```
  machine  sun
  cpu  "SUN2"
  options  "INET"
  pseudo-device  inet
  pseudo-device  ether
  pseudo-device  loop
  controller  \\
```

If you include these lines you can make any changes you wish to the configuration file, provided you do not configure in more devices of a particular type than are allowed by the distributed object code in ..\texttt{/OBJ}. Attempting to do so will not be detected and may cause the kernel to appear to work but have only occasional failures. Double check the \texttt{.h} files in ..\texttt{/OBJ} if you change the number of devices configured for any standard drivers.

\section*{1.8. Adding New Device Drivers}

New device drivers require entries in the files \texttt{/sys/sun/conf.c}, \texttt{/sys/conf/files.sun}, and possibly \texttt{/sys/sun/swappositive.c} and \texttt{sun/devices.sun}. New devices also require one or more new special files to be added to the \texttt{/dev} directory. See the \textit{Device Driver Manual} in the \textit{System Internals Manual for the Sun UNIX System}. 7 January 1984
2. Configuration File Syntax

In this section we consider the specific rules used in writing a configuration file. Appendix A gives a complete grammar for the input language; use it if you have problems with syntax errors.

A configuration file has three logical parts:
• configuration parameters global to all kernel images specified in the configuration file,
• parameters specific to each kernel image to be generated, and
• device specifications.

2.1. Global Configuration Parameters

The global configuration parameters are the type of machine, cpu types, options, timezone, system identifier, and maximum users. Each is specified with a separate line in the configuration file.

machine type
The system is to run on the machine type specified. No more than one machine type can appear in the configuration file. The appropriate line for the sun is machine sun.

cpu "type"
This system is to run on the cpu type specified. The appropriate line is cpu "SUN2".

options optionlist
Compile the listed optional code into the system. Options in this list are separated by commas. Possible options are listed at the top of the generic makefile. A line of the form "options FUNNY,HAHA" generates global "#define"s -DFUNNY -DHAHA in the resultant makefile. An option may be given a value by following its name with "="; then the value enclosed in (double) quotes.

timezone number [ dst [ number ] ]
Specifies the timezone you are in. This is measured in the number of hours your timezone is west of GMT. EST is 5 hours west of GMT, PST is 8. Negative numbers indicate hours east of GMT. If you specify dst, the kernel will operate under daylight savings time. An optional integer or floating point number may be included to specify a particular daylight saving time correction algorithm; the default value is 1, indicating the United States. Other values are: 2 (Australian style), 3 (Western European), 4 (Middle European), and 5 (Eastern European). See gettimeofday(2) and ctime(3) for more information.

ident name
Gives the system identifier – a name for the machine or machines to run this kernel. name must be enclosed in double quotes if it contains both letters and numbers.

maxusers number
The maximum expected number of simultaneously active users on this kernel is number. This number is used to size several kernel data structures. Typical values are 2 for 1 Megabyte memory single-user Sun Workstation, 4 for Sun Workstations with 2 Megabytes, and 8 for Sun servers.
2.2. Kernel Image Parameters

Multiple bootable images may be specified in a single configuration file. The kernels will have the same global configuration parameters and devices, but the location of the root file system and other kernel-specific devices may be different. A kernel image is specified with a "config" line:

```
config kername config-clauses
```

The `kername` field is the name given to the loaded kernel image; the standard kernel image is "vmunix". The configuration clauses are one or more specifications indicating where the root file system is located, how many paging devices there are and where they go.

A configuration clause is one of the following:

```
root [ on ] root-device
swap [ on ] swap-device [ and swap-device ]
dumps [ on ] dump-device
args [ on ] arg-device
```

(the "on" is optional.) Multiple configuration clauses are separated by white space; `config` allows specifications to be continued across multiple lines by beginning the continuation line with a tab character. The "root" clause specifies where the root file system is located, the "swap" clause indicates swapping and paging area(s), and the "dumps" clause can be used to force crash dumps to be taken on a particular device, and the "args" clause can be used to specify that argument list processing for `execve` should be done on a particular disk.

The device names supplied in the clauses may be fully specified — as a device, unit, and file system partition — or underspecified. If underspecified, `config` will use builtin rules to select default unit numbers and file system partitions. The defaulting rules are dependent on the overall system configuration. For example, the swap area need not be specified at all if the root device is specified; in this case the swap area is placed in the "b" partition of the same disk where the root file system is located. Appendix B contains a complete list of the defaulting rules used in selecting devices.

The device names are translated to the appropriate major and minor device numbers on a per-machine basis. A file, `/sys/conf/devices.machine` (where "machine" is the machine type specified in the configuration file), is used to map a device name to its major block device number. The minor device number is calculated using the standard disk partitioning rules.

If the default mapping of device name to major/minor device number is incorrect for your configuration, it can be replaced by an explicit specification of the major/minor device. This is done by substituting

```
major x minor y
```

where the device name would normally be found. For example,

```
config vmunix root on major 99 minor 1
```

Normally, the areas configured for swap space are sized by the kernel at boot time. If a non-standard partition size is to be used for one or more swap areas, this can also be specified. To do this, the device name specified for a swap area should have a "size" specification appended. For example,

```
config vmunix root on xy0 swap on xy0b size 12000
```

would force swapping to be done in partition "b" of "xy0" and the swap partition size would be set to 1200 sectors. A swap area sized larger than the associated disk partition is trimmed to the partition size.
Configuration File Syntax

Building Sun Workstation Kernels

To create a generic configuration, only the clause "swap generic" should be specified; any extra clauses cause an error. Note that if you use the "swap generic" clause, your system identification line must read "ident GENERIC".

2.3. Device Specifications

Each device attached to a machine must be specified to config so that the system generated will know to probe for it during the autoconfiguration process carried out at boot time. Hardware specified in the configuration need not actually be present on the machine where the generated system is to be run. Only the hardware actually found at boot time will be used by the system.

A device specification takes one of the following forms:

- `controller device-name device-info [ interrupt-spec ]`
- `device device-name device-info interrupt-spec`
- `disk device-name device-info`
- `tape device-name device-info`

A "controller" is a disk or tape controller. A "device" is an autonomous device which connects directly to the Multibus. "Disk" and "tape" identify disk drives and tape drives connected to a "controller".

device-name is a standard UNIX device name (see the section 4 pages of the System Interface Manual) concatenated with the logical unit number to be assigned the device. The logical unit number may be different than the physical unit number indicated on the front of something like a disk; the logical unit number is used to refer to the UNIX device, not the physical unit number. The device-info clause specifies how the hardware is connected in the interconnection hierarchy. On a Sun Workstation, the following specification should be used:

```plaintext
controller mb0 at nexus ?
```

The remaining interconnections on the Sun Workstation are:

- a controller may be connected to the Multibus (mb0),
- a disk or tape is always attached to a controller, and
- devices may be attached to controllers or to the Multibus.

For controllers, the control status register must be given explicitly, as well the interrupt priority level of the device. The following lines give an example of each of these interconnections:

```plaintext
controller xyc0 at mb0 csr 0xee40 priority 2
disk xy0 at xyc0 drive 0
device cg0 at mb0 csr 0xe8000 priority 3
```

Certain device drivers require extra information passed to them at boot time to tailor their operation to the actual hardware present. For example, the drivers for the terminal multiplexors need to know which lines are attached to modem lines so that no one will be allowed to use them unless a connection is present. For this reason, one last parameter may be specified to a device, a flags field. It has the syntax

```plaintext
flags number
```

and is usually placed after the csr specification. The number is passed directly to the associated driver. The manual pages in section 4 should be consulted to determine how each driver uses this value (if at all).

7 January 1984
2.4. **Pseudo-devices**

A number of drivers and software subsystems are treated like device drivers without any associated hardware. To include any of these pieces, a "pseudo-device" specification must be used. A specification for a pseudo device takes the form

```
pseudo-device device-name [ howmany ]
```

Examples of pseudo devices are the pseudo terminal driver (see *pty*(4)), where the optional *howmany* value indicates the number of pseudo terminals to configure, and the Sun Window System (see *win*(4)).
Appendix A. Configuration File Grammar

The following grammar is a compressed form of the actual yacc(1) grammar used by config to parse configuration files. Terminal symbols are shown all in upper case, literals are emboldened; optional clauses are enclosed in brackets, "[" and "]"; zero or more instantiations are denoted with "*".

Configuration ::= [ Spec ; ]*

Spec ::= Config_spec
    | Device_spec
    | trace
    | /* lambda */

/* configuration specifications */

Config_spec ::= machine ID
    | cpu ID
    | options Opt_list
    | ident ID
    | System_spec
    | timezone [ - ] NUMBER [ dst [ NUMBER ] ]
    | timezone [ - ] FPNUMBER [ dst [ NUMBER ] ]
    | maxusers NUMBER

/* system configuration specifications */

System_spec ::= config ID System_parameter [ System_parameter ]*

System_parameter ::= swap_spec | root_spec | dump_spec | arg_spec

swap_spec ::= swap [ on ] swap_dev [ and swap_dev ]*

swap_dev ::= dev_spec [ size NUMBER ]

root_spec ::= root [ on ] dev_spec

dump_spec ::= dumps [ on ] dev_spec

arg_spec ::= args [ on ] dev_spec

dev_spec ::= dev_name | major_minor

major_minor ::= major NUMBER minor NUMBER

dev_name ::= ID [ NUMBER [ ID ]]

/* option specifications */

Opt_list ::= Option [ , Option ]*
Option ::= ID [ = Opt_value ]

Opt_value ::= ID | NUMBER

/* device specifications */

Device_spec ::= device Dev_name Dev_info Int_spec
   | disk Dev_name Dev_info
   | tape Dev_name Dev_info
   | controller Dev_name Dev_info [ Int_spec ]
   | pseudo-device Dev [ NUMBER ]

Dev_name ::= Dev NUMBER

Dev ::= uba | mba | ID

Dev_info ::= Con_info [ Info ]*

Con_info ::= at Dev NUMBER
   | at nexus NUMBER

Info ::= car NUMBER
   | drive NUMBER
   | slave NUMBER
   | flags NUMBER

Int_spec ::= vector ID [ ID ]*
   | priority NUMBER

Lexical Conventions
The terminal symbols are loosely defined as:

ID
   One or more alphabetics, either upper or lower case, and underscore, "_".

NUMBER
   Approximately the C language specification for an integer number. That is, a leading "0x" indicates a hexadecimal value, a leading "0" indicates an octal value, otherwise the number is expected to be a decimal value. Hexadecimal numbers may use either upper or lower case alphabetics.

FPNUMBER
   A floating point number without exponent. That is a number of the form "nnn.ddd", where the fractional component is optional.

In special instances a question mark, "?", can be substituted for a "NUMBER" token. This is used to effect wildcarding in device interconnection specifications.

Comments in configuration files are indicated by a "#" character at the beginning of the line; the remainder of the line is discarded.

A specification is interpreted as a continuation of the previous line if the first character of the line is tab.

7 January 1984
Appendix B. Rules for Defaulting System Devices

When `config` processes a "config" rule which does not fully specify the location of the root file system, paging area(s), device for system dumps, and device for argument list processing it applies a set of rules to define those values left unspecified. The following list of rules are used in defaulting system devices.

1) If a root device is not specified, the swap specification must indicate a "generic" system is to be built.
2) If the root device does not specify a unit number, it defaults to unit 0.
3) If the root device does not include a partition specification, it defaults to the "a" partition.
4) If no swap area is specified, it defaults to the "b" partition of the root device.
5) If no device is specified for processing argument lists, the first swap partition is selected.
6) If no device is chosen for system dumps, the first swap partition is selected (see below to find out where dumps are placed within the partition).

The following table summarizes the default partitions selected when a device specification is incomplete, e.g. "hp0".

<table>
<thead>
<tr>
<th>Type</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>&quot;a&quot;</td>
</tr>
<tr>
<td>swap</td>
<td>&quot;b&quot;</td>
</tr>
<tr>
<td>args</td>
<td>&quot;b&quot;</td>
</tr>
<tr>
<td>dumps</td>
<td>&quot;b&quot;</td>
</tr>
</tbody>
</table>

B.1. Multiple Swap/Paging Areas

When multiple swap partitions are specified, the system treats the first specified as a "primary" swap area which is always used. The remaining partitions are then interleaved into the paging system at the time a `swapon(2)` system call is made. This is normally done at boot time with a call to `swapon(8)` from the `/etc/rc` file.

B.2. System Dumps

System dumps are automatically taken after a system crash, provided the device driver for the "dumps" device supports this. The dump contains the contents of memory, but not the swap areas. Normally the dump device is a disk in which case the information is copied to a location near the back of the partition. The dump is placed in the back of the partition because the primary swap and dump device are commonly the same device and this allows the system to be rebooted without immediately overwriting the saved information. When a dump has occurred, the system variable `dumps` is set to a non-zero value indicating the size (in bytes) of the dump. The `savecore(8)` program then copies the information from the dump partition to a file in a "crash" directory and also makes a copy of the system which was running at the time of the crash (usually `/vmunix`).
Appendix C. Sample Configuration Files

The following pages present several sample configuration files.

The following GENERIC configuration file is used to build the release kernel. Lines are interpreted in #comments#. Lines marked #mandatory# must be included in every configuration file; lines which describe devices which your system does not have should be deleted when you produce your system configuration file.

```
# GENERIC SUN
#
machine sun          #Mandatory
cpu "SUN2"           #Mandatory
ident GENERIC       #Mandatory
timezone 8 dst      #Mandatory Number and "dst" are variable
maxusers 2          #Mandatory Number may vary
options INET        #Mandatory INET means include Internet code
options SYSACCT     #Optional; include only with pseudo-device sysacct.

config vmunix swap generic     #Mandatory Specify root and swap devices
pseudo-device pty          #Pseudo-tty's. Needed for network or window system.
pseudo-device bk           #Berkenet line discipline for high speed tty input - see bk(4).
pseudo-device sysacct      #Include only with SYSACCT options clause, above.
pseudo-device inet         #Internet code - see inet(4).
pseudo-device ether        #ARP code. Must include if using Ethernet - see arp(4).
pseudo-device loop         #Software loop back network device driver; must include if INET - see lo(4).
pseudo-device nd           #Network disk. Needed if server or diskless - see nd(4).
pseudo-device win128        #Window system. Number indicates max windows. Must include dtop, below.
pseudo-device dtop4        #Max Screens (‘desktops’); required for window system.
pseudo-device ms3          #Max Mice; required for window system - see ms(4).
pseudo-device kb3          #Max Sun keyboards; required if using any Sun keyboard; omit if using serial terminal for console.
pseudo-device ingres       #Ingres lock device
controller mb0 at nexus    #Multibus code.
controller ip0 at mb0 csr0x40 priority 2    #1st Interphase controller - see ip(4).
controller ip1 at mb0 csr0x44 priority 2    #2nd Interphase controller.
disk ip0 at ip0 drive 0    #1st disk on 1st Interphase controller.
disk ip1 at ip0 drive 1    #2nd disk on 1st Interphase controller.
disk ip2 at ip1 drive 0    #1st disk on 2nd Interphase controller.
disk ip3 at ip1 drive 1    #2nd disk on 2nd Interphase controller.
```

7 January 1984
### Sample Configuration Files

<table>
<thead>
<tr>
<th>Device</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>controller</td>
<td><code>xy0 at mb0 csr 0xee40 priority 2</code></td>
</tr>
<tr>
<td>controller</td>
<td><code>xy1 at mb0 csr 0xee48 priority 2</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>xy0 at xy0 drive 0</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>xy1 at xy0 drive 1</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>xy2 at xy1 drive 0</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>xy3 at xy1 drive 1</code></td>
</tr>
<tr>
<td>controller</td>
<td><code>sc0 at mb0 csr 0x80000 priority 2</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>sd0 at sc0 drive 0 flags 0</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>sd1 at sc0 drive 1 flags 0</code></td>
</tr>
<tr>
<td>tape</td>
<td><code>st0 at sc0 drive 32 flags 1</code></td>
</tr>
<tr>
<td>controller</td>
<td><code>sc1 at mb0 csr 0x84000 priority 2</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>sd2 at sc1 drive 0 flags 0</code></td>
</tr>
<tr>
<td>disk</td>
<td><code>sd3 at sc1 drive 1 flags 0</code></td>
</tr>
<tr>
<td>tape</td>
<td><code>st1 at sc1 drive 32 flags 1</code></td>
</tr>
<tr>
<td>device</td>
<td><code>ropc0 at mb0 csr 0xee0800 priority 1</code></td>
</tr>
<tr>
<td>device</td>
<td><code>sky0 at mb0 csr 0x20000 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>zs0 at mb0 csr 0xee0800 flags 3 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>zs1 at mb0 csr 0xee000 flags 3 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>zs2 at mb0 csr 0x80800 flags 3 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>zs3 at mb0 csr 0x81000 flags 3 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>zs4 at mb0 csr 0x84800 flags 3 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>zs5 at mb0 csr 0x85000 flags 3 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>oct0 at mb0 csr 0x520 flags 0xff priority 4</code></td>
</tr>
<tr>
<td>device</td>
<td><code>mti0 at mb0 csr 0x620 flags 0xff priority 4</code></td>
</tr>
<tr>
<td>device</td>
<td><code>iel0 at mb0 csr 0x88000 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>iel1 at mb0 csr 0x8c000 flags 2 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>iel2 at mb0 csr 0x8e000 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>ec0 at mb0 csr 0xe0000000 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>ec1 at mb0 csr 0xe20000000 priority 3</code></td>
</tr>
<tr>
<td>controller</td>
<td><code>tm0 at mb0 csr 0xa0 priority 3</code></td>
</tr>
<tr>
<td>controller</td>
<td><code>tm1 at mb0 csr 0xa2 priority 3</code></td>
</tr>
<tr>
<td>tape</td>
<td><code>mt0 at tm0 drive 0 flags 1</code></td>
</tr>
<tr>
<td>tape</td>
<td><code>mt1 at tm1 drive 0 flags 1</code></td>
</tr>
<tr>
<td>device</td>
<td><code>ar0 at mb0 csr 0x200 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>ar1 at mb0 csr 0x208 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>cgon0 at mb0 csr 0xe8000 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>bwto0 at mb0 csr 0x7000000 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>bwone0 at mb0 csr 0x2e000000 priority 3</code></td>
</tr>
<tr>
<td>device</td>
<td><code>vpo at mb0 csr 0x400 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>vpc0 at mb0 csr 0x480 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>vpc0 at mb0 csr 0x500 priority 2</code></td>
</tr>
<tr>
<td>device</td>
<td><code>pii0 at mb0 csr 0xee2000 priority 1</code></td>
</tr>
</tbody>
</table>

### Building Sun Workstation Kernels

- **1st Xylogics controller** - see `xy(4)`.#
- **2nd Xylogics controller**
- **1st disk on 1st Xylogics controller**
- **2nd disk on 1st Xylogics controller**
- **1st disk on 2nd Xylogics controller**
- **2nd disk on 2nd Xylogics controller**
- **1st SCSI controller**
- **2nd SCSI controller**
- **1st SCSI tape**
- **2nd SCSI controller**
- **1st disk on 2nd SCSI controller**
- **2nd disk on 2nd SCSI controller**
- **SCSI tape**
- **Mandatory Raster Op chip** - see `ropc(4)`.
- **Sky Floating Point board**
- **CPU ports** - see `zc(4)`.
- **Sun-2 Video Board ports**; required if using Sun-2 keyboard and mouse.
- **1st SCS Board ports**
- **1st SCS Board ports**
- **2nd SCS Board ports**
- **2nd SCS Board ports**
- **Central Data Octal Card** - see `oct(4)`.
- **Systech terminal MUX** — see `mti(4)`.
- **1st Sun-2 Ethernet Controller**
- **2nd Sun-2 Ethernet Controller**
- **1st 3COM Ethernet Controller** - see `ec(4)`
- **2nd 3COM Ethernet Controller**
- **1st TAPEMASTER tape controller** - see `tm(4)`
- **2nd TAPEMASTER tape controller**
- **1st 1/2" tape drive on 1st controller**
- **1st 1/2" tape drive on 2nd controller**
- **2nd 1/4" tape drive**
- **1st Sun Color Board** - see `cp(4)`.
- **1st monochrome Sun-2 monitor**
- **1st monochrome Sun-1 monitor**
- **1st monochrome VersaCebo Board** - see `vp(4)`.
- **1st Systech Centronics/VersaCebo Board** - see `vpc(4s)`.
- **2nd Systech Centronics/VersaCebo Board**
- **Parallel input. Only used on Sun Models 100U and 150U, for keyboard and mouse.**
- **Not needed if using terminal for console.**

---

**7 January 1984**
Building Sun Workstation Kernels

Sample Configuration Files

# Diskless Model 100
#
machine sun
cpu "SUN2"
ident "ND100"
timezone 8 dst
maxusers 2
options INET

config vmunix root on nd

pseudo-device pty
pseudo-device inet
pseudo-device ether
pseudo-device loop
pseudo-device nd
pseudo-device win32
pseudo-device dttop1
pseudo-device ms1
pseudo-device kb1
controller mb0 at nexus ?
device ropc0 at mb0 csr 0xee0800 priority 1
device zs0 at mb0 csr 0xee800 flags 3 priority 2 # cpu ports
device ec0 at mb0 csr 0xe0000 priority 3
device bwone0 at mb0 csr 0xc0000 priority 3
device pi0 at mb0 csr 0xee2000 priority 1

7 January 1984
Sample Configuration Files

Building Sun Workstation Kernels

# Diskless Model 120
#
machine       sun
cpu           "SUN2"
ident         "ND120"
timezone      8 dst
maxusers      2
options       INET

cfg            vmunix root on nd

pseudo-device pty
pseudo-device inet
pseudo-device ether
pseudo-device loop
pseudo-device nd
pseudo-device win32
pseudo-device dtopr
pseudo-device ms1
pseudo-device kb1
controller mb0 at nexus ?
device        ropc0 at mb0 csr 0xee0800 priority 1
device        sky0 at mb0 csr 0x2000 priority 2
device        zs0 at mb0 csr 0xee0800 priority 2 # cpu ports
device        zs1 at mb0 csr 0xeec000 priority 2 # video ports
device        ec0 at mb0 csr 0xe0000 priority 3
device        ie0 at mb0 csr 0x88000 priority 3
device        bwtwo0 at mb0 csr 0x700000 priority 3

7 January 1984
# Model 120 with one SCSI disk and tape
#
machine sun
cpu "SUN2"
ident "SDST120"
timezone 8 dst
maxusers 2
options INET

cfg config vmunix  root on sd

pseudo-device pty
pseudo-device inet
pseudo-device ether
pseudo-device loop
pseudo-device win32
pseudo-device dtop1
pseudo-device ms1
pseudo-device kb1
controller mb0 at nexus?
controller sc0 at mb0 csr 0x80000 priority 2
disk sd0 at sc0 drive 0 flags 0
tape st0 at sc0 drive 32 flags 1
device ropc0 at mb0 csr 0xeee000 priority 1
device sky0 at mb0 csr 0x2000 priority 2
device zs0 at mb0 csr 0xeec800 flags 3 priority 2 # cpu ports
device zs1 at mb0 csr 0xeec000 flags 3 priority 2 # video ports
device zs2 at mb0 csr 0x80800 flags 3 priority 2
device zs3 at mb0 csr 0x81000 flags 3 priority 2
device ec0 at mb0 csr 0xe00000 priority 3
device ie0 at mb0 csr 0x88000 priority 3
device bwtwo0 at mb0 csr 0x700000 priority 3

7 January 1984
# Model 100 with one Xylogics disk

machine sun
cpu "SUN2"
ident "XY100"
timezone 8 dst
maxusers 2
options INET

cfg vmunix root on xy

pseudo-device pty
pseudo-device inet
pseudo-device ether
pseudo-device loop
pseudo-device win32
pseudo-device dtop1
pseudo-device ms1
pseudo-device kb1
controller mb0 at nexus
controller xyc0 at mb0 csr 0xee40 priority 2
disk xy0 at xyc0 drive 0
device ropc0 at mb0 csr 0xee0800 priority 1
device zs0 at mb0 csr 0xee8000 flags 3 priority 2 # cpu ports
device ec0 at mb0 csr 0xe00000 priority 3
device bwone0 at mb0 csr 0xc00000 priority 3
device pi0 at mb0 csr 0xee2000 priority 1
# Model 100 with one Xylogics disk and Archive tape

machine     sun
cpu         "SUN2"
ident       "XYAR100"
timezone    8 dst
maxusers    2
options     INET

config      vmunix root on xy

pseudo-device pty
pseudo-device inet
pseudo-device ether
pseudo-device loop
pseudo-device win32
pseudo-device dtop1
pseudo-device ms1
pseudo-device kb1
controller   mb0 at nexus ?
controller   xyc0 at mb0 csr 0xee40 priority 2
disk        xy0 at xyc0 drive 0
device      ropc0 at mb0 csr 0xee0800 priority 1
device      zso at mb0 csr 0xeec800 flags 3 priority 2 # cpu ports
device      ec0 at mb0 csr 0xe0000 priority 3
device      ar0 at mb0 csr 0x200 priority 3
device      bwone0 at mb0 csr 0xc0000 priority 3
device      pi0 at mb0 csr 0xee2000 priority 1
# Model 150 server with two Xylogics disks and one 1/2" tape
#
machine sun
cpu "SUN2"
ident "XYMT150"
timezone 8 dst
maxusers 8
options INET
options SYSACCT

cfg

config vmunix root on xy

pseudo-device pty
pseudo-device bk
pseudo-device sysacct
pseudo-device inet
pseudo-device ether
pseudo-device loop
pseudo-device nd
pseudo-device win32
pseudo-device dtop1
pseudo-device ms1
pseudo-device kb1
controller mb0 at nexus ?
controller xyc0 at mb0 csr 0xee40 priority 2
disk xy0 at xyc0 drive 0
disk xy1 at xyc0 drive 1
device ropc0 at mb0 csr 0xee0800 priority 1
device sky0 at mb0 csr 0x2000 priority 2
device zso at mb0 csr 0xeec800 flags 3 priority 2 # cpu ports
device mt0 at mb0 csr 0x620 flags 0xff priority 4
device ec0 at mb0 csr 0xe0000 priority 3
device ec1 at mb0 csr 0xe2000 priority 3
device ic0 at mb0 csr 0x88000 priority 3
controller tm0 at mb0 csr 0xa0 priority 3
tape mt0 at tm0 drive 0 flags 1
device cgone0 at mb0 csr 0xee8000 priority 3
device bwone0 at mb0 csr 0xce0000 priority 3
device pi0 at mb0 csr 0xee2000 priority 1

7 January 1984
Appendix D. Data Structure Sizing Rules

Certain kernel data structures are sized at compile time according to the maximum number of simultaneous users expected, while others are calculated at boot time based on the physical resources present; e.g. memory. This appendix lists both sets of rules and also includes some hints on changing built-in limitations on certain data structures.

D.1. Compile Time Rules

The file `sys/conf/param.c` contains the definitions of almost all data structures sized at compile time. This file is copied into the directory of each configured kernel to allow configuration-dependent rules and values to be maintained. The rules implied by its contents are summarized below (here MAXUSERS refers to the value defined in the configuration file in the "maxusers" rule).

\[ \text{nproc} \]

The maximum number of processes which may be running at any time. It is defined to be \(20 + 8 \times \text{MAXUSERS}\) and referred to in other calculations as NPROC.

\[ \text{ntext} \]

The maximum number of active shared text segments. Defined as \(24 + \text{MAXUSERS}\).

\[ \text{ninode} \]

The maximum number of files in the file system which may be active at any time. This includes files in use by users, as well as directory files being read or written by the system and files associated with bound sockets in the UNIX ipc domain. This is defined as \((\text{NPROC} + 16 + \text{MAXUSERS}) + 32\).

\[ \text{nfile} \]

The number of "file table" structures. One file table structure is used for each open, unshared, file descriptor. Multiple file descriptors may reference a single file table entry when they are created through a `dup` call, or as the result of a `fork`. This is defined to be \[
16 \times (\text{NPROC} + 16 + \text{MAXUSERS}) / 10 + 32
\]

\[ \text{ncallout} \]

The number of "callout" structures. One callout structure is used per internal system event handled with a timeout. Timeouts are used for terminal delays, watchdog routines in device drivers, protocol timeout processing, etc. This is defined as \(16 + \text{NPROC}\).

\[ \text{nclist} \]

The number of "c-list" structures. C-list structures are used in terminal i/o. This is defined as \(100 + 16 \times \text{MAXUSERS}\).

\[ \text{nmbclusters} \]

The maximum number of pages which may be allocated by the network. This is defined as \(256\) (a quarter megabyte of memory) in `/sys/h/mbuf.h`. In practice, the network rarely uses this much memory. It starts off by allocating 64 kilobytes of memory, then requesting more as required. This value represents an upper bound.

\[ \text{nquota} \]

The number of "quota" structures allocated. Quota structures are present only when disc quotas are configured in the system. One quota structure is kept per user. This is defined to be \((\text{MAXUSERS} \times 9) / 7 + 3\).
ndquot
The number of "dquot" structures allocated. Dquot structures are present only when disc quotas are configured in the system. One dquot structure is required per user, per active file system quota. That is, when a user manipulates a file on a file system on which quotas are enabled, the information regarding the user's quotas on that file system must be in-core. This information is cached, so that not all information must be present in-core all the time. This is defined as \((\text{MAXUSERS} \times \text{N_MOUNT}) / 4 + \text{NPROC}\), where N_MOUNT is the maximum number of mountable file systems.

D.2. Run-time Calculations
The most important data structure sized at run-time is the file system buffer cache. The system allocates 10% of each half-megabyte after the first half-megabyte to the cache. Thus on a 1 Megabyte machine, 50 kilobytes is allocated to the cache, while on a 2 Megabyte machine, 150 kilobytes is allocated to the cache. In any case, not less than 16 pages of file system buffers is allocated.

The number of buffers to be allocated can be forced to a specific value by patching the kernel variable nbuf with adb:

```
# adb -w /vmunix
nbuf?W 0=32
nbuf: 0 = 20
$q
```

sets the number of buffers to be 32 (decimal) independent of the amount of main memory available. Reboot after performing this series.

D.3. System Size Limitations
Because the file system block numbers are stored in page table pg_blkno entries, the maximum size of a file system is limited to \(2^{19} 1024\) byte blocks. Thus no file system can be larger than 512M bytes.

The count of mountable file systems is limited to 15. This should be sufficient. If you have many disks it makes sense to make some of them single file systems, and the paging areas don't count in this total. To increase this, you must change the core-map (only if you have source) /sys/h/cmap.h, since there is a 4 bit field used here. The size of the core-map will then expand to 16 bytes per 2048 byte page. Don't forget to change MSWAPX and N_MOUNT in /sys/h/param.h also.

The maximum value NOFILE (open files per process limit) can be raised to is 30 because of a bit field in the page table entry in /sys/machine/pte.h.
# Fsck — The UNIX File System Check Program

## Table of Contents

1. Overview of the File System ................................................................. 1
   1.1. Superblock .................................................................................. 1
   1.2. Summary Information ................................................................. 2
   1.3. Cylinder Groups ........................................................................ 2
   1.4. Fragments .................................................................................. 2
   1.5. Updates to the File System .......................................................... 3

2. Fixing Corrupted File Systems ............................................................ 4
   2.1. Detecting and Correcting Corruption ......................................... 4
   2.2. Super-block Checking ................................................................. 4
   2.3. Free Block Checking ................................................................. 5
   2.4. Checking the Inode State ........................................................... 5
   2.5. Inode Links .............................................................................. 5
   2.6. Inode Data Size ....................................................................... 6
   2.7. Checking the Data Associated with an Inode ............................ 6
   2.8. File System Connectivity ........................................................... 7

A. Fsck Error Conditions .......................................................................... 8
   A.1. Conventions .............................................................................. 8
   A.2. Initialization ............................................................................. 8
   A.3. Phase 1 Check Blocks and Sizes .............................................. 10
   A.4. Phase 1B: Rescan for More Dup's ........................................... 12
   A.5. Phase 2 Check Pathnames ....................................................... 12
   A.6. Phase 3 Check Connectivity ..................................................... 14
   A.7. Phase 4 Check Reference Counts ............................................ 15
   A.8. Phase 5 - Check Cyl Groups .................................................. 18
   A.9. Phase 6 - Salvage Cylinder Groups ....................................... 19
   A.10. Cleanup ................................................................................. 19
Fsck – The UNIX File System Check Program

When a UNIX operating system is brought up, a consistency check of the file systems should always be performed. This precautionary measure helps to insure a reliable environment for file storage on disk. If an inconsistency is discovered, corrective action must be taken. Fsck runs in two modes. Normally it is run non-interactively by the system after a normal boot. When running in this mode, it will only make changes to the file system that are known to always be correct. If an unexpected inconsistency is found, Fsck will exit with a non-zero exit status, leaving the system running single-user. Typically the operator then runs Fsck interactively. When running in this mode, each problem is listed followed by a suggested corrective action. The operator must decide whether or not the suggested correction should be made.

The purpose of this memo is to dispel the mystique surrounding file system inconsistencies. It first describes the updating of the file system (the calm before the storm) and then describes file system corruption (the storm). Finally, the set of deterministic corrective actions used by Fsck (the Coast Guard to the rescue) is presented.

1. Overview of the File System

The file system is discussed in detail in [Mckusick83]; this section gives a brief overview.

1.1. Superblock

A file system is described by its super-block. The super-block is built when the file system is created (see newfs(8)) and never changes. The super-block contains the basic parameters of the file system, such as the number of data blocks it contains and a count of the maximum number of files. Because the super-block contains critical data, newfs replicates it to protect against catastrophic loss. The default super block always resides at a fixed offset from the beginning of the file system's disk partition. The redundant super blocks are not referenced unless a head crash or other hard disk error causes the default super-block to be unusable. The redundant blocks are sprinkled throughout the disk partition.

Within the file system are files. Certain files are distinguished as directories and contain collections of pointers to files that may themselves be directories. Every file has a descriptor associated with it called an inode. The inode contains information describing ownership of the file, time stamps indicating modification and access times for the file, and an array of indices pointing to the data blocks for the file. In this section, we assume that the first 12 blocks of the file...

This document reflects the use of Fsck with the revised file system organization implemented in release 0.1 of the Sun UNIX operating system. This is a revision of the original paper written by T. J. Kowalski.

28 July 1983
are directly referenced by values stored in the inode structure itself. The inode structure may also contain references to indirect blocks containing further data block indices. In a file system with a 4096 byte block size, a singly indirect block contains 1024 further block addresses, a doubly indirect block contains 1024 addresses of further single indirect blocks, and a triply indirect block contains 1024 addresses of further doubly indirect blocks.

Sun's block size is 4K; fragment size is 1K.

In order to create files with up to $2^{32}$ bytes, using only two levels of indirection, the minimum size of a file system block is 4096 bytes. The size of file system blocks can be any power of two greater than or equal to 4096. The block size of the file system is maintained in the superblock, so it is possible for file systems of different block sizes to be accessible simultaneously on the same system. The block size must be decided when `newfs` creates the file system; the block size cannot be subsequently changed without rebuilding the file system.

1.2. Summary Information

Associated with the super block is non replicated summary information. The summary information changes as the file system is modified. The summary information contains the number of blocks, fragments, inodes and directories in the file system.

1.3. Cylinder Groups

The file system partitions the disk into one or more areas called cylinder groups. A cylinder group is comprised of one or more consecutive cylinders on a disk. Each cylinder group includes inode slots for files, a block map describing available blocks in the cylinder group, and summary information describing the usage of data blocks within the cylinder group. A fixed number of inodes is allocated for each cylinder group when the file system is created. The current policy is to allocate one inode for each 2048 bytes of disk space; this is expected to be far more inodes than will ever be needed.

All the cylinder group bookkeeping information could be placed at the beginning of each cylinder group. However if this approach were used, all the redundant information would be on the top platter. A single hardware failure that destroyed the top platter could cause the loss of all copies of the redundant super-blocks. Thus the cylinder group bookkeeping information begins at a floating offset from the beginning of the cylinder group. The offset for the $(i+1)$st cylinder group is about one track further from the beginning of the cylinder group than it was for the $i$th cylinder group. In this way, the redundant information spirals down into the pack; any single track, cylinder, or platter can be lost without losing all copies of the super-blocks. Except for the first cylinder group, the space between the beginning of the cylinder group and the beginning of the cylinder group information stores data.

1.4. Fragments

To avoid waste in storing small files, the file system space allocator divides a single file system block into one or more fragments. The fragmentation of the file system is specified when the file system is created; each file system block can be optionally broken into 2, 4, or 8 addressable fragments. The lower bound on the size of these fragments is constrained by the disk sector

†The actual number may vary from system to system, but is usually in the range 5-13.
size; typically 512 bytes is the lower bound on fragment size. The block map associated with each cylinder group records the space availability at the fragment level. Aligned fragments are examined to determine block availability.

On a file system with a block size of 4096 bytes and a fragment size of 1024 bytes, a file is represented by zero or more 4096 byte blocks of data, and possibly a single fragmented block. If a file system block must be fragmented to obtain space for a small amount of data, the remainder of the block is made available for allocation to other files. For example, consider an 11000 byte file stored on a 4096/1024 byte file system. This file uses two full size blocks and a 3072 byte fragment. If no fragments with at least 3072 bytes are available when the file is created, a full size block is split yielding the necessary 3072 byte fragment and an unused 1024 byte fragment. This remaining fragment can be allocated to another file, as needed.

1.5. Updates to the File System

Every working day hundreds of files are created, modified, and removed. Every time a file is modified, the operating system performs a series of file system updates. These updates, when written on disk, yield a consistent file system. The file system stages all modifications of critical information; modification can either be completed or cleanly backed out after a crash. Knowing the information that is first written to the file system, deterministic procedures can be developed to repair a corrupted file system. To understand this process, the order that the update requests were being honored must first be understood.

When a user program does an operation to change the file system, such as a write, the data to be written is copied into an internal in-core buffer in the kernel. Normally, the disk update is handled asynchronously; the user process is allowed to proceed even though the data has not yet been written to the disk. The data, along with the inode information reflecting the change, is eventually written out to disk. The real disk write may not happen until long after the write system call has returned. Thus at any given time, the file system, as it resides on the disk, lags behind the state of the file system represented by the in-core information.

The disk information is updated to reflect the in-core information when the buffer is required for another use, when a sync(2) is done (at 30 second intervals) by /etc/update(8), or by manual operator intervention with the sync(8) command. If the system is halted without writing out the in-core information, the file system on the disk will be in an inconsistent state.

If all updates are done asynchronously, several serious inconsistencies can arise. One inconsistency is that a block may be claimed by two inodes. Such an inconsistency can occur when the system is halted before the pointer to the block in the old inode has been cleared in the copy of the old inode on the disk, and after the pointer to the block in the new inode has been written out to the copy of the new inode on the disk. Here, there is no deterministic method for deciding which inode should really claim the block. A similar problem can arise with a multiply claimed inode.

The problem with asynchronous inode updates can be avoided by doing all inode deallocations synchronously. Consequently, inodes and indirect blocks are written to the disk synchronously (i.e. the process blocks until the information is really written to disk) when they are being deallocated. Similarly inodes are kept consistent by synchronously deleting, adding, or changing directory entries.
2. Fixing Corrupted File Systems

A file system can become corrupted in several ways. The most common of these ways are improper shutdown procedures and hardware failures.

File systems may become corrupted during an *unclean halt*. This happens when proper shutdown procedures are not observed, physically write-protecting a mounted file system, or a mounted file system is taken off-line. The most common operator procedural failure is forgetting to `sync` the system before halting the CPU.

File systems may become further corrupted if proper startup procedures are not observed, e.g., not checking a file system for inconsistencies, and not repairing inconsistencies. Allowing a corrupted file system to be used (and, thus, to be modified further) can be disastrous.

Any piece of hardware can fail at any time. Failures can be as subtle as a bad block on a disk pack, or as blatant as a non-functional disk-controller.

2.1. Detecting and Correcting Corruption

Normally *fsck* is run non-interactively. In this mode it will only fix corruptions that are expected to occur from an unclean halt. These actions are a proper subset of the actions that *fsck* will take when it is running interactively. Throughout this paper we assume that *fsck* is being run interactively, and all possible errors can be encountered. When an inconsistency is discovered in this mode, *fsck* reports the inconsistency for the operator to choose a corrective action.

A quiescent file system may be checked for structural integrity by performing consistency checks on the redundant data intrinsic to a file system. The redundant data is either read from the file system, or computed from other known values. The file system must be in a quiescent state when *fsck* is run, since *fsck* is a multi-pass program.

In the following sections, we discuss methods to discover inconsistencies and possible corrective actions for the cylinder group blocks, the inodes, the indirect blocks, and the data blocks containing directory entries.

2.2. Super-block Checking

The most commonly corrupted item in a file system is the summary information associated with the super-block. The summary information is prone to corruption because it is modified with every change to the file system's blocks or inodes, and is usually corrupted after an unclean halt.

The super-block is checked for inconsistencies involving file-system size, number of inodes, free-block count, and the free_inode count. The file-system size must be larger than the number of blocks used by the super-block and the number of blocks used by the list of inodes. The file-system size and layout information are the most critical pieces of information for *fsck*. While there is no way to actually check these sizes, since they are statically determined by `newfs`, *fsck* can check that these sizes are within reasonable bounds. All other file system checks require that these sizes be correct. If *fsck* detects corruption in the static parameters of the default super-block, *fsck* requests the operator to specify the location of an alternate super-block.

† I.e., unmounted and not being written on.
2.3. Free Block Checking

*Fsck* checks that all the blocks marked as free in the cylinder group block maps are not claimed by any files. When all the blocks have been initially accounted for, *fsck* checks that the number of free blocks plus the number of blocks claimed by the inodes equals the total number of blocks in the file system.

If anything is wrong with the block allocation maps, *fsck* will rebuild them, based on the list it has computed of allocated blocks.

The summary information associated with the super-block counts the total number of free blocks within the file system. *Fsck* compares this count to the number of free blocks it found within the file system. If the two counts do not agree, then *fsck* replaces the incorrect count in the summary information by the actual free-block count.

The summary information counts the total number of free inodes within the file system. *Fsck* compares this count to the number of free inodes it found within the file system. If the two counts do not agree, then *fsck* replaces the incorrect count in the summary information by the actual free-inode count.

2.4. Checking the Inode State

An individual inode is not as likely to be corrupted as the allocation information. However, because of the great number of active inodes, a few of the inodes are usually corrupted.

The list of inodes in the file system is checked sequentially starting with inode 2 (inode 0 marks unused inodes; inode 1 is saved for future generations) and progressing through the last inode in the file system. The state of each inode is checked for inconsistencies involving format and type, link count, duplicate blocks, bad blocks, and inode size.

Each inode contains a mode word. This mode word describes the type and state of the inode. Inodes must be one of six types: regular inode, directory inode, symbolic link inode, special block inode, special character inode, or socket inode. Inodes may be found in one of three allocation states: unallocated, allocated, and neither unallocated nor allocated. This last state suggests an incorrectly formatted inode. An inode can get in this state if bad data is written into the inode list. The only possible corrective action is for *fsck* to clear the inode.

2.5. Inode Links

Each inode counts the total number of directory entries linked to the inode. *Fsck* verifies the link count of each inode by starting at the root of the file system, and descending through the directory structure. The actual link count for each inode is calculated during the descent.

If the stored link count is non-zero and the actual link count is zero, then no directory entry appears for the inode. If this happens, *fsck* will place the disconnected file in the *lost+found* directory. If the stored and actual link counts are non-zero and unequal, a directory entry may have been added or removed without the inode being updated. If this happens, *fsck* replaces the incorrect stored link count by the actual link count.

Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by the inode. Since indirect blocks are owned by an inode, inconsistencies in indirect blocks directly affect the inode that owns it.

*Fsck* compares each block number claimed by an inode against a list of already allocated blocks. If another inode already claims a block number, then the block number is added to a list of
duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number.

If there are any duplicate blocks, fsck will perform a partial second pass over the inode list to find the inode of the duplicated block. The second pass is needed, since without examining the files associated with these inodes for correct content, not enough information is available to determine which inode is corrupted and should be cleared. If this condition does arise (only hardware failure will cause it), then the inode with the earliest modify time is usually incorrect, and should be cleared. If this happens, fsck prompts the operator to clear both inodes. The operator must decide which one should be kept and which one should be cleared.

Fsck checks the range of each block number claimed by an inode. If the block number is lower than the first data block in the file system, or greater than the last data block, then the block number is a bad block number. Many bad blocks in an inode are usually caused by an indirect block that was not written to the file system, a condition which can only occur if there has been a hardware failure. If an inode contains bad block numbers, fsck prompts the operator to clear it.

### 2.6. Inode Data Size

Each inode contains a count of the number of data blocks that it contains. The number of actual data blocks is the sum of the allocated data blocks and the indirect blocks. Fsck computes the actual number of data blocks and compares that block count against the actual number of blocks the inode claims. If an inode contains an incorrect count fsck prompts the operator to fix it.

Each inode contains a thirty-two bit size field. The size is the number of data bytes in the file associated with the inode. The consistency of the byte size field is roughly checked by computing from the size field the maximum number of blocks that should be associated with the inode, and comparing that expected block count against the actual number of blocks the inode claims.

### 2.7. Checking the Data Associated with an Inode

An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be the same kind. The three types of data blocks are: plain data blocks, symbolic link data blocks, and directory data blocks. Plain data blocks contain the information stored in a file; symbolic link data blocks contain the path name stored in a link. Directory data blocks contain directory entries. Fsck can only check the validity of directory data blocks.

Each directory data block is checked for several types of inconsistencies. These inconsistencies include directory inode numbers pointing to unallocated inodes, directory inode numbers that are greater than the number of inodes in the file system, incorrect directory inode numbers for "." and "..", and directories that are not attached to the file system. If the inode number in a directory data block references an unallocated inode, then fsck will remove that directory entry. Again, this condition can only arise when there has been a hardware failure.

If a directory entry inode number references outside the inode list, then fsck will remove that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for "." must be the first entry in the directory data block. The inode number for "." must reference itself; e.g., it must equal the inode number for the directory data block. The directory inode number entry for ".." must be the second entry in the directory data block. Its value must equal the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory).
the directory inode numbers are incorrect, *fsck* will replace them with the correct values.

### 2.8. File System Connectivity

*Fsck* checks the general connectivity of the file system. If directories are not linked into the file system, then *fsck* links the directory back into the file system in the *lost+found* directory. This condition only occurs when there has been a hardware failure.

---

**References**


Appendix A. Fsck Error Conditions

A.1. Conventions

*Fsck* is a multi-pass file system check program. Each file system pass invokes a different Phase of the *fsck* program. After the initial setup, *fsck* performs successive Phases over each file system, checking blocks and sizes, path-names, connectivity, reference counts, and the map of free blocks, (possibly rebuilding it), and performs some cleanup.

Normally *fsck* is run non-interactively to *preen* the file systems after an unclean halt. While preening a file system, it will only fix corruptions that are expected to occur from an unclean halt. These actions are a proper subset of the actions that *fsck* will take when it is running interactively. Throughout this appendix many errors have several options that the operator can take. When an inconsistency is detected, *fsck* reports the error condition to the operator. If a response is required, *fsck* prints a prompt message and waits for a response. When preening most errors are fatal. For those that are expected, the response taken is noted. This appendix explains the meaning of each error condition, the possible responses, and the related error conditions.

The error conditions are organized by the Phase of the *fsck* program in which they can occur. The error conditions that may occur in more than one Phase will be discussed in initialization.

A.2. Initialization

Before a file system check can be performed, certain tables have to be set up and certain files opened. This section concerns itself with the opening of files and the initialization of tables. This section lists error conditions resulting from command line options, memory requests, opening of files, status of files, file system size checks, and creation of the scratch file. All of the initialization errors are fatal when the file system is being preen’ed.

*C option?*

*C* is not a legal option to *fsck*; legal options are -b, -y, -n, and -p. *Fsck* terminates on this error condition. See the *fsck*(8) manual entry for further detail.

```
cannot alloc NNN bytes for blockmap
cannot alloc NNN bytes for freemap
cannot alloc NNN bytes for statemap
cannot alloc NNN bytes for lncmp
Fsck's request for memory for its virtual memory tables failed. This should never happen. Fsck terminates on this error condition. See a guru.
```

*Can't open checklist file: F*

The file system checklist file *F* (usually */etc/fstab*) can not be opened for reading. *Fsck* terminates on this error condition. Check access modes of *F*.

*Can't stat root*

*Fsck*’s request for statistics about the root directory “/” failed. This should never happen. *Fsck* terminates on this error condition. See a guru.
Can't stat $F$
Can't make sense out of name $F$

$Fsck$'s request for statistics about the file system $F$ failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of $F$.

Can't open $F$

$Fsck$'s request attempt to open the file system $F$ failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of $F$.

$F$: (NO WRITE)

Either the -n flag was specified or $fsck$'s attempt to open the file system $F$ for writing failed. When running manually, all the diagnostics are printed out, but no modifications are attempted to fix them.

file is not a block or character device; OK

You have given $fsck$ a regular file name by mistake. Check the type of the file specified.

Possible responses to the OK prompt are:

YES
  Ignore this error condition.

NO ignore this file system and continues checking the next file system given.

One of the following messages will appear:

MAGIC NUMBER WRONG
NCG OUT OF RANGE
CPG OUT OF RANGE
NSECT < 1
NTRAK < 1
SPC DOES NOT JIVE w/NTRAK*NSECT
INODES NOT MULTIPLE OF A BLOCK
IMPLIES MORE INODE THAN DATA BLOCKS
NCYL DOES NOT JIVE WITH NCN*CNP
FPG DOES NOT JIVE WITH CPG & SPC
SIZE PREPOSTEROUSLY SMALL
SIZE PREPOSTEROUSLY LARGE
CGSIZE INCORRECT
CSSIZE INCORRECT

and will be followed by the message:

$F$: BAD SUPER BLOCK: B
USE -b OPTION TO FSCK TO SPECIFY LOCATION OF AN ALTERNATE SUPER-BLOCK TO SUPPLY NEEDED INFORMATION; SEE fsck(8).

The super block has been corrupted. An alternative super block must be selected from among those listed by newfs(8) when the file system was created. For file systems with a blocksize less than 32K, specifying -b32 is a good first choice.

CAN NOT SEEK: BLK B (CONTINUE)

$Fsck$'s request for moving to a specified block number $B$ in the file system failed. This should never happen. See a guru.

28 July 1983
Fsck Error Conditions

Possible responses to the CONTINUE prompt are:

YES
- attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If the block was part of the virtual memory buffer cache, *fsck* will terminate with the message "Fatal I/O error".

NO terminate the program.

**CAN NOT READ: BLK B (CONTINUE)**

*Fsck*’s request for reading a specified block number B in the file system failed. This should never happen. See a guru.

Possible responses to the CONTINUE prompt are:

YES
- attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If the block was part of the virtual memory buffer cache, *fsck* will terminate with the message "Fatal I/O error".

NO terminate the program.

**CAN NOT WRITE: BLK B (CONTINUE)**

*Fsck*’s request for writing a specified block number B in the file system failed. The disk is write-protected. See a guru.

Possible responses to the CONTINUE prompt are:

YES
- attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If the block was part of the virtual memory buffer cache, *fsck* will terminate with the message "Fatal I/O error".

NO terminate the program.

A.3. Phase 1 – Check Blocks and Sizes

This phase concerns itself with the inode list. This section lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format. All errors in this phase except INCORRECT BLOCK COUNT are fatal if the file system is being preen’ed,

cg C: bad magic number The magic number of cylinder group C is wrong. This usually indicates that the cylinder group maps have been destroyed. When running manually the cylinder group is marked as needing to be reconstructed.

UNKNOWN FILE TYPE I==I (CLEAR) The mode word of the inode I indicates that the inode is not a special block inode, special character inode, socket inode, regular inode, symbolic link, or directory inode.

Possible responses to the CLEAR prompt are:
YES

de-allocate inode \( I \) by zeroing its contents. This will always invoke the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode.

NO ignore this error condition.

**LINK COUNT TABLE OVERFLOW (CONTINUE)**

An internal table for *fsck* containing allocated inodes with a link count of zero has no more room. Recompile *fsck* with a larger value of MAXLNCNT.

Possible responses to the CONTINUE prompt are:

YES

continue with the program. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If another allocated inode with a zero link count is found, this error condition is repeated.

NO terminate the program.

**B BAD I==I**

Inode \( I \) contains block number \( B \) with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the **EXCESSIVE BAD BLKS** error condition in Phase 1 if inode \( I \) has too many block numbers outside the file system range. This error condition will always invoke the **BAD/DUP** error condition in Phase 2 and Phase 4.

**EXCESSIVE BAD BLKS I==I (CONTINUE)**

There is more than a tolerable number (usually 10) of blocks with a number lower than the number of the first data block in the file system or greater than the number of last block in the file system associated with inode \( I \).

Possible responses to the CONTINUE prompt are:

YES

ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system.

NO terminate the program.

**B DUP I==I**

Inode \( I \) contains block number \( B \) which is already claimed by another inode. This error condition may invoke the **EXCESSIVE DUP BLKS** error condition in Phase 1 if inode \( I \) has too many block numbers claimed by other inodes. This error condition will always invoke Phase 1b and the **BAD/DUP** error condition in Phase 2 and Phase 4.

**EXCESSIVE DUP BLKS I==I (CONTINUE)**

There is more than a tolerable number (usually 10) of blocks claimed by other inodes.

Possible responses to the CONTINUE prompt are:

YES

ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system.
NO terminate the program.

**DUP TABLE OVERFLOW (CONTINUE)**
An internal table in `fsck` containing duplicate block numbers has no more room. Recompile `fsck` with a larger value of DUPTBLSIZE.
Possible responses to the CONTINUE prompt are:
YES
continue with the program. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to re-check this file system. If another duplicate block is found, this error condition will repeat.
NO terminate the program.

**PARTIALLY ALLOCATED INODE I==I (CLEAR)**
Inode I is neither allocated nor unallocated.
Possible responses to the CLEAR prompt are:
YES
de-allocate inode I by zeroing its contents.
NO ignore this error condition.

**INCORRECT BLOCK COUNT I==I (X should be Y) (CORRECT)**
The block count for inode I is X blocks, but should be Y blocks. When preen'ing the count is corrected.
Possible responses to the CORRECT prompt are:
YES
replace the block count of inode I with Y.
NO ignore this error condition.

### A.4. Phase 1B: Rescan for More Dup's

When a duplicate block is found in the file system, the file system is rescanned to find the inode which previously claimed that block. This section lists the error condition when the duplicate block is found.

**B DUP I==I**
Inode I contains block number B that is already claimed by another inode. This error condition will always invoke the BAD/DUP error condition in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the DUP error condition in Phase 1.

### A.5. Phase 2 – Check Pathnames

This phase concerns itself with removing directory entries pointing to error conditioned inodes from Phase 1 and Phase 1b. This section lists error conditions resulting from root inode mode and status, directory inode pointers in range, and directory entries pointing to bad inodes. All errors in this phase are fatal if the file system is being preen'ed.
ROOT INODE UNALLOCATED. TERMINATING.
The root inode (usually inode number 2) has no allocate mode bits. This should never happen. The program will terminate.

NAME TOO LONG F
An excessively long path name has been found. This is usually indicative of loops in the file system name space. This can occur if the super user has made circular links to directories. The offending links must be removed (by a guru).

ROOT INODE NOT DIRECTORY (FIX)
The root inode (usually inode number 2) is not directory inode type.
Possible responses to the FIX prompt are:
YES
    replace the root inode's type to be a directory. If the root inode's data blocks are not directory blocks, a VERY large number of error conditions will be produced.
NO terminate the program.

DUPS/BAD IN ROOT INODE (CONTINUE)
Phase 1 or Phase 1b have found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system.
Possible responses to the CONTINUE prompt are:
YES
    ignore the DUPS/BAD error condition in the root inode and attempt to continue to run the file system check. If the root inode is not correct, then this may result in a large number of other error conditions.
NO terminate the program.

I OUT OF RANGE I=I NAME=F (REMOVE)
A directory entry F has an inode number I which is greater than the end of the inode list.
Possible responses to the REMOVE prompt are:
YES
    the directory entry F is removed.
NO ignore this error condition.

UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE)
A directory entry F has a directory inode I without allocate mode bits. The owner O, mode M, size S, modify time T, and directory name F are printed.
Possible responses to the REMOVE prompt are:
YES
    the directory entry F is removed.
NO ignore this error condition.

28 July 1983
A directory entry $F$ has an inode $I$ without allocate mode bits. The owner $O$, mode $M$, size $S$, modify time $T$, and file name $F$ are printed.

Possible responses to the REMOVE prompt are:

YES
the directory entry $F$ is removed.

NO ignore this error condition.

DUP/BAD $I==I$ OWNER==0 MODE==M SIZE==S MTIME==T DIR==F (REMOVE)
Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory entry $F$, directory inode $I$. The owner $O$, mode $M$, size $S$, modify time $T$, and directory name $F$ are printed.

Possible responses to the REMOVE prompt are:

YES
the directory entry $F$ is removed.

NO ignore this error condition.

DUP/BAD $I==I$ OWNER==0 MODE==M SIZE==S MTIME==T FILE==F (REMOVE)
Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory entry $F$, inode $I$. The owner $O$, mode $M$, size $S$, modify time $T$, and file name $F$ are printed.

Possible responses to the REMOVE prompt are:

YES
the directory entry $F$ is removed.

NO ignore this error condition.

**A.6. Phase 3 – Check Connectivity**

This phase concerns itself with the directory connectivity seen in Phase 2. This section lists error conditions resulting from unreferenced directories, and missing or full *lost+found* directories.

**DIRECTORY D CORRUPTED (SALVAGE)**

A directory with an inconsistent internal state has been found. This error is fatal if the file system is being preen’ed.

Possible responses to the SALVAGE prompt are:

YES
throw away all entries up to the next 512-byte boundary. This rather drastic action can throw away up to 42 entries, and should be taken only after other recovery efforts have failed.

NO Skip up to the next 512-byte boundary and resume reading, but do not modify the directory.

**UNREF DIR I==I OWNER==0 MODE==M SIZE==S MTIME==T (RECONNECT)**
The directory inode $I$ was not connected to a directory entry when the file system was traversed. The owner $O$, mode $M$, size $S$, and modify time $T$ of directory inode $I$ are printed. When preen’ing, the directory is reconnected if its size is non-zero, otherwise it is cleared.
Possible responses to the RECONNECT prompt are:

YES
  reconnect directory inode I to the file system in the directory for lost files (usually lost+found). This may invoke the lost+found error condition in Phase 3 if there are problems connecting directory inode I to lost+found. This may also invoke the CONNECTED error condition in Phase 3 if the link was successful.

NO ignore this error condition. This will always invoke the UNREF error condition in Phase 4.

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 will always invoke the UNREF error condition in Phase 4. Check access modes of lost+found. See fsck(8) manual entry for further detail. This error is fatal if the file system is being preen'ed.

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 will always invoke the UNREF error condition in Phase 4. Clean out unnecessary entries in lost+found or make lost+found larger. See fsck(8) manual entry for further detail. This error is fatal if the file system is being preen'ed.

DIR I=I1 CONNECTED. PARENT WAS I=I2
This is an advisory message indicating a directory inode I1 was successfully connected to the lost+found directory. The parent inode I2 of the directory inode I1 is replaced by the inode number of the lost+found directory.

A.7. Phase 4 - Check Reference Counts

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This section lists error conditions resulting from unreferenced files, missing or full lost+found directory, incorrect link counts for files, directories, symbolic links, or special files, unreferenced files, symbolic links, and directories, bad and duplicate blocks in files, symbolic links, and directories, and incorrect total free-inode counts. All errors in this phase are correctable if the file system is being preen'ed except running out of space in the lost+found directory.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)
Inode I was not connected to a directory entry when the file system was traversed. The owner O, mode M, size S, and modify time T of inode I are printed. When preen'ing the file is cleared if either its size or its link count is zero, otherwise it is reconnected.

Possible responses to the RECONNECT prompt are:

YES
  reconnect inode I to the file system in the directory for lost files (usually lost+found). This may invoke the lost+found error condition in Phase 4 if there are problems connecting inode I to lost+found.

NO ignore this error condition. This will always invoke the CLEAR error condition in Phase 4.
(CLEAR)
The inode mentioned in the immediately previous error condition cannot be reconnected. This cannot occur if the file system is being preen'ed, since lack of space to reconnect files is a fatal error.

Possible responses to the CLEAR prompt are:

YES
de-allocate the inode mentioned in the immediately previous error condition by zeroing its contents.

NO ignore this error condition.

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 will always invoke the CLEAR error condition in Phase 4. Check access modes of lost+found. This error is fatal if the file system is being preen'ed.

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 will always invoke the CLEAR error condition in Phase 4. Check size and contents of lost+found. This error is fatal if the file system is being preen'ed.

LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)
The link count for inode I which is a file, is X but should be Y. The owner O, mode M, size S, and modify time T are printed. When preen'ing the link count is adjusted.

Possible responses to the ADJUST prompt are:

YES
replace the link count of file inode I with Y.

NO ignore this error condition.

LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)
The link count for inode I which is a directory, is X but should be Y. The owner O, mode M, size S, and modify time T of directory inode I are printed. When preen'ing the link count is adjusted.

Possible responses to the ADJUST prompt are:

YES
replace the link count of directory inode I with Y.

NO ignore this error condition.

LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)
The link count for F inode I is X but should be Y. The name F, owner O, mode M, size S, and modify time T are printed. When preen'ing the link count is adjusted.
Possible responses to the ADJUST prompt are:

YES
   replace the link count of inode I with Y.

NO ignore this error condition.

**UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)**
Inode 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 inode I are printed. When preen'ing, this is a file that was not connected because its size or link count was zero, hence it is cleared.

Possible responses to the CLEAR prompt are:

YES
   de-allocate inode I by zeroing its contents.

NO ignore this error condition.

**UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)**
Inode 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 inode I are printed. When preen'ing, this is a directory that was not connected because its size or link count was zero, hence it is cleared.

Possible responses to the CLEAR prompt are:

YES
   de-allocate inode I by zeroing its contents.

NO ignore this error condition.

**BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)**
Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with file inode I. The owner O, mode M, size S, and modify time T of inode I are printed. This error cannot arise when the file system is being preen'ed, as it would have caused a fatal error earlier.

Possible responses to the CLEAR prompt are:

YES
   de-allocate inode I by zeroing its contents.

NO ignore this error condition.

**BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)**
Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory inode I. The owner O, mode M, size S, and modify time T of inode I are printed. This error cannot arise when the file system is being preen'ed, as it would have caused a fatal error earlier.

Possible responses to the CLEAR prompt are:

YES
   de-allocate inode I by zeroing its contents.

NO ignore this error condition.

**FREE INODE COUNT WRONG IN SUPERBLK (FIX)**
The actual count of the free inodes does not match the count in the super-block of the file system.
Fsck Error Conditions

system. When preen’ing, the count is fixed.
Possible responses to the FIX prompt are:
YES
    replace the count in the super-block by the actual count.
NO ignore this error condition.

A.8. Phase 5 - Check Cyl Groups

This phase concerns itself with the free-block maps. This section lists error conditions resulting
from allocated blocks in the free-block maps, free blocks missing from free-block maps, and the
total free-block count incorrect.

cg C: bad magic number
The magic number of cylinder group C is wrong. This usually indicates that the cylinder group
maps have been destroyed. When running manually the cylinder group is marked as needing to
be reconstructed. This error is fatal if the file system is being preen’ed.

EXCESSIVE BAD BLKS IN BIT MAPS (CONTINUE)
An inode contains more than a tolerable number (usually 10) of blocks claimed by other inodes
or that are out of the legal range for the file system. This error is fatal if the file system is
being preen’ed.
Possible responses to the CONTINUE prompt are:
YES
    ignore the rest of the free-block maps and continue the execution of fsck.
NO terminate the program.

SUMMARY INFORMATION T BAD
where T is one or more of:
(INODE FREE)
(BLOCK OFFSETS)
(FRAG SUMMARIES)
(SUPER BLOCK SUMMARIES)
The indicated summary information was found to be incorrect. This error condition will always
invoke the BAD CYLINDER GROUPS condition in Phase 6. When preen’ing, the summary
information is recomputed.

X BLK(S) MISSING
X blocks unused by the file system were not found in the free-block maps. This error condition
will always invoke the BAD CYLINDER GROUPS condition in Phase 6. When preen’ing,
the block maps are rebuilt.

BAD CYLINDER GROUPS (SALVAGE)
Phase 5 has found bad blocks in the free-block maps, duplicate blocks in the free-block maps, or
blocks missing from the file system. When preen’ing, the cylinder groups are reconstructed.
Possible responses to the SALVAGE prompt are:

18 28 July 1983
YES
   replace the actual free-block maps with a new free-block maps.
NO ignore this error condition.

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. When preen’ing, the counts are fixed.
Possible responses to the FIX prompt are:
YES
   replace the count in the super-block by the actual count.
NO ignore this error condition.

A.9. Phase 8 - Salvage Cylinder Groups
This phase concerns itself with the free-block maps reconstruction. No error messages are produced.

A.10. Cleanup
Once a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system and modify status of the file system.

V files, W used, X free (Y frags, Z blocks)
This is an advisory message indicating that the file system checked contained V files using W fragment sized blocks leaving X fragment sized blocks free in the file system. The numbers in parenthesis breaks the free count down into Y free fragments and Z free full sized blocks.

***** REBOOT UNIX *****
This is an advisory message indicating that the root file system has been modified by fsck. If UNIX is not rebooted immediately, the work done by fsck may be undone by the in-core copies of tables UNIX keeps. When preen’ing, fsck will exit with a code of 4. The auto-reboot script interprets an exit code of 4 by issuing a reboot system call.

***** FILE SYSTEM WAS MODIFIED *****
This is an advisory message indicating that the current file system was modified by fsck. If this file system is mounted or is the current root file system, fsck should be halted and UNIX rebooted. If UNIX is not rebooted immediately, the work done by fsck may be undone by the in-core copies of tables UNIX keeps.
SENDMAIL – An Internetwork Mail Router

Eric Allman†
Britton-Lee, Inc.
1919 Addison Street, Suite 105.
Berkeley, California 94704.

ABSTRACT
Routing mail through a heterogenous internet presents many new problems. Among the worst of these is that of address mapping. Historically, this has been handled on an ad hoc basis. However, this approach has become unmanageable as internets grow.

Sendmail acts as a unified “post office” to which all mail can be submitted. Address interpretation is controlled by a production system, which can parse both domain-based addressing and old-style ad hoc addresses. The production system is powerful enough to rewrite addresses in the message header to conform to the standards of a number of common target networks, including old (NCP/RFC733) Arpanet, new (TCP/RFC822) Arpanet, UUCP, and Phonenet. Sendmail also implements an SMTP server, message queueing, and aliasing.

Sendmail implements a general internetwork mail routing facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration.

In a simple network, each node has an address, and resources can be identified with a host-resource pair; in particular, the mail system can refer to users using a host-username pair. Host names and numbers have to be administered by a central authority, but usernames can be assigned locally to each host.

In an internetwork, multiple networks with different characteristics and managements must communicate. In particular, the syntax and semantics of resource identification change. Certain special cases can be handled trivially by ad hoc techniques, such as providing network names that appear local to hosts on other networks, as with the Ethernet at Xerox PARC. However, the general case is extremely complex. For example, some networks require point-to-point routing, which simplifies the database update problem since only adjacent hosts must be entered into the system tables, while others use end-to-end addressing. Some networks use a left-associative syntax and others use a right-associative syntax, causing ambiguity in mixed addresses.

Internet standards seek to eliminate these problems. Initially, these proposed expanding the address pairs to address triples, consisting of {network, host, resource} triples. Network numbers must be universally agreed upon, and hosts can be assigned locally on each network. The user-level presentation was quickly expanded to address domains, comprised of a local resource identification and a hierarchical domain specification with a common static root. The domain technique separates the issue of physical versus logical addressing. For example, an address of the form “eric@ac.berkeley.edu” describes only the logical organization of the address space.

Sendmail is intended to help bridge the gap between the totally ad hoc world of networks that know nothing of each other and the clean, tightly-coupled world of unique network numbers. It can accept old arbitrary address syntaxes, resolving ambiguities using heuristics specified by the

†A considerable part of this work was done while under the employ of the INGRES Project at the University of California at Berkeley.
system administrator, as well as domain-based addressing. It helps guide the conversion of message formats between disparate networks. In short, sendmail is designed to assist a graceful transition to consistent internetwork addressing schemes.

Section 1 discusses the design goals for sendmail. Section 2 gives an overview of the basic functions of the system. In section 3, details of usage are discussed. Section 4 compares sendmail to other internet mail routers, and an evaluation of sendmail is given in section 5, including future plans.

1. DESIGN GOALS

Design goals for sendmail include:

1. Compatibility with the existing mail programs, including Bell version 6 mail, Bell version 7 mail [UNIX83], Berkeley Mail [Shoens79], BerkNet mail [Schmidt79], and hopefully UUCP mail [Nowitz78a, Nowitz78b]. ARPANET mail [Crocker77a, Postel77] was also required.

2. Reliability, in the sense of guaranteeing that every message is correctly delivered or at least brought to the attention of a human for correct disposal; no message should ever be completely lost. This goal was considered essential because of the emphasis on mail in our environment. It has turned out to be one of the hardest goals to satisfy, especially in the face of the many anomalous message formats produced by various ARPANET sites. For example, certain sites generate improperly formatted addresses, occasionally causing error-message loops. Some hosts use blanks in names, causing problems with UNIX mail programs that assume that an address is one word. The semantics of some fields are interpreted slightly differently by different sites. In summary, the obscure features of the ARPANET mail protocol really are used and are difficult to support, but must be supported.

3. Existing software to do actual delivery should be used whenever possible. This goal derives as much from political and practical considerations as technical.

4. Easy expansion to fairly complex environments, including multiple connections to a single network type (such as with multiple UUCP or Ether nets [Metcalfe76]). This goal requires consideration of the contents of an address as well as its syntax in order to determine which gateway to use. For example, the ARPANET is bringing up the TCP protocol to replace the old NCP protocol. No host at Berkeley runs both TCP and NCP, so it is necessary to look at the ARPANET host name to determine whether to route mail to an NCP gateway or a TCP gateway.

5. Configuration should not be compiled into the code. A single compiled program should be able to run as is at any site (barring such basic changes as the CPU type or the operating system). We have found this seemingly unimportant goal to be critical in real life. Besides the simple problems that occur when any program gets recompiled in a different environment, many sites like to "fiddle" with anything that they will be recompiling anyway.

6. Sendmail must be able to let various groups maintain their own mailing lists, and let individuals specify their own forwarding, without modifying the system alias file.

7. Each user should be able to specify which mailer to execute to process mail being delivered for him. This feature allows users who are using specialized mailers that use a different format to build their environment without changing the system, and facilitates specialized functions (such as returning an "I am on vacation" message).

8. Network traffic should be minimized by batching addresses to a single host where possible, without assistance from the user.

These goals motivated the architecture illustrated in Figure 1. The user interacts with a mail generating and sending program. When the mail is created, the generator calls sendmail,
which routes the message to the correct mailer(s). Since some of the senders may be network servers and some of the mailers may be network clients, sendmail may be used as an internet mail gateway.

2. OVERVIEW

2.1. System Organization

Sendmail neither interfaces with the user nor does actual mail delivery. Rather, it collects a message generated by a user interface program (UIP) such as Berkeley Mail, MS [Crocker77b], or MH [Borden79], edits the message as required by the destination network, and calls appropriate mailers to do mail delivery or queueing for network transmission. This discipline allows the insertion of new mailers at minimum cost. In this sense sendmail resembles the Message Processing Module (MPM) of [Postel79b].

2.2. Interfaces to the Outside World

There are three ways sendmail can communicate with the outside world, both in receiving and in sending mail. These are using the conventional UNIX argument vector/return status, speaking SMTP over a pair of UNIX pipes, and speaking SMTP over an interprocess(or) channel.

2.2.1. Argument vector/return status

This technique is the standard UNIX method for communicating with the process. A list of recipients is sent in the argument vector, and the message body is sent on the standard input. Anything that the mailer prints is simply collected and sent back to the sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

\[\text{except when mailing to a file, when sendmail does the delivery directly.}\]
2.2.2. SMTP over pipes

The SMTP protocol [Postel82] can be used to run an interactive lock-step interface with the mailer. A subprocess is still created, but no recipient addresses are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the processes standard input. Anything appearing on the standard output must be a reply code in a special format.

2.2.3. SMTP over an IPC connection

This technique is similar to the previous technique, except that it uses a 4.2BSD IPC channel [UNIX83]. This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a sendmail process on another machine.

2.3. Operational Description

When a sender wants to send a message, it issues a request to sendmail using one of the three methods described above. Sendmail operates in two distinct phases. In the first phase, it collects and stores the message. In the second phase, message delivery occurs. If there were errors during processing during the second phase, sendmail creates and returns a new message describing the error and/or returns a status code telling what went wrong.

2.3.1. Argument processing and address parsing

If sendmail is called using one of the two subprocess techniques, the arguments are first scanned and option specifications are processed. Recipient addresses are then collected, either from the command line or from the SMTP RCPT command, and a list of recipients is created. Aliases are expanded at this step, including mailing lists. As much validation as possible of the addresses is done at this step: syntax is checked, and local addresses are verified, but detailed checking of host names and addresses is deferred until delivery. Forwarding is also performed as the local addresses are verified.

Sendmail appends each address to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list, and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates, preventing alias loops and duplicate messages delivered to the same recipient, as might occur if a person is in two groups.

2.3.2. Message collection

Sendmail then collects the message. The message should have a header at the beginning. No formatting requirements are imposed on the message except that they must be lines of text (in other words, binary data is not allowed). The header is parsed and stored in memory, and the body of the message is saved in a temporary file.

To simplify the program interface, the message is collected even if no addresses were valid. The message will be returned with an error.

2.3.3. Message delivery

For each unique mailer and host in the recipient list, sendmail calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that only accept one recipient at a time are handled properly.

The message is sent to the mailer using one of the same three interfaces used to submit a message to sendmail. Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and a suitable error message given as appropriate. The exit code must conform to a system standard or a generic message ("Service unavailable") is given.
2.3.4. Queueing for retransmission

If the mailer returned an status that indicated that it might be able to handle the mail later, sendmail will queue the mail and try again later.

2.3.5. Return to sender

If errors occur during processing, sendmail returns the message to the sender for retransmission. The letter can be mailed back or written in the file “dead.letter” in the sender’s home directory\(^2\).

2.4. Message Header Editing

Certain editing of the message header occurs automatically. Header lines can be inserted under control of the configuration file. Some lines can be merged; for example, a “From:” line and a “Full-name:” line can be merged under certain circumstances.

2.5. Configuration File

Almost all configuration information is read at runtime from an ASCII file, encoding macro definitions (defining the value of macros used internally), header declarations (telling sendmail the format of header lines that it will process specially, i.e., lines that it will add or reformat), mailer definitions (giving information such as the location and characteristics of each mailer), and address rewriting rules (a limited production system to rewrite addresses which is used to parse and rewrite the addresses).

To improve performance when reading the configuration file, a memory image can be provided. This provides a “compiled” form of the configuration file.

3. USAGE AND IMPLEMENTATION

3.1. Arguments

Arguments may be flags and addresses. Flags set various processing options. Following flag arguments, address arguments may be given, unless we are running in SMTP mode. Addresses follow the syntax in RFC822 [Crocker82] for ARPANET address formats. In brief, the format is:

1. Anything in parentheses is thrown away (as a comment).
2. Anything in angle brackets ("< >") is preferred over anything else. This rule implements the ARPANET standard that addresses of the form

   user name <machine-address>

   will send to the electronic “machine-address” rather than the human “user name.”
3. Double quotes (" ) quote phrases; backslashes quote characters. Backslashes are more powerful in that they will cause otherwise equivalent phrases to compare differently – for example, user and "user" are equivalent, but \user is different from either of them.

Parenses, angle brackets, and double quotes must be properly balanced and nested. The rewriting rules control remaining parsing\(^5\).

\(^2\)Obviously, if the site giving the error is not the originating site, the only reasonable option is to mail back to the sender. Also, there are many more error disposition options, but they only effect the error message – the “return to sender” function is always handled in one of these two ways.

\(^5\)Disclaimer: Some special processing is done after rewriting local names; see below.
3.2. Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to maintain a public repository of systems messages (such as the Berkeley msg program, or the MARS system [Sattley78]).

Any address passing through the initial parsing algorithm as a local address (i.e., not appearing to be a valid address for another mailer) is scanned for two special cases. If prefixed by a vertical bar ("|") the rest of the address is processed as a shell command. If the user name begins with a slash mark ("/") the name is used as a file name, instead of a login name.

Files that have setuid or setgid bits set but no execute bits set have those bits honored if sendmail is running as root.

3.3. Aliasing, Forwarding, Inclusion

Sendmail reroutes mail three ways. Aliasing applies system wide. Forwarding allows each user to reroute incoming mail destined for that account. Inclusion directs sendmail to read a file for a list of addresses, and is normally used in conjunction with aliasing.

3.3.1. Aliasing

Aliasing maps names to address lists using a system-wide file. This file is indexed to speed access. Only names that parse as local are allowed as aliases; this guarantees a unique key (since there are no nicknames for the local host).

3.3.2. Forwarding

After aliasing, recipients that are local and valid are checked for the existence of a "forward" file in their home directory. If it exists, the message is not sent to that user, but rather to the list of users in that file. Often this list will contain only one address, and the feature will be used for network mail forwarding.

Forwarding also permits a user to specify a private incoming mailer. For example, forwarding to:

"|/usr/local/newmail myname"
will use a different incoming mailer.

3.3.3. Inclusion

Inclusion is specified in RFC 733 [Crocker77a] syntax:

:Include: pathname

An address of this form reads the file specified by pathname and sends to all users listed in that file.

The intent is not to support direct use of this feature, but rather to use this as a subset of aliasing. For example, an alias of the form:

project: :include:/usr/project/userlist

is a method of letting a project maintain a mailing list without interaction with the system administration, even if the alias file is protected.

It is not necessary to rebuild the index on the alias database when a :include: list is changed.

3.4. Message Collection

Once all recipient addresses are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body, separated by a blank
The header is formatted as a series of lines of the form

field-name: field-value

Field-value can be split across lines by starting the following lines with a space or a tab. Some header fields have special internal meaning, and have appropriate special processing. Other headers are simply passed through. Some header fields may be added automatically, such as time stamps.

The body is a series of text lines. It is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

3.5. Message Delivery

The send queue is ordered by receiving host before transmission to implement message batching. Each address is marked as it is sent so rescanning the list is safe. An argument list is built as the scan proceeds. Mail to files is detected during the scan of the send list. The interface to the mailer is performed using one of the techniques described in section 2.2.

After a connection is established, sendmail makes the per-mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery completes.

3.6. Queued Messages

If the mailer returns a "temporary failure" exit status, the message is queued. A control file is used to describe the recipients to be sent to and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other salient parameter of the message. The header of the message is stored in the control file, so that the associated data file in the queue is just the temporary file that was originally collected.

3.7. Configuration

Configuration is controlled primarily by a configuration file read at startup. Sendmail should not need to be recompiled except

1. To change operating systems (V6, V7/32V, 4BSD).
2. To remove or insert the DBM (UNIX database) library.
3. To change ARPANET reply codes.
4. To add headers fields requiring special processing.

Adding mailers or changing parsing (i.e., rewriting) or routing information does not require recompilation.

If the mail is being sent by a local user, and the file ".mailcf" exists in the sender's home directory, that file is read as a configuration file after the system configuration file. The primary use of this feature is to add header lines.

The configuration file encodes macro definitions, header definitions, mailer definitions, rewriting rules, and options.

3.7.1. Macros

Macros can be used in three ways. Certain macros transmit unstructured textual information into the mail system, such as the name sendmail will use to identify itself in error messages. Other macros transmit information from sendmail to the configuration file for use in creating other fields (such as argument vectors to mailers): the name of the sender, and the host and user of the recipient. Other macros are
unused internally, and can be used as shorthand in the configuration file.

3.7.2. Header declarations

Header declarations inform sendmail of the format of known header lines. Knowledge of a few header lines is built into sendmail, such as the "From:" and "Date:" lines.

Most configured headers will be automatically inserted in the outgoing message if they don't exist in the incoming message. Certain headers are suppressed by some mailers.

3.7.3. Mailer declarations

Mailer declarations tell sendmail of the various mailers available to it. The definition specifies the internal name of the mailer, the pathname of the program to call, some flags associated with the mailer, and an argument vector to be used on the call; this vector is macro-expanded before use.

3.7.4. Address rewriting rules

The heart of address parsing in sendmail is a set of rewriting rules. These are an ordered list of pattern-replacement rules, (somewhat like a production system, except that order is critical), which are applied to each address. The address is rewritten textually until it is either rewritten into a special canonical form (i.e., a (mailer, host, user) 3-tuple, such as {arpanet, usc-isif, postel} representing the address "postel@usc-isif"), or it falls off the end. When a pattern matches, the rule is reapplied until it fails.

The configuration file also supports the editing of addresses into different formats. For example, an address of the form:

```
ucsfcl!tef
```

might be mapped into:

```
tef@ucsfcl.UUCP
```

to conform to the domain syntax. Translations can also be done in the other direction.

3.7.5. Option setting

There are several options that can be set from the configuration file. These include the pathnames of various support files, timeouts, default modes, etc.

4. COMPARISON WITH OTHER MAILERS

4.1. Delivermail

Sendmail is an outgrowth of delivermail. The primary differences are:

1. Configuration information is not compiled in. This change simplifies many of the problems of moving to other machines. It also allows easy debugging of new mailers.

2. Address parsing is more flexible. For example, delivermail only supported one gateway to any network, whereas sendmail can be sensitive to host names and reroute to different gateways.

3. Forwarding and :include: features eliminate the requirement that the system alias file be writable by any user (or that an update program be written, or that the system administration make all changes).

4. Sendmail supports message batching across networks when a message is being sent to multiple recipients.
A mail queue is provided in `sendmail`. Mail that cannot be delivered immediately but can potentially be delivered later is stored in this queue for a later retry. The queue also provides a buffer against system crashes; after the message has been collected it may be reliably redelivered even if the system crashes during the initial delivery.

`Sendmail` uses the networking support provided by 4.2BSD to provide a direct interface to networks such as the ARPANET and/or Ethernet using SMTP (the Simple Mail Transfer Protocol) over a TCP/IP connection.

**4.2. MMDF**

MMDF [Crocker79] spans a wider problem set than `sendmail`. For example, the domain of MMDF includes a “phone network” mailer, whereas `sendmail` calls on preexisting mailers in most cases.

MMDF and `sendmail` both support aliasing, customized mailers, message batching, automatic forwarding to gateways, queueing, and retransmission. MMDF supports two-stage timeout, which `sendmail` does not support.

The configuration for MMDF is compiled into the code.

Since MMDF does not consider backwards compatibility as a design goal, the address parsing is simpler but much less flexible.

It is somewhat harder to integrate a new channel into MMDF. In particular, MMDF must know the location and format of host tables for all channels, and the channel must speak a special protocol. This allows MMDF to do additional verification (such as verifying host names) at submission time.

MMDF strictly separates the submission and delivery phases. Although `sendmail` has the concept of each of these stages, they are integrated into one program, whereas in MMDF they are split into two programs.

**4.3. Message Processing Module**

The Message Processing Module (MPM) discussed by Postel [Postel79a] matches `sendmail` closely in terms of its basic architecture. However, like MMDF, the MPM includes the network interface software as part of its domain.

MPM also postulates a duplex channel to the receiver, as does MMDF, thus allowing simpler handling of errors by the mailer than is possible in `sendmail`. When a message queued by `sendmail` is sent, any errors must be returned to the sender by the mailer itself. Both MPM and MMDF mailers can return an immediate error response, and a single error processor can create an appropriate response.

MPM prefers passing the message as a structured object, with type-length-value tuples. Such a convention requires a much higher degree of cooperation between mailers than is required by `sendmail`. MPM also assumes a universally agreed upon internet name space (with each address in the form of a net-host-user tuple), which `sendmail` does not.

**5. EVALUATIONS AND FUTURE PLANS**

`Sendmail` is designed to work in a nonhomogeneous environment. Every attempt is made to avoid imposing unnecessary constraints on the underlying mailers. This goal has driven much of the design. One of the major problems has been the lack of a uniform address space, as postulated in [Postel79a] and [Postel79b].

---

4Dynamic configuration tables are currently being considered for MMDF, allowing the installer to select either compiled or dynamic tables.

5The MMDF equivalent of a `sendmail` “mailer.”

6This is similar to the NBS standard.
A nonuniform address space implies that a path will be specified in all addresses, either explicitly (as part of the address) or implicitly (as with implied forwarding to gateways). This restriction has the unpleasant effect of making replying to messages exceedingly difficult, since there is no one "address" for any person, but only a way to get there from wherever you are.

Interfacing to mail programs that were not initially intended to be applied in an internet environment has been amazingly successful, and has reduced the job to a manageable task.

Sendmail has knowledge of a few difficult environments built in. It generates ARPANET FTP/SMTP compatible error messages ( prepended with three-digit numbers [Neigus73, Postel74, Postel82]) as necessary, optionally generates UNIX-style "From" lines on the front of messages for some mailers, and knows how to parse the same lines on input. Also, error handling has an option customized for BerkNet.

The decision to avoid doing any type of delivery where possible (even, or perhaps especially, local delivery) has turned out to be a good idea. Even with local delivery, there are issues of the location of the mailbox, the format of the mailbox, the locking protocol used, etc., that are best decided by other programs. One surprisingly major annoyance in many internet mailers is that the location and format of local mail is built in. The feeling seems to be that local mail is so common that it should be efficient. This feeling is not born out by our experience; on the contrary, the location and format of mailboxes seems to vary widely from system to system.

The ability to automatically generate a response to incoming mail (by forwarding mail to a program) seems useful ("I am on vacation until late August.....") but can create problems such as forwarding loops (two people on vacation whose programs send notes back and forth, for instance) if these programs are not well written. A program could be written to do standard tasks correctly, but this would solve the general case.

It might be desirable to implement some form of load limiting. I am unaware of any mail system that addresses this problem, nor am I aware of any reasonable solution at this time.

The configuration file is currently practically inscrutable; considerable convenience could be realized with a higher-level format.

It seems clear that common protocols will be changing soon to accommodate changing requirements and environments. These changes will include modifications to the message header (e.g., [NBS80]) or to the body of the message itself (such as for multimedia messages [Postel80]). Experience indicates that these changes should be relatively trivial to integrate into the existing system.

In tightly coupled environments, it would be nice to have a name server such as Grapvine [Birrell82] integrated into the mail system. This would allow a site such as "Berkeley" to appear as a single host, rather than as a collection of hosts, and would allow people to move transparently among machines without having to change their addresses. Such a facility would require an automatically updated database and some method of resolving conflicts. Ideally this would be effective even without all hosts being under a single management. However, it is not clear whether this feature should be integrated into the aliasing facility or should be considered a "value added" feature outside sendmail itself.

As a more interesting case, the CSNET name server [Solomon81] provides an facility that goes beyond a single tightly-coupled environment. Such a facility would normally exist outside of sendmail however.

ACKNOWLEDGEMENTS

Thanks are due to Kurt Shoens for his continual cheerful assistance and good advice, Bill Joy for pointing me in the correct direction (over and over), and Mark Horton for more advice, prodding, and many of the good ideas. Kurt and Eric Schmidt are to be credited lor using delivermail as a server for their programs (Mail and BerkNet respectively) before any sane person should have, and making the necessary modifications promptly and happily. Eric gave me
considerable advice about the perils of network software which saved me an unknown amount of work and grief. Mark did the original implementation of the DBM version of aliasing, installed the VFORK code, wrote the current version of rmail, and was the person who really convinced me to put the work into delivermail to turn it into sendmail. Kurt deserves accolades for using sendmail when I was myself afraid to take the risk; how a person can continue to be so enthusiastic in the face of so much bitter reality is beyond me.

Kurt, Mark, Kirk McKusick, Marvin Solomon, and many others have reviewed this paper, giving considerable useful advice.

Special thanks are reserved for Mike Stonebraker at Berkeley and Bob Epstein at Britton-Lee, who both knowingly allowed me to put so much work into this project when there were so many other things I really should have been working on.

REFERENCES


TABLE OF CONTENTS

1. BASIC INSTALLATION ........................................................................................................ 1
   1.1. Off-The-Shelf Configurations ............................................................................... 2
   1.2. Installation Using the Makefile ......................................................................... 2
   1.3. Installation by Hand ......................................................................................... 2
       1.3.1. /usr/lib/sendmail ............................................................................... 2
       1.3.2. /usr/lib/sendmail.cf ........................................................................ 2
       1.3.3. /usr/ucb/newaliases ........................................................................... 2
       1.3.4. /usr/lib/sendmail.cf ........................................................................ 2
       1.3.5. /usr/spool/mqueue .............................................................................. 2
       1.3.6. /usr/lib/aliases* .................................................................................. 3
       1.3.7. /usr/lib/sendmail.fc ............................................................................ 3
       1.3.8. /etc/rc .................................................................................................. 3
       1.3.9. /usr/lib/sendmail.hl ............................................................................
       1.3.10. /usr/lib/sendmail.st ...........................................................................
       1.3.11. /usr/etc/in.syslog ................................................................................
       1.3.12. /usr/ucb/newaliases ............................................................................

2. NORMAL OPERATIONS ........................................................................................................ 4
   2.1. Quick Configuration Startup ............................................................................. 4
   2.2. The System Log ...................................................................................................... 4
       2.2.1. Format ................................................................................................. 4
       2.2.2. Levels ................................................................................................... 4
   2.3. The Mail Queue ...................................................................................................... 4
       2.3.1. Printing the queue ............................................................................... 4
       2.3.2. Format of queue files ............................................................................ 4
       2.3.3. Forcing the queue ................................................................................. 6
   2.4. The Alias Database ................................................................................................. 6
       2.4.1. Rebuilding the alias database ................................................................
       2.4.2. Potential problems .............................................................................. 7
       2.4.3. List owners ........................................................................................... 7
   2.5. Per-User Forwarding (.forward Files) ................................................................... 7
   2.6. Special Header Lines ............................................................................................. 8
       2.6.1. Return-Receipt-To: ............................................................................. 8
       2.6.2. Errors-To: ............................................................................................. 8
       2.6.3. Apparently-To: ..................................................................................... 8

3. ARGUMENTS ...................................................................................................................... 8
   3.1. Queue Interval ....................................................................................................... 8
   3.2. Daemon Mode ........................................................................................................ 8
   3.3. Forcing the Queue ................................................................................................ 8
   3.4. Debugging ............................................................................................................... 8
   3.5. Trying a Different Configuration File .................................................................... 9
   3.6. Changing the Values of Options .......................................................................... 9

4. TUNING .............................................................................................................................. 9
   4.1. Timeouts ................................................................................................................. 9
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1. Queue interval</td>
<td>9</td>
</tr>
<tr>
<td>4.1.2. Read timeouts</td>
<td>9</td>
</tr>
<tr>
<td>4.1.3. Message timeouts</td>
<td>10</td>
</tr>
<tr>
<td>4.2. Delivery Mode</td>
<td>10</td>
</tr>
<tr>
<td>4.3. Log Level</td>
<td>10</td>
</tr>
<tr>
<td>4.4. File Modes</td>
<td>10</td>
</tr>
<tr>
<td>4.4.1. To suid or not to suid?</td>
<td>10</td>
</tr>
<tr>
<td>4.4.2. Temporary file modes</td>
<td>11</td>
</tr>
<tr>
<td>4.4.3. Should my alias database be writable?</td>
<td>11</td>
</tr>
<tr>
<td>5. THE WHOLE SCOOP ON THE CONFIGURATION FILE</td>
<td>11</td>
</tr>
<tr>
<td>5.1. The Syntax</td>
<td>11</td>
</tr>
<tr>
<td>5.1.1. R and S - rewriting rules</td>
<td>11</td>
</tr>
<tr>
<td>5.1.2. D - define macro</td>
<td>12</td>
</tr>
<tr>
<td>5.1.3. C and F - define classes</td>
<td>12</td>
</tr>
<tr>
<td>5.1.4. M - define mailer</td>
<td>12</td>
</tr>
<tr>
<td>5.1.5. H - define header</td>
<td>12</td>
</tr>
<tr>
<td>5.1.6. O - set option</td>
<td>13</td>
</tr>
<tr>
<td>5.1.7. T - define trusted users</td>
<td>13</td>
</tr>
<tr>
<td>5.1.8. P - precedence definitions</td>
<td>13</td>
</tr>
<tr>
<td>5.2. The Semantics</td>
<td>13</td>
</tr>
<tr>
<td>5.2.1. Special macros, conditionals</td>
<td>13</td>
</tr>
<tr>
<td>5.2.2. Special classes</td>
<td>15</td>
</tr>
<tr>
<td>5.2.3. The left hand side</td>
<td>15</td>
</tr>
<tr>
<td>5.2.4. The right hand side</td>
<td>15</td>
</tr>
<tr>
<td>5.2.5. Semantics of rewriting rule sets</td>
<td>16</td>
</tr>
<tr>
<td>5.2.6. Mailer flags etc.</td>
<td>16</td>
</tr>
<tr>
<td>5.2.7. The &quot;error&quot; mailer</td>
<td>17</td>
</tr>
<tr>
<td>5.3. Building a Configuration File From Scratch</td>
<td>17</td>
</tr>
<tr>
<td>5.3.1. What you are trying to do</td>
<td>17</td>
</tr>
<tr>
<td>5.3.2. Philosophy</td>
<td>18</td>
</tr>
<tr>
<td>5.3.2.1. Large site, many hosts - minimum information</td>
<td>18</td>
</tr>
<tr>
<td>5.3.2.2. Small site - complete information</td>
<td>19</td>
</tr>
<tr>
<td>5.3.2.3. Single host</td>
<td>19</td>
</tr>
<tr>
<td>5.3.3. Relevant issues</td>
<td>19</td>
</tr>
<tr>
<td>5.3.4. How to proceed</td>
<td>19</td>
</tr>
<tr>
<td>5.3.5. Testing the rewriting rules - the -bt flag</td>
<td>20</td>
</tr>
<tr>
<td>5.3.6. Building mailer descriptions</td>
<td>20</td>
</tr>
<tr>
<td>Appendix A. COMMAND LINE FLAGS</td>
<td>22</td>
</tr>
<tr>
<td>Appendix B. CONFIGURATION OPTIONS</td>
<td>23</td>
</tr>
<tr>
<td>Appendix C. MAILER FLAGS</td>
<td>25</td>
</tr>
<tr>
<td>Appendix D. SUMMARY OF SUPPORT FILES</td>
<td>27</td>
</tr>
</tbody>
</table>
Sendmail implements a general purpose internetwork mail routing facility under the UNIX* operating system. It is not tied to any one transport protocol - its function may be likened to a crossbar switch, relaying messages from one domain into another. In the process, it can do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain. All of this is done under the control of a configuration file.

Due to the requirements of flexibility for sendmail, the configuration file can seem somewhat unapproachable. However, there are only a few basic configurations for most sites, for which standard configuration files have been supplied. Most other configurations can be built by adjusting an existing configuration file incrementally.

Although sendmail is intended to run without the need for monitoring, it has a number of features that may be used to monitor or adjust the operation under unusual circumstances. These features are described.

Section one describes how to do a basic sendmail installation. Section two explains the day-to-day information you should know to maintain your mail system. If you have a relatively normal site, these two sections should contain sufficient information for you to install sendmail and keep it happy. Section three describes some parameters that may be safely tweaked. Section four has information regarding the command line arguments. Section five contains the nitty-gritty information about the configuration file. This section is for masochists and people who must write their own configuration file. The appendices give a brief but detailed explanation of a number of features not described in the rest of the paper.

The references in this paper are actually found in the companion paper Sendmail – An Internetwork Mail Router. Read that paper before this one to gain a basic understanding of how the pieces fit together.

1. BASIC INSTALLATION

There are two basic steps to installing sendmail. The hard part is to build the configuration table. This is a file that sendmail reads when it starts up that describes the mailers it knows about, how to parse addresses, how to rewrite the message header, and the settings of various options. Although the configuration table is quite complex, a configuration can usually be built by adjusting an existing off-the-shelf configuration. The second part is actually doing the installation, i.e., creating the necessary files, etc.

The remainder of this section will describe the installation of sendmail assuming you can use one of the existing configurations and that the standard installation parameters are acceptable. All pathnames and examples are given from the root of the sendmail subtree.

*UNIX is a trademark of Bell Laboratories.
1.1. Off-The-Shelf Configurations

You can produce your own sendmail.cf file by editing the generic configuration file provided with this release: /usr/lib/sendmail.generic.cf. Procedures are given in the Setting Up the Mail System section of the System Set-up and Operation chapter.

1.2. Installation Using the Makefile

A makefile exists in the root of the sendmail directory that will do all of these steps for a 4.2BSD system. It may have to be slightly tailored for use on other systems.

Before using this makefile, you should already have created your configuration file and left it in the file "cf/system.cf" where *system* is the name of your system (i.e., what is returned by hostname(1)). If you do not have hostname you can use the declaration "HOST=system" on the make(1) command line. You should also examine the file m4/config.m4 and change the m4 macros there to reflect any libraries and compilation flags you may need.

The basic installation procedure is to type:

```make
make
make install
```

in the root directory of the sendmail distribution. This will make all binaries and install them in the standard places. The second make command must be executed as the superuser (root).

1.3. Installation by Hand

Along with building a configuration file, you will have to install the sendmail startup into your UNIX system. If you are doing this installation in conjunction with a regular Berkeley UNIX install, these steps will already be complete. Many of these steps will have to be executed as the superuser (root).

1.3.1. /usr/lib/sendmail

The binary for sendmail is located in /usr/lib.

1.3.2. /usr/lib/sendmail.cf

The configuration file that you created earlier should be installed in /usr/lib/sendmail.cf; see the Setting up the Mail System section in the System Set-up and Operation chapter.

1.3.3. /usr/ucb/newaliases

If you are running delivermail, it is critical that the newaliases command be replaced. This can just be a link to sendmail:

```bash
rm -f /usr/ucb/newaliases
ln /usr/lib/sendmail /usr/ucb/newaliases
```

1.3.4. /usr/lib/sendmail.cf

The configuration file must be installed in /usr/lib. This is described above.

1.3.5. /usr/spool/mqueue

The directory /usr/spool/mqueue should be created to hold the mail queue. This directory should be mode 777 unless sendmail is run setuid, when mqueue should be owned by the sendmail owner and mode 755.
1.3.1. /usr/lib/aliases

The file "/usr/lib/aliases" is the master file for system aliases. This file is a symbolic link to /private/aliases, used for clients.

1.3.7. /usr/lib/sendmail.fc

If you intend to install the frozen version of the configuration file (for quick startup) you should create the file /usr/lib/sendmail.fc and initialize it. This step may be safely skipped.

cp /dev/null /usr/lib/sendmail.fc
/usr/lib/sendmail -bx

1.3.8. /etc/rc

It will be necessary to start up the sendmail daemon when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically to insure that mail gets delivered when hosts come up.

Add the following lines to /etc/rc (or /etc/rc.local as appropriate) in the area where it is starting up the daemons:

if [-f /usr/lib/sendmail ]; then
  (cd /usr/spool/mqueue; rm -f [lnx]*)
  /usr/lib/sendmail -bd -q30m &
  echo -n ' sendmail' >/dev/console
fi

The cd and rm commands insure that all lock files have been removed; extraneous lock files may be left around if the system goes down in the middle of processing a message. The line that actually invokes sendmail has two flags: -bd causes it to listen on the SMTP port, and -q30m causes it to run the queue every half hour.

If you are not running a version of UNIX that supports Berkeley TCP/IP, do not include the -bd flag.

1.3.9. /usr/lib/sendmail.hf

This is the help file used by the SMTP HELP command. The file is already installed in the distribution.

1.3.10. /usr/lib/sendmail.st

If you wish to collect statistics about your mail traffic, create the file

/usr/lib/sendmail.st:

cp /dev/null /usr/lib/sendmail.st
chmod 666 /usr/lib/sendmail.st

This file does not grow. It is printed with the program aux/mailstats.

1.3.11. /usr/etc/in.syslog

You may want to run the syslog program (to collect log information about sendmail). This program normally resides in /usr/etc/in.syslog, with support files /etc/syslog.conf and /etc/syslog.pid. The file /etc/syslog.conf describes the file(s) that sendmail will log in. For a complete description of syslog, see the manual page for syslog(8).
1.3.13. `/usr/ucb/newaliases`

If `sendmail` is invoked as "newaliases," it will simulate the `-bl` flag (that is, will rebuild the alias database; see below). This should be a link to `/usr/lib/sendmail`.

2. NORMAL OPERATIONS

2.1. Quick Configuration Startup

A fast version of the configuration file may be set up by using the `-bu` flag:

```
/usr/lib/sendmail -bu
```

This creates the file `/usr/lib/sendmail.fc` ("frozen configuration"). This file is an image of `sendmail`'s data space after reading in the configuration file. If this file exists, it is used instead of `/usr/lib/sendmail.cf`. `sendmail.cf` time is changed.

The frozen configuration file will be ignored if a `-C` flag is specified or if `sendmail` detects that it is out of date. However, the heuristics are not strong so this should not be trusted.

2.2. The System Log

The system log is supported by the `syslog(8)` program.

2.2.1. Format

Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the ethernet), the word "sendmail:"

2.2.2. Levels

If you have `syslog(8)` or an equivalent installed, you will be able to do logging. `Syslog` installation is performed by the `setup` program during first-time UNIX installation for Sun systems. There is a large amount of information that can be logged. The log is arranged as a succession of levels. At the lowest level only extremely strange situations are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered "useful;" log levels above ten are usually for debugging purposes.

A complete description of the log levels is given in section 4.3.

2.3. The Mail Queue

The mail queue should be processed transparently. However, you may find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time the queue may become clogged. Although `sendmail` ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

2.3.1. Printing the queue

The contents of the queue can be printed by specifying the `-bp` flag to `sendmail`:

```
/usr/lib/sendmail -bp
```

This will produce a listing of the queue id's, the size of the message, the date the message entered the queue, and the sender and recipients.

2.3.2. Format of queue files

All queue files have the form `zfAA99999` where `AA99999` is the id for this file and the `x` is a type. The types are:
Sendmail Installation and Operation Guide

The data file. The message body (excluding the header) is kept in this file.

The lock file. If this file exists, the job is currently being processed, and a queue run will not process the file. For that reason, an extraneous /f file can cause a job to apparently disappear (it will not even time out!).

This file is created when an id is being created. It is a separate file to insure that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time.

The queue control file. This file contains the information necessary to process the job.

A temporary file. These are an image of the /f file when it is being rebuilt. It should be renamed to a /f file very quickly.

A transcript file, existing during the life of a session showing everything that happens during that session.

The /f file is structured as a series of lines each beginning with a code letter. The lines are as follows:

D The name of the data file. There may only be one of these lines.

H A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.

R A recipient address. This will normally be completely aliased, but is actually realiased when the job is processed. There will be one line for each recipient.

S The sender address. There may only be one of these lines.

T The job creation time. This is used to compute when to time out the job.

P The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.

M A message. This line is printed by using sendmail with the –bp flag, and is generally used to store status information. It can contain any text.

As an example, the following is a queue file sent to “mckusick@calder” and “wnj”:

DdfA13557
Seric
T404261372
P132
Rmckusick@calder
Rwnj
H?F?from: erie (Eric Allman)
H?x?full-name: Eric Allman
Hsubject: this is an example message
Hmessage-id: <8209232249.13557@UCBARPA.BERKELEY.ARPA>
Hreceived: by UCBARPA.BERKELEY.ARPA (3.227 [10/22/82])
   id A13557; 23-Oct-82 15:49:32-PDT (Sat)
Hphone: (415) 548-3211
HTo: mckusick@calder, wnj

This shows the name of the data file, the person who sent the message, the submission time (in seconds since January 1, 1970), the message priority, the message class, the recipients, and the headers for the message.
2.3.3. Forcing the queue

Sendmail should run the queue automatically at intervals. The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, sendmail first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to insure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process. Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no way to resolve this without violating the protocol.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in sendmail spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory:

```
cd /usr/spool
mv mqueue omqueue; mkdir mqueue; chmod 777 mqueue
```

You should then kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon.

To run the old mail queue, run the following command:

```
/usr/lib/sendmail -oQ/usr/spool/omqueue -q
```

The -oQ flag specifies an alternate queue directory and the -q flag says to just run every job in the queue. If you have a tendency toward voyeurism, you can use the -v flag to watch what is going on.

When the queue is finally emptied, you can remove the directory:

```
rm -d /usr/spool/omqueue
```

2.4. The Alias Database

The alias database exists in two forms. One is a text form, maintained in the file /usr/lib/aliases. The aliases are of the form

```
name: name1, name2, ...
```

Only local names may be aliased; e.g.,

```
eric@mit-xx: eric@berkeley
```

will not have the desired effect. Aliases may be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a sharp sign (“#”) are comments.

The second form is processed by the dbm(3) library. This form is in the files /usr/lib/aliases.dir and /usr/lib/aliases.pag. This is the form that sendmail actually uses to resolve aliases. This technique is used to improve performance.

2.4.1. Rebuilding the alias database

The DBM version of the database may be rebuilt explicitly by executing the command

```
newaliases
```

This is equivalent to giving sendmail the -bl flag:
/usr/lib/sendmail -bi

If the "D" option is specified in the configuration, sendmail will rebuild the alias database automatically if possible when it is out of date. The conditions under which it will do this are:

1. The DBM version of the database is mode 666. —or—
2. Sendmail is running setuid to root.

Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

2.4.2. Potential problems

There are a number of problems that can occur with the alias database. They all result from a sendmail process accessing the DBM version while it is only partially built. This can happen under two circumstances: One process accesses the database while another process is rebuilding it, or the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

Sendmail has two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, at the end of the rebuild it adds an alias of the form

0: 0

(which is not normally legal). Before sendmail will access the database, it checks to insure that this entry exists1. It will wait up to five minutes for this entry to appear, at which point it will force a rebuild itself2.

2.4.3. List owners

If an error occurs on sending to a certain address, say "z", sendmail will look for an alias of the form "owner-z" to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example:

unix-wizards: eric@ucbarpa, wnj@monet, nosuchuser, sam@matisse
owner-unix-wizards: eric@ucbarpa

would cause "eric@ucbarpa" to get the error that will occur when someone sends to unix-wizards due to the inclusion of "nosuchuser" on the list.

2.5. Per-User Forwarding (.forward Files)

As an alternative to the alias database, any user may put a file with the name ".forward" in his or her home directory. If this file exists, sendmail redirects mail for that user to the list of addresses listed in the .forward file. For example, if the home directory for user "mckusick" has a .forward file with contents:

mckusick@ernie
kirk@calder

then any mail arriving for "mckusick" will be redirected to the specified accounts.

1The "a" option is required in the configuration for this action to occur. This should normally be specified unless you are running delivermail in parallel with sendmail.
2Note: the "D" option must be specified in the configuration file for this operation to occur.
2.6. Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into sendmail that cannot be changed without changing the code. These builtins are described here.

2.6.1. Return-Receipt-To:

If this header is sent, a message will be sent to any specified addresses when the final delivery is complete. If the mailer has the l flag (local delivery) set in the mailer descriptor.

2.6.2. Errors-To:

If errors occur anywhere during processing, this header will cause error messages to go to the listed addresses rather than to the sender. This is intended for mailing lists.

2.6.3. Apparently-To:

If a message comes in with no recipients listed in the message (in a To:, Cc:, or Bcc: line) then sendmail will add an "Apparently-To:" header line for any recipients it is aware of. This is not put in as a standard recipient line to warn any recipients that the list is not complete.

At least one recipient line is required under RFC 822.

3. ARGUMENTS

The complete list of arguments to sendmail is described in detail in Appendix A. Some important arguments are described here.

3.1. Queue Interval

The amount of time between forking a process to run through the queue is defined by the -q flag. If you run in mode f or a this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in q mode it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue.

3.2. Daemon Mode

If you allow incoming mail over an IPC connection, you should have a daemon running. This should be set by your /etc/rc file using the -bd flag. The -bd flag and the -q flag may be combined in one call:

/usr/lib/sendmail -bd -q30m

3.3. Forcing the Queue

In some cases you may find that the queue has gotten clogged for some reason. You can force a queue run using the -q flag (with no value). It is entertaining to use the -v flag (verbose) when this is done to watch what happens:

/usr/lib/sendmail -q -v

3.4. Debugging

There are a fairly large number of debug flags built into sendmail. Each debug flag has a number and a level, where higher levels mean to print out more information. The convention is that levels greater than nine are "absurd," because they print out so much information that you wouldn't normally want to see them except for debugging that
particular piece of code. Debug flags are set using the -d option; the syntax is:

- debug-flag: -d debug-list
- debug-list: debug-option | debug-option
- debug-option: debug-range | debug-level
- debug-range: integer | integer - integer
- debug-level: integer

where spaces are for reading ease only. For example,

- -d12  Set flag 12 to level 1
- -d12.3 Set flag 12 to level 3
- -d3-17 Set flags 3 through 17 to level 1
- -d3-17.4 Set flags 3 through 17 to level 4

For a complete list of the available debug flags you will have to look at the code (they are too dynamic to keep this documentation up to date).

3.5. Trying a Different Configuration File

An alternative configuration file can be specified using the -C flag; for example,

```
/usr/lib/sendmail -C test.cf
```

uses the configuration file test.cf instead of the default /usr/lib/sendmail.cf. If the -C flag has no value it defaults to sendmail.cf in the current directory.

3.6. Changing the Values of Options

Options can be overridden using the -o flag. For example,

```
/usr/lib/sendmail -oT2m
```

sets the T (timeout) option to two minutes for this run only.

4. TUNING

There are a number of configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line “OT3d” sets option T to the value “3d” (three days).

4.1. Timeouts

All time intervals are set using a scaled syntax. For example, “10m” represents ten minutes, whereas “2h30m” represents two and a half hours. The full set of scales is:

- s seconds
- m minutes
- h hours
- d days
- w weeks

4.1.1. Queue Interval

The argument to the -q flag specifies how often a subdaemon will run the queue. This is typically set to between five minutes and one half hour.

4.1.2. Read timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. Technically, this is not acceptable within the published protocols. However, it might be appropriate to set it to something large in certain environments (such as an hour). This will reduce the chance of large numbers of idle daemons piling up on your system. This timeout is set using the r option in the configuration
4.1.3. Message timeouts

After sitting in the queue for a few days, a message will time out. This is to
insure that at least the sender is aware of the inability to send a message. The timeout
is typically set to three days. This timeout is set using the T option in the
configuration file.

The time of submission is set in the queue, rather than the amount of time left
until timeout. As a result, you can flush messages that have been hanging for a short
period by running the queue with a short message timeout. For example,

```
/usr/lib/sendmail -OT1d -q
```

will run the queue and flush anything that is one day old.

4.2. Delivery Mode

There are a number of delivery modes that sendmail can operate in, set by the "d"
configuration option. These modes specify how quickly mail will be delivered. Legal
modes are:

- i deliver interactively (synchronously)
- b deliver in background (asynchronously)
- q queue only (don't deliver)

There are tradeoffs. Mode "i" passes the maximum amount of information to the sender,
but is hardly ever necessary. Mode "q" puts the minimum load on your machine, but
means that delivery may be delayed for up to the queue interval. Mode "b" is probably a
good compromise. However, this mode can cause large numbers of processes if you have a
mailer that takes a long time to deliver a message.

4.3. Log Level

The level of logging can be set for sendmail. The default using a standard
configuration table is level 9. The levels are as follows:

- 0 No logging.
- 1 Major problems only.
- 2 Message collections and failed deliveries.
- 3 Successful deliveries.
- 4 Messages being deferred (due to a host being down, etc.).
- 5 Normal message queueups.
- 6 Unusual but benign incidents, e.g., trying to process a locked queue file.
- 9 Log internal queue id to external message id mappings. This can be useful for trac-
ing a message as it travels between several hosts.
- 12 Several messages that are basically only of interest when debugging.
- 16 Verbose information regarding the queue.

4.4. File Modes

There are a number of files that may have a number of modes. The modes depend
on what functionality you want and the level of security you require.

4.4.1. To suid or not to suid?

Sendmail can safely be made setuid to root. At the point where it is about to
eexec(2) a mailer, it checks to see if the userid is zero; if so, it resets the userid and
groupid to a default (set by the u and g options). (This can be overridden by setting the S flag to the mailer for mailers that are trusted and must be called as root.) However, this will cause mail processing to be accounted (using sa(8)) to root rather than to the user sending the mail.

4.4.2. Temporary file modes

The mode of all temporary files that sendmail creates is determined by the F option. Reasonable values for this option are 0600 and 0644. If the more permissive mode is selected, it will not be necessary to run sendmail as root at all (even when running the queue).

4.4.3. Should my alias database be writable?

At Sun Microsystems, we have the alias database (/usr/lib/aliases) mode 666. There are some dangers inherent in this approach: any user can add him-/her-self to any list, or can read any other user's mail. However, we have found users to be basically trustworthy, and the cost of having a read-only database greater than the expense of finding and eradicating the rare nasty person.

5. THE WHOLE SCOOP ON THE CONFIGURATION FILE

This section describes the configuration file in detail, including hints on how to write one of your own if you have to.

There is one point that should be made clear immediately: the syntax of the configuration file is designed to be reasonably easy to parse, since this is done every time sendmail starts up, rather than easy for a human to read or write. On the "future project" list is a configuration-file compiler.

An overview of the configuration file is given first, followed by details of the semantics.

5.1. The Syntax

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a sharp symbol ('#') are comments.

5.1.1. R and S – rewriting rules

The core of address parsing are the rewriting rules. These are an ordered production system. Sendmail scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the address is replaced by the right hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and may be referenced by the mailer definitions or by other rewriting sets.

The syntax of these two commands are:

Sn
Sets the current ruleset being collected to n. If you begin a ruleset more than once it deletes the old definition.

R lhs rhs comments
The fields must be separated by at least one tab character; there may be embedded spaces in the fields. The lhs is a pattern that is applied to the input. If it matches, the input is rewritten to the rhs. The comments are ignored.
5.1.2. D — define macro

Macros are named with a single character. These may be selected from the entire ASCII set, but user-defined macros should be selected from the set of upper case letters only. Lower case letters and special symbols are used internally.

The syntax for macro definitions is:

\[D\text{val}\]

where \(z\) is the name of the macro and \(val\) is the value it should have. Macros can be interpolated in most places using the escape sequence \(\$z\).

5.1.3. C and F — define classes

Classes of words may be defined to match on the left hand side of rewriting rules. For example a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes may be given names from the set of upper case letters. Lower case letters and special characters are reserved for system use.

The syntax is:

\[C\text{c\text{word\text{word\text{...}}}\text{F\text{c\text{file\text{[\text{format}\text{)}}}}}}\]

The first form defines the class \(c\) to match any of the named words. It is permissible to split them among multiple lines; for example, the two forms:

\[\text{CHmonet ucbmonet}\]

and

\[\text{CHmonet}\]
\[\text{CHucbmonet}\]

are equivalent. The second form reads the elements of the class \(c\) from the named \text{file}; the \text{format} is a \text{scanf}(3) pattern that should produce a single string.

5.1.4. M — define mailer

Programs and interfaces to mailers are defined in this line. The format is:

\[M\text{name, \{field=value\}*}\]

where \text{name} is the name of the mailer (used internally only) and the "field=name" pairs define attributes of the mailer. Fields are:

- \text{Path} — The pathname of the mailer
- \text{Flags} — Special flags for this mailer
- \text{Sender} — A rewriting set for sender addresses
- \text{Recipient} — A rewriting set for recipient addresses
- \text{Argv} — An argument vector to pass to this mailer
- \text{Eol} — The end-of-line string for this mailer
- \text{Maxsize} — The maximum message length to this mailer

Only the first character of the field name is checked.

5.1.5. H — define header

The format of the header lines that \text{sendmail} inserts into the message are defined by the H line. The syntax of this line is:

\[H[\text{mflags}]\text{\{names htemplate\}}\]

Continuation lines in this spec are reflected directly into the outgoing message. The \text{htemplate} is macro expanded before insertion into the message. If the \text{mflags}
(surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input it is reflected to the output regardless of these flags.

Some headers have special semantics that will be described below.

5.1.6. O - set option

There are a number of "random" options that can be set from a configuration file. Options are represented by single characters. The syntax of this line is:

\[\text{O} \text{ value}\]

This sets option \(O\) to be \textit{value}. Depending on the option, \textit{value} may be a string, an integer, a boolean (with legal values "t", "T", "f", or "F"; the default is TRUE), or a time interval.

5.1.7. T - define trusted users

Trusted users are those users who are permitted to override the sender address using the \(-f\) flag. These typically are "root," "uucp," and "network," but on some users it may be convenient to extend this list to include other users, perhaps to support a separate UUCP login for each host. The syntax of this line is:

\[\text{T} \text{user1 user2...}\]

There may be more than one of these lines.

5.1.8. P - precedence definitions

Values for the "Precedence:" field may be defined using the \(P\) control line. The syntax of this field is:

\[\text{P} \text{name=num}\]

When the \textit{name} is found in a "Precedence:" field, the message class is set to \textit{num}. Higher numbers mean higher precedence. Numbers less than zero have the special property that error messages will not be returned. The default precedence is zero. For example, our list of precedences is:

\[\text{Pfirst-class=0}\]
\[\text{Pspecial-delivery=100}\]
\[\text{Pjunk=-100}\]

5.2. The Semantics

This section describes the semantics of the configuration file.

5.2.1. Special macros, conditionals

Macros are interpolated using the construct \$x, where \(x\) is the name of the macro to be interpolated. In particular, lower case letters are reserved to have special semantics, used to pass information in or out of \textit{sendmail}, and some special characters are reserved to provide conditionals, etc.

The following macros must be defined to transmit information into \textit{sendmail}:

\[\text{e} \text{ The SMTP entry message}\]
\[\text{j} \text{ The "official" domain name for this site}\]
\[\text{l} \text{ The format of the UNIX from line}\]
\[\text{n} \text{ The name of the daemon (for error messages)}\]
\[\text{o} \text{ The set of "operators" in addresses}\]
\[\text{q} \text{ default format of sender address}\]

The \$e macro is printed out when SMTP starts up. The first word must be the \$j
The $j$ macro should be in RFC821 format. The $s_i$ and $s_n$ macros can be considered constants except under terribly unusual circumstances. The $s_0$ macro consists of a list of characters which will be considered tokens and which will separate tokens when doing parsing. For example, if "$r" were in the $s_0$ macro, then the input "address" would be scanned as three tokens: "add," "r," and "ess." Finally, the $s_q$ macro specifies how an address should appear in a message when it is defaulted. For example, on our system these definitions are:

De$j$ Sendmail $v$ ready at $b$
DnMAILER-DAEMON
DIFrom $s_f$ $d$
Do.:%@!"=/
Dq$s_g$x ($x)s.$
Dj$:H.$D

An acceptable alternative for the $s_q$ macro is "$?x$x $.$<$g$>". These correspond to the following two formats:

eric@Berkeley (Eric Allman)
Eric Allman <eric@Berkeley>

Some macros are defined by sendmail for interpolation into argv's for mailers or for other contexts. These macros are:

- a: The origination date in Arpanet format
- b: The current date in Arpanet format
- c: The hop count
- d: The date in UNIX (ctime) format
- f: The sender (from) address
- g: The sender address relative to the recipient
- h: The recipient host
- i: The queue id
- p: Sendmail's pid
- r: Protocol used
- s: Sender's host name
- t: A numeric representation of the current time
- u: The recipient user
- v: The version number of sendmail
- w: The hostname of this site
- x: The full name of the sender
- y: The id of the sender's tty
- z: The home directory of the recipient

There are three types of dates that can be used. The $s_a$ and $s_b$ macros are in Arpanet format; $s_a$ is the time as extracted from the "Date:" line of the message (if there was one), and $s_b$ is the current date and time (used for postmarks). If no "Date:" line is found in the incoming message, $s_a$ is set to the current time also. The $s_d$ macro is equivalent to the $s_a$ macro in UNIX (ctime) format.

The $s_f$ macro is the id of the sender as originally determined; when mailing to a specific host the $s_g$ macro is set to the address of the sender relative to the recipient. For example, if I send to "bollard@matisse" from the machine "ucharpaa" the $s_f$ macro will be "eric" and the $s_g$ macro will be "eric@ucharpaa."

The $s_x$ macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to sendmail. The second choice is the value of the "Full-name:" line in the header if it exists, and the third choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the /etc/passwd file.
When sending, the $h, $u, and $a macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the $Q and $i part of the rewriting rules, respectively.

The $p and $t macros are used to create unique strings (e.g., for the "Message-Id:" field). The $t macro is set to the queue id on this host; if put into the timestamp line it can be extremely useful for tracking messages. The $y macro is set to the id of the terminal of the sender (if known); some systems like to put this in the Unix "From" line. The $v macro is set to be the version number of sendmail; this is normally put in timestamps and has been proven extremely useful for debugging. The $w macro is set to the name of this host if it can be determined. The $c field is set to the "hop count," i.e., the number of times this message has been processed. This can be determined by the -h flag on the command line or by counting the timestamps in the message.

The $x and $s fields are set to the protocol used to communicate with sendmail and the sending hostname; these are not supported in the current version.

Conditionals can be specified using the syntax:

```console
$?x text1 $! text2 $.
```

This interpolates text1 if the macro $x is set, and text2 otherwise. The "else" ($!) clause may be omitted.

### 5.2.2. Special classes

The class $==w is set to be the set of all names this host is known by. This can be used to delete local hostnames.

### 5.2.3. The left hand side

The left hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced using a dollar sign. The metasymbols are:

- $* Match zero or more tokens
- $+ Match one or more tokens
- $ Match exactly one token
- $==x Match any token in class x
- $^x Match any token not in class x

If any of these match, they are assigned to the symbol $n for replacement on the right hand side, where n is the index in the LHS. For example, if the LHS:

```console
$*:$+
```

is applied to the input:

```console
UCBARPA:eric
```

the rule will match, and the values passed to the RHS will be:

```console
$1 UCBARPA
$2 eric
```

### 5.2.4. The right hand side

When the right hand side of a rewriting rule matches, the input is deleted and replaced by the right hand side. Tokens are copied directly from the RHS unless they are begin with a dollar sign. Metasymbols are:
The $n$ syntax substitutes the corresponding value from a $+$, $-$, $*$, $-$, or $-$ match on the LHS. It may be used anywhere.

The $>$ syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset $n$. The final value of ruleset $n$ then becomes the substitution for this rule.

The $#$ syntax should only be used in ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to sendmail that the address has completely resolved. The complete syntax is:

$#$ mailer$host$user

This specifies the (mailer, host, user) 3-tuple necessary to direct the mailer. If the mailer is local the host part may be omitted. The mailer and host must be a single word, but the user may be multi-part.

A RHS may also be preceeded by a $|$ or a $.$ to control evaluation. A $|$ prefix causes the ruleset to return with the remainder of the RHS as the value. A $.$ prefix causes the rule to terminate immediately, but the ruleset to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The $|$ and $.$ prefixes may preceed a $>$ spec; for example:

R$+$ $|$ $>$ $|$ $1$

matches anything, passes that to ruleset seven, and continues; the $.$ is necessary to avoid an infinite loop.

5.2.5. Semantics of rewriting rule sets

There are five rewriting sets that have specific semantics. These are related as depicted by Figure 2.

Ruleset three should turn the address into "canonical form." This form should have the basic syntax:

local-part$host$domain-spec

If no "$" sign is specified, then the host-domain-spec may be appended from the sender address (if the C flag is set in the mailer definition corresponding to the sending mailer). Ruleset three is applied by sendmail before doing anything with any address.

Ruleset zero is applied after ruleset three to addresses that are going to actually specify recipients. It must resolve to a (mailer, host, user) triple. The mailer must be defined in the mailer definitions from the configuration file. The host is defined into the $h$ macro for use in the argv expansion of the specified mailer.

Rulesets one and two are applied to all sender and recipient addresses respectively. They are applied before any specification in the mailer definition. They must never resolve.

Ruleset four is applied to all addresses in the message. It is typically used to translate internal to external form.

5.2.6. Mailer flags etc.

There are a number of flags that may be associated with each mailer, each identified by a letter of the alphabet. Many of them are assigned semantics internally. These are detailed in Appendix C. Any other flags may be used freely to conditionally
assign headers to messages destined for particular mailers.

5.2.7. The “error” maller

The maller with the special name “error” can be used to generate a user error. The (optional) host field is a numeric exit status to be returned, and the user field is a message to be printed. For example, the entry:

```
$.#error$:Host unknown in this domain
```
on the RHS of a rule will cause the specified error to be generated if the LHS matches. This maller is only functional in ruleset zero.

5.3. Building a Configuration File From Scratch

Building a configuration table from scratch is an extremely difficult job. Fortunately, it is almost never necessary to do so; nearly every situation that may come up may be resolved by changing an existing table. In any case, it is critical that you understand what it is that you are trying to do and come up with a philosophy for the configuration table. This section is intended to explain what the real purpose of a configuration table is and to give you some ideas for what your philosophy might be.

5.3.1. What you are trying to do

The configuration table has three major purposes. The first and simplest is to set up the environment for sendmail. This involves setting the options, defining a few critical macros, etc. Since these are described in other places, we will not go into more detail here.

The second purpose is to rewrite addresses in the message. This should typically be done in two phases. The first phase maps addresses in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving maller. Sendmail does this in three subphases. Rulesets one and two are applied to all sender and recipient addresses respectively. After this, you may specify per-maller rulesets for both sender and recipient
addresses; this allows mailer-specific customization. Finally, ruleset four is applied to do any default conversion to external form.

The third purpose is to map addresses into the actual set of instructions necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

5.3.2. Philosophy

The particular philosophy you choose will depend heavily on the size and structure of your organization. I will present a few possible philosophies here.

One general point applies to all of these philosophies: it is almost always a mistake to try to do full name resolution. For example, if you are trying to get names of the form "user@host" to the Arpanet, it does not pay to route them to "xyzvax!devvax!ucbvax!c70:user@host" since you then depend on several links not under your control. The best approach to this problem is to simply forward to "xyzvax!user@host" and let xyzvax worry about it from there. In summary, just get the message closer to the destination, rather than determining the full path.

5.3.2.1. Large site, many hosts - minimum information

Berkeley is an example of a large site; it has more than two or three hosts. Berkeley has decided that the only reasonable philosophy for their environment is to designate one host as site guru. It must be able to resolve any piece of mail it receives. The other sites should have the minimum amount of information they can get away with, and even this minimum should be hints rather than solid information.

For example, a typical site on the Berkeley local ether network is "monet." Monet has a list of known ethernet hosts; if it receives mail for any of them, it can do direct delivery. If it receives mail for any unknown host, it just passes it directly to "ucbvax," the Berkeley master host. Ucbvax may determine that the host name is illegal and reject the message, or may be able to do delivery. However, it is important to note that when a new ethernet host is added, the only host that must have its tables updated is ucbvax; the others may be updated as convenient, but this is not critical.

This picture is slightly muddied due to network connections that are not actually located on ucbvax. For example, the Berkeley TCP connection is currently on "ucbarpa." However, monet does not know about this; the information is hidden between ucbvax and ucbarpa. Mail going from monet to a TCP host is transferred via the ethernet from monet to ucbvax, then via the ethernet from ucbvax to ucbarpa, and then is submitted to the Arpanet. Although this involves some extra hops, Berkeley feels this is an acceptable tradeoff.

An interesting point is that it would be possible to update monet to send TCP mail directly to ucbarpa if the load got too high: if monet failed to note a host as a TCP host, the mail would go via ucbvax as before; and if monet incorrectly sent a message to ucbarpa, it would still be sent by ucbarpa to ucbvax as before. The only problem that might occur in this scenario is 'looping': if ucbarpa thought that ucbvax had the TCP connection and vice versa. For this reason, updates should always happen to the master host first.

This philosophy results as much from the need to have a single source for the configuration files (typically built using $M4(1)$ or some similar tool) as any logical need. Maintaining more than three separate tables by hand is essentially an impossible job.
5.3.3. Small site - complete information

A small site (two or three hosts) may find it more reasonable to have complete information at each host. This would require that each host know exactly where each network connection is, possibly including the names of each host on that network. As long as the site remains small and the configuration remains relatively static, the update problem will probably not be too great.

5.3.3.1. Single host

This is in some sense the trivial case. The only major issue is trying to insure that you don't have to know too much about your environment. For example, if you have a UUCP connection you might find it useful to know about the names of hosts connected directly to you, but this is really not necessary since this may be determined from the syntax.

5.3.3. Relevant issues

The canonical form you use should almost certainly be as specified in the Arpanet protocols RFC819 and RFC822. Copies of these RFC's are included on the sendmail tape as doc/rfc819.lpr and doc/rfc822.lpr.

RFC822 describes the format of the mail message itself. Sendmail follows this RFC closely, to the extent that many of the standards described in this document can not be changed without changing the code. In particular, the following characters have special interpretations:

\(<\>\(\)"\"

Any attempt to use these characters for other than their RFC822 purpose in addresses is probably doomed to disaster.

RFC819 describes the specifics of the domain-based addressing. This is touched on in RFC822 as well. Essentially each host is given a name which is a right-to-left dot qualified pseudo-path from a distinguished root. The elements of the path need not be physical hosts; the domain is logical rather than physical. For example, at Berkeley one legal host is "a.cc.berkeley.arpa". Reading from right to left, "arpa" is a top level domain (related to, but not limited to, the physical Arpanet), "berkeley" is both an Arpanet host and a logical domain which is actually interpreted by a host called ucvax (the "major" host for this domain), "cc" represents the Computer Center, (in this case a strictly logical entity), and "a" is a host in the Computer Center; this particular host happens to be connected via berknet, the Berkeley network, but other hosts might be connected via one of two eternets or some other network.

Be aware when reading RFC819 that there are a number of errors in it.

5.3.4. How to proceed

Once you have decided on a philosophy, it is worth examining the available configuration tables to decide if any of them are close enough to steal major parts of. Even under the worst of conditions, there is a fair amount of boiler plate that can be collected safely.

The next step is to build ruleset three. This will be the hardest part of the job. Beware of doing too much to the address in this ruleset, since anything you do will reflect through to the message. In particular, stripping of local domains is best deferred, since this can leave you with addresses with no domain spec at all. Since sendmail likes to append the sending domain to addresses with no domain, this can change the semantics of addresses. Also try to avoid fully qualifying domains in this ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long addresses reflected into messages. The Berkeley configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get
all addresses into a cleaner form.

Once you have ruleset three finished, the other rulesets should be relatively trivial. If you need hints, examine the supplied configuration tables.

5.3.5. Testing the rewriting rules — the -bt flag

When you build a configuration table, you can do a certain amount of testing using the "test mode" of sendmail. For example, you could invoke sendmail as:

sendmail -bt -Ctest.cf

which would read the configuration file test.cf and enter test mode. In this mode, you enter lines of the form:

rwset address

where rwset is the rewriting set you want to use and address is an address to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the address it ends up with. You may use a comma separated list of rwsets for sequential application of rules to an input; ruleset three is always applied first. For example:

1,2,4 monet:bollard

first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the -d21 flag to turn on more debugging. For example,

sendmail -bt -d21.99

turns on an incredible amount of information; a single word address is probably going to print out several pages worth of information.

5.3.6. Building mailer descriptions

To add an outgoing mailer to your mail system, you will have to define the characteristics of the mailer.

Each mailer must have an internal name. This can be arbitrary, except that the names "local" and "prog" must be defined.

The pathname of the mailer must be given in the P field. If this mailer should be accessed via an IPC connection, use the string "[IPC]" instead.

The F field defines the mailer flags. You should specify an "f" or "r" flag to pass the name of the sender as a -f or -r flag respectively. These flags are only passed if they were passed to sendmail, so that mailers that give errors under some circumstances can be placated. If the mailer is not picky you can just specify "-f $g" in the argv template. If the mailer must be called as root the "S" flag should be given; this will not reset the userid before calling the mailer. If this mailer is local (i.e., will perform final delivery rather than another network hop) the "l" flag should be given. Quote characters (backslashes and " marks) can be stripped from addresses if the "s" flag is specified; if this is not given they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction the "m" flag should be stated. If this flag is on, then the argv template containing $u will be repeated for each unique user on a given host. The "e" flag will mark the mailer as being "expensive," which will cause sendmail to defer connection until a queue run.

An unusual case is the "C" flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if set, the domain spec of the

---

Sendmail must be running setuid to root for this to work.

The "c" configuration option must be given for this to be effective.
sender (i.e., the "@domain" part) is saved and is appended to any addresses in the message that do not already contain a domain spec. For example, a message of the form:

From: eric@ucbarpa
To: wnj@monet, mckusick

will be modified to:

From: eric@ucbarpa
To: wnj@monet, mckusick@ucbarpa

If and only if the "C" flag is defined in the mailer corresponding to "eric@ucbarpa."

Other flags are described in Appendix C.

The S and R fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses respectively. These are applied after the sending domain is appended and the general rewriting sets (numbers one and two) are applied, but before the output rewrite (ruleset four) is applied. A typical use is to append the current domain to addresses that do not already have a domain. For example, a header of the form:

From: eric

might be changed to be:

From: eric@ucbarpa

or

From: ucbvax!eric

depending on the domain it is being shipped into. These sets can also be used to do special purpose output rewriting in cooperation with ruleset four.

The E field defines the string to use as an end-of-line indication. A string containing only newline is the default. The usual backslash escapes (\r, \n, \t, \b) may be used.

Finally, an argv template is given as the E field. It may have embedded spaces. If there is no argv with a $u macro in it, sendmail will speak SMTP to the mailer. If the pathname for this mailer is "[IPC]," the argv should be

```
IPC $h [ port ]
```

where port is the optional port number to connect to.

For example, the specifications:

```
Mlocal, P=/bin/mail, F=rlsm S=10, R=20, A=mail -d $u
Mether,P=[IPC], F=meC, S=11, R=21, A=IPC $h, M=100000
```

specifies a mailer to do local delivery and a mailer for ethernet delivery. The first is called "local," is located in the file "/bin/mail," takes a picky -r flag, does local delivery, quotes should be stripped from addresses, and multiple users can be delivered at once; ruleset ten should be applied to sender addresses in the message and ruleset twenty should be applied to recipient addresses; the argv to send to a message will be the word "mail," the word "-d," and words containing the name of the receiving user. If a -r flag is inserted it will be between the words "mail" and "-d." The second mailer is called "ether," it should be connected to via an IPC connection, it can handle multiple users at once, connections should be deferred, and any domain from the sender address should be appended to any receiver name without a domain; sender addresses should be processed by ruleset eleven and recipient addresses by ruleset twenty-one. There is a 100,000 byte limit on messages passed through this mailer.
APPENDIX A

COMMAND LINE FLAGS

Arguments must be presented with flags before addresses. The flags are:

- \texttt{-f \textit{addr}}
  The sender's machine address is \textit{addr}. This flag is ignored unless the real user is listed as a "trusted user" or if \textit{addr} contains an exclamation point (because of certain restrictions in UUCP).

- \texttt{-r \textit{addr}}
  An obsolete form of \texttt{-f}.

- \texttt{-h \textit{cnt}}
  Sets the "hop count" to \textit{cnt}. This represents the number of times this message has been processed by \texttt{sendmail} (to the extent that it is supported by the underlying networks). \textit{Cnt} is incremented during processing, and if it reaches MAXHOP (currently 30) \texttt{sendmail} throws away the message with an error.

- \texttt{-Fname}
  Sets the full name of this user to \textit{name}.

- \texttt{-n}
  Don't do aliasing or forwarding.

- \texttt{-t}
  Read the header for "To:", "Cc:“, and "Bcc:" lines, and send to everyone listed in those lists. The "Bcc:" line will be deleted before sending. Any addresses in the argument vector will be deleted from the send list.

- \texttt{-bz}
  Set operation mode to \texttt{z}. Operation modes are:
  \begin{itemize}
  \item \texttt{m} Deliver mail (default)
  \item \texttt{a} Run in arpanet mode (see below)
  \item \texttt{s} Speak SMTP on input side
  \item \texttt{d} Run as a daemon
  \item \texttt{t} Run in test mode
  \item \texttt{v} Just verify addresses, don't collect or deliver
  \item \texttt{i} Initialize the alias database
  \item \texttt{p} Print the mail queue
  \item \texttt{z} Freeze the configuration file
  \end{itemize}

  The special processing for the ARPANET includes reading the "From:" line from the header to find the sender, printing ARPANET style messages (preceded by three digit reply codes for compatibility with the FTP protocol [Neigus73, Postel74, Postel77]), and ending lines of error messages with \texttt{<CRLF>}.

- \texttt{-qtime}
  Try to process the queued up mail. If the time is given, a \texttt{sendmail} will run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.

- \texttt{-Cfile}
  Use a different configuration file.

- \texttt{-dlevel}
  Set debugging level.

- \texttt{-ozvalue}
  Set option \texttt{z} to the specified \textit{value}. These options are described in Appendix B.

There are a number of options that may be specified as primitive flags (provided for compatibility with \texttt{delivermail}). These are the \texttt{e}, \texttt{i}, \texttt{m}, and \texttt{v} options. Also, the \texttt{f} option may be specified as the \texttt{-s} flag.

Sendmail Installation and Operation Guide

22
APPENDIX B

CONFIGURATION OPTIONS

The following options may be set using the -o flag on the command line or the O line in the configuration file:

Afile
Use the named file as the alias file. If no file is specified, use aliases in the current directory.

a
If set, wait for an "O:O" entry to exist in the alias database before starting up. If it does not appear in five minutes, rebuild the database.

c
If an outgoing mailer is marked as being expensive, don't connect immediately. This requires that queueing be compiled in, since it will depend on a queue run process to actually send the mail.

dx
Deliver in mode x. Legal modes are:

   i  Deliver interactively (synchronously)
   b  Deliver in background (asynchronously)
   q  Just queue the message (deliver during queue run)

D
If set, rebuild the alias database if necessary and possible. If this option is not set, sendmail will never rebuild the alias database unless explicitly requested using -bl.

ez
Dispose of errors using mode x. The values for x are:

   p  Print error messages (default)
   q  No messages, just give exit status
   m  Mail back errors
   w  Write back errors (mail if user not logged in)
   e  Mail back errors and give zero exit stat always

Fn
The temporary file mode, in octal. 644 and 600 are good choices.

f
Save Unix-style "From" lines at the front of headers. Normally they are assumed redundant and discarded.

gn
Set the default group id for mailers to run in to n.

Hfile
Specify the help file for SMTP.

i
Ignore dots in incoming messages.

Ln
Set the default log level to n.

Mxvalue
Set the macro x to value. This is intended only for use from the command line.

m
Send to me too, even if I am in an alias expansion.

o
Assume that the headers may be in old format, i.e., spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

Qdir
Use the named dir as the queue directory.

rtime
Timeout reads after time interval.
Sendmail Installation and Operation Guide

S|file | Log statistics in the named file.
s | Be super-safe when running things, i.e., always instantiate the queue file, even if you are going to attempt immediate delivery. Sendmail always instantiates the queue file before returning control to the client under any circumstances.

T|ime | Set the queue timeout to time. After this interval, messages that have not been successfully sent will be returned to the sender.

T|ime |

t|ime |

S|u|D | Set the local timezone name to S for standard time and D for daylight time; this is only used under version six.

un | Set the default userid for mailers to n. Mailers without the S flag in the mailer definition will run as this user.

v | Run in verbose mode.
APPENDIX C

MAILER FLAGS

The following flags may be set in the mailer description.

f The mailer wants a -f from flag, but only if this is a network forward operation (i.e., the mailer will give an error if the executing user does not have special permissions).

r Same as f, but sends a -r flag.

S Don't reset the userid before calling the mailer. This would be used in a secure environment where sendmail ran as root. This could be used to avoid forged addresses. This flag is suppressed if given from an "unsafe" environment (e.g., a user's mail.cf file).

n Do not insert a UNIX-style "From" line on the front of the message.

I This mailer is local (i.e., final delivery will be performed).

s Strip quote characters off of the address before calling the mailer.

m This mailer can send to multiple users on the same host in one transaction. When a $u macro occurs in the argv part of the mailer definition, that field will be repeated as necessary for all qualifying users.

F This mailer wants a "From:" header line.

D This mailer wants a "Date:" header line.

M This mailer wants a "Message-Id:" header line.

x This mailer wants a "Full-Name:" header line.

P This mailer wants a "Return-Path:" line.

u Upper case should be preserved in user names for this mailer.

h Upper case should be preserved in host names for this mailer.

A This is an Arpanet-compatible mailer, and all appropriate modes should be set.

U This mailer wants Unix-style "From" lines with the ugly UUCP-style "remote from <host>" on the end.

E This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.

X This mailer wants to use the hidden dot algorithm as specified in RFC821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This insures that lines in the message containing a dot will not terminate the message prematurely.

L Limit the line lengths as specified in RFC821.

P Use the return-path in the SMTP "MAIL FROM:" command rather than just the return address; although this is required in RFC821, many hosts do not process return paths properly.

I This mailer will be speaking SMTP to another sendmail – as such it can use special protocol features. This option is not required (i.e., if this option is omitted the transmission will still operate successfully, although perhaps not as efficiently as possible).

C If mail is received from a mailer with this flag set, any addresses in the header that do not have an at sign ("@") after being rewritten by ruleset three will have the "@domain" clause from the sender tacked on. This allows mail with headers of the form:
From: usera@hosta
To: userb@hostb, userc

to be rewritten as:
From: usera@hosta
To: userb@hostb, userc@hosta
automatically.
This is a summary of the support files that sendmail creates or generates.

/usr/lib/sendmail
   The binary of sendmail.

/usr/bin/newaliases
   A link to /usr/lib/sendmail; causes the alias database to be rebuilt. Running this program is completely equivalent to giving sendmail the -bl flag.

/usr/lib/sendmail.cf
   The configuration file, in textual form.

/usr/lib/sendmail.fc
   The configuration file represented as a memory image.

/usr/lib/sendmail.hf
   The SMTP help file.

/usr/lib/sendmail.st
   A statistics file; need not be present.

/usr/lib/aliases
   The textual version of the alias file.

/usr/lib/aliases.{pag,dir}
   The alias file in dbm(3) format.

/usr/etc/in.syslog
   The program to do logging.

/etc/syslog.conf
   The configuration file for syslog.

/etc/syslog.pid
   Contains the process id of the currently running syslog.

/usr/spool/mqueue
   The directory in which the mail queue and temporary files reside.

/usr/spool/mqueue/qf
   Control (queue) files for messages.

/usr/spool/mqueue/df
   Data files.

/usr/spool/mqueue/lf
   Lock files.

/usr/spool/mqueue/tf
   Temporary versions of the qf files, used during queue file rebuild.

/usr/spool/mqueue/af
   A file used when creating a unique id.

/usr/spool/mqueue/xf
   A transcript of the current session.
## UUCP Implementation Description

### Table of Contents

1. Introduction ..................................................................................................................................... 1
2. UUCP UNIX to UNIX File Copy ................................................................................................. 1  
   2.1. Type 1 Local Copy ........................................................................................................... 3  
   2.2. Type 2 Receive Files ........................................................................................................ 3  
   2.3. Type 3 Send Files .............................................................................................................. 3  
   2.4. Type 4 and Type 5 Remote UUCP Required ..................................................................... 3  
   2.5. Type 6 Remote Execution ............................................................................................ 4  
3. UUX UNIX to UNIX Execution ................................................................................................. 4  
   3.1. User Line ................................................................................................................................. 5  
   3.2. Required File Line ............................................................................................................. 5  
   3.3. Standard Input Line ........................................................................................................ 5  
   3.4. Standard Output Line .................................................................................................... 5  
   3.5. Command Line .................................................................................................................... 6  
4. UUXQT UUCP Command Execution .............................................................................. 6  
5. UUCICO Copy In, Copy Out ................................................................................................... 6  
   5.1. Scan For Work .................................................................................................................... 7  
   5.2. Call Remote System ........................................................................................................ 7  
   5.3. Line Protocol Selection ................................................................................................... 8  
   5.4. Work Processing ............................................................................................................ 8  
   5.5. Conversation Termination ............................................................................................... 9  
6. UULOG UUCP Log Inquiry .................................................................................................. 9  
7. UUCLEAN UUCP Spool Directory Cleanup ...................................................................... 10  
8. Security ........................................................................................................................................... 10  
9. UUCP Installation ........................................................................................................................ 11  
   9.1. uucp.h Modification ......................................................................................................... 11  
   9.2. Makefile Modification ...................................................................................................... 12
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3</td>
<td>Compiling the System</td>
<td>12</td>
</tr>
<tr>
<td>9.4</td>
<td>Files Required for Execution</td>
<td>12</td>
</tr>
<tr>
<td>9.5</td>
<td>L-devices Call Unit Information File</td>
<td>13</td>
</tr>
<tr>
<td>9.6</td>
<td>L-dialcodes Dial Code Abbreviations File</td>
<td>13</td>
</tr>
<tr>
<td>9.7</td>
<td>USERFILE</td>
<td>13</td>
</tr>
<tr>
<td>9.8</td>
<td>L.sys</td>
<td>15</td>
</tr>
<tr>
<td>9.9</td>
<td>L.cmds</td>
<td>16</td>
</tr>
<tr>
<td>9.10</td>
<td>Device Types</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>Administration</td>
<td>17</td>
</tr>
<tr>
<td>10.1</td>
<td>SQFILE Sequence Check File</td>
<td>17</td>
</tr>
<tr>
<td>10.2</td>
<td>TM Temporary Data Files</td>
<td>18</td>
</tr>
<tr>
<td>10.3</td>
<td>LOG Log Entry Files</td>
<td>18</td>
</tr>
<tr>
<td>10.4</td>
<td>STST System Status Files</td>
<td>18</td>
</tr>
<tr>
<td>10.5</td>
<td>LCK Lock Files</td>
<td>19</td>
</tr>
<tr>
<td>10.6</td>
<td>Shell Files</td>
<td>19</td>
</tr>
<tr>
<td>10.7</td>
<td>Login Entry</td>
<td>19</td>
</tr>
<tr>
<td>10.8</td>
<td>File Modes</td>
<td>20</td>
</tr>
</tbody>
</table>
1. Introduction

Uucp is a series of programs designed such that UNIX† systems can communicate with each other using either dial-up or hardwired communication lines. Uucp transfers files between UNIX systems, and can also run commands on remote machines. The first version of the system was designed and implemented by M. E. Lesk. This paper describes the current (second) implementation of the system. This document gives a detailed description of the current implementation of uucp. It is designed for use by an administrator or installer of the system. It is not meant as a user's guide.

Uucp is a batch operation. Files are created in a spool directory for processing by the uucp daemons. There are three types of files used for the execution of work: Data files contain data for transfer to remote systems; Work files contain directions for file transfers between systems; Execute files are scripts for UNIX commands that involve the resources of one or more systems.

There are four primary programs involved in uucp's operation:

- **uucp** builds work files and gathers data files in the spool directory for data transmission.
- **uuze** creates work files and execute files, and gathers data files for the remote execution of UNIX commands.
- **uucico** executes the work files for data transmission.
- **uuqt** executes the scripts for UNIX command execution.

There are two administrative programs:

- **uulog** gathers temporary log files that may occur due to lockout of the uucp log file and reports some information such as copy requests and completion status.
- **uuclean** removes old files from the spool directory.

The remaining sections of this manual describes the operation of each program, security, installation details, files required for execution, and administration of the uucp system.

2. UUCP – UNIX to UNIX File Copy

The uucp command was is designed to look like cp(1) to the user. The syntax is:

```
uucp [option] ... source ... destination
```

where the source and destination may contain the prefix system-name!, which indicates the system where the file or files reside or where they will be copied.

Uucp has several options:

- **-d** Make directories when necessary for copying the file.
- **-c** Don't copy source files to the spool directory, but use the specified source when the actual transfer takes place. Note that the files, and all directories leading to them, must be accessible by everybody.
- **-esys** Send this job to system sys to execute. Note that this only works when the system sys allows uuqt to execute a uucp command. See the sections entitled Uuqt and Security.

† UNIX is a trademark of Bell Laboratories.

7 January 1984
-C  Force the source files to be copied to the spool directory.
-f   Do not make directories.
-g/letter  Put letter in as the grade in the name of the work file. This can be used to change the order of work for a particular machine.
-m   Send mail to the requester on completion of the work.
-n/user  Notify user on the remote machine that a file has been sent.

Then there are options available for debugging:
-sdir  Use dir as the spool directory.
-r   Queue the job but do not start uucico program.
-xnum  Num is a level number between 1 and 9; higher numbers give more debugging output.

The destination may be a directory name, in which case the file name is taken from the last part of the source's name. If the directory exists, it must be writable by everybody. Note that if the destination is a directory name and the -d option is specified to create the directory, the directory name must be followed by “/”. The source name can contain special shell characters such as ?, *, and [ and ]. These are expanded on the appropriate system.

The command

    uucp *.c usg!/usr/dan

sets up the transfer of all files whose names end with .c to the /usr/dan directory on the usg machine.

The source and/or destination names may also contain a ~user prefix. This translates to the login directory of user on the specified system. File names beginning with “~/” translate into the public directory (usually /usr/spool/uucppublic) on the remote system. For names with partial path-names, the current directory is prepended to the file name. File names beginning with ../ are not permitted for security reasons.

The command

    uucp usg!~dan/*.h ~dan

sets up the transfer of files whose names end with .h in dan's login directory on system usg to dan's local login directory.

For each source file, uucp checks the source and destination file-names, the system-part of each argument, and the options to classify the work into several types:

[1]  Copy source to destination on local system.
[2]  Receive files from other systems.
[3]  Send files to a remote system.
[4]  Send files from remote systems to another remote system.
[5]  Receive files from remote systems when the source contains special shell characters as mentioned above.
[6]  Request that the uucp command be executed by a remote system.

After the work has been set up in the spool directory, the uucico program is started to try to contact the other machine and execute the work (unless the -r option was specified).
2.1. Type 1 – Local Copy

The copy is done locally. The \(-m\) and \(-n\) options are not honored in this case.

2.2. Type 2 – Receive Files

A work file is created or appended with a one line entry for each request. The upper limit to the number of files per work file is set in uuep.h. The default setting is 20. After the limit has been reached, a new work file is created.

All work files and execute files use a blank as the field separator. The fields for these entries are given below.

[1] R
[2] The full path-name of the source or a `something/path-name`. The `something` part is expanded on the remote system.
[3] The full path-name of the destination file. If the `something` notation is used, it is immediately expanded.
[5] A "-" followed by an option list. The options \(-m\) and \(-d\) may appear.

2.3. Type 3 – Send Files

Each source file is copied into a data file in the spool directory. (A \(-c\) option on the uucp command prevents the data file from being made. In this case, the file is transmitted from the indicated source.) The fields for these entries are given below.

[1] S
[3] The full-path name of the destination or `something/file-name`.
[5] A "-" followed by an option list. The options \(-d\), \(-m\), and \(-n\) may appear.
[6] The name of the data file in the spool directory. A dummy name, "D.0" is used when the \(-c\) option is specified.
[8] The user on the remote system who should be notified upon completion of the file copy when the \(-n\) option is specified.

2.4. Type 4 and Type 5 – Remote UUCP Required

Uucp generates a uucp command and sends it to the remote machine; the remote uucico executes the uucp command.
2.5. Type 6 – Remote Execution

Remote execution occurs when the -e option is used. In this case, the uux facility is used to create and send the request. This requires that the remote uuzqt program allows the uucp command to be executed.

3. UUX – UNIX to UNIX Execution

The uuz command sets up the execution of a UNIX command where the execution machine and/or some of the files are remote. The syntax of the uuz command is

\[
uuz [-] \text{[option]} \ldots \text{command-string}
\]

where the command-string is made up of one or more arguments. All special shell characters such as <, >, |, and *, must be quoted either by quoting the entire command-string or quoting the special character as a separate argument.

Within the command-string, the command and file names may contain a system-name/ prefix. Arguments which do not contain a ! character are assumed to be part of the command string and are not treated as files (that is, uuz does not copy them to the execution machine).

An argument containing a ! but should not be treated as a file at the present time, can be escaped by surrounding the argument in parentheses ( and ). Note that the ( and ) characters themselves must usually be escaped with a \ character.

The "-" indicates that the standard input for command-string should be inherited from the standard input of the uuz command.

The following options are available for uuz:

- Read from the standard input.
-x Read from the standard input.
-n Do not notify the requestor (by mail) of completion status.
-z Only notify the requestor (by mail) of non-zero completion status.

The following options are available for debugging:

-r Don't start uucico or uuzqt after queuing the job.
-x num Num is a level number between 1 and 9; higher numbers give more debugging output.

The command:

\[
\text{pr abc | uux - usg!lpr}
\]

sets up the output of the

\[
\text{pr abc}
\]

command as standard input to an lpr command to be executed on system usg.

Uux generates an execute file containing the names of the files required for execution (including standard input), the user's login name, the destination of the standard output, and the command to be executed. This execute file file is either put in the spool directory for local execution or sent to the remote system using a send command (uucp type 3 command, described previously).

For required files that are not on the execution machine, uux generates receive command files (uucp type 2 command, described previously). The uucico program puts these command-files on
the execution machine for execution.

The *execute file* contains a script that is processed by the `uuzqt` program. It is made up of several lines, each of which contains an identification character and one or more arguments. The lines are described in the subsections below. Here is a summary of the types of lines that appear in the file. They are described in detail in the sections following.

- **User Line** Identifies the requestor's login name and system.
- **File Line** Identifies a filename for transmission.
- **Standard Input Line**
  Specifies a standard input file.
- **Standard Output Line**
  Specifies a standard output file.
- **Command Line**
  Identifies a UNIX system command for `uuzqt` to execute.

### 3.1. User Line

```
U user system
```

where the *user* and *system* are the requester's login name and system.

### 3.2. Required File Line

```
F file-name real-name
```

where *file-name* is a unique name used for file transmission and *real-name* is the last part of the actual file name (contains no path information).

Zero or more of these lines may be present. The `uuzqt` program checks for the existence of all these files before the command is executed.

### 3.3. Standard Input Line

```
I file-name
```

The standard input is either specified by a `<` in the command-string or inherited from the standard input of the `uux` command if the "-" option is used. If a standard input is not specified, `/dev/null` is used. Note that if there is a standard input specified, it will also appear in an "F" line.

### 3.4. Standard Output Line

```
O file-name system-name
```

The standard output is specified by a `>` within the command-string. If a standard output is not specified, `/dev/null` is used. Note that the appending to a file by using `>>` is not implemented.
3.5. Command Line

C command [arguments] ...

The arguments are those specified in the command-string. The standard input and standard output will not appear on this line. All required files are moved to the execution directory (usually /usr/lib/uucp/XQTDIR) and the UNIX command is executed using the shell specified in the uucp.h header file. In addition, a shell "PATH" statement is prepended to the command line as specified in the uuzzqt program.

Note that a check is made to see that the command is allowed as specified in the uuzzqt program. After execution, the standard output is copied or sent to the proper place.

4. UUXQT – UUCP Command Execution

The uuzzqt program executes scripts generated by uuz.

The uuzzqt program may be started by either the uucico or uuz programs or a daemon specified by a crontab entry. Uuzzqt scans the spool directory for execute files (prefix "X."). Each execute file is checked to see if all the required files are available and if so, the command line is verified and executed.

The execute file is described in the section entitled uuz, above.

The execution is accomplished by executing a

    sh -c

of the command line after appropriate standard input and standard output have been opened.

If a standard output is specified, the program creates a send command, or copies the output file as appropriate.

Uuzzqt accepts the standard debugging option:

    -x num   Num is a level number between 1 and 9; higher numbers give more debug output.

5. UUCICO – Copy In, Copy Out

The uucico program performs several major functions:

- Scan the spool directory for work.
- Place a call to a remote system.
- Negotiate a line protocol to be used.
- Execute all requests from both systems.
- Log work requests and work completions.

Uucico may be started in several ways:

a. by a system daemon specified in a crontab entry,
b. by one of the uucp, uuz, uuzzqt or uucico programs,
c. directly by the user (this is usually for testing),
d. by a remote system. The uucico program should be specified as the "shell" field in the /etc/passwd file for the logins used by remote systems to access uucp.
When started by method a, b or c, uucico is considered to be in *MASTER* mode. In this mode, a connection is made to a remote system. If started by a remote system (method d), uucico is considered to be in *SLAVE* mode.

The *MASTER* mode operates in one of two ways. If no system name is specified (–s option not specified) uucico scans the spool directory for systems to call. If a system name is specified, that system is called, and work is only done for that system.

*Uucico* is generally started by another program. There are several options used for execution:

- `–rl` Start uucico in *MASTER* mode. This is used when *uucico* is started by a program or “cron” shell.
- `–s sys` Do work only for system *sys*. If –s is specified, a call to the specified system is made even if there is no work for system *sys* in the spool directory. This is useful for polling systems that do not have the hardware to initiate a connection.

The following options are used primarily for debugging:

- `–dir dir` Use directory *dir* for the spool directory.
- `–x num` *Num* is a level number between 1 and 9; higher numbers give more debugging output.

The next part of this section describes the major steps within the *uucico* program.

### 5.1. Scan For Work

The names of the work related files in the spool directory have format

```
type . system-name grade number
```

- `type` is an upper case letter (C – copy command file, D – data file, X – execute file),
- `system-name` is the remote system, *truncated to seven characters*.
- `grade` is a character,
- `number` is a four digit, hexadecimal, zero padded sequence number.

The file

```
C.res45n0031
```

would be a *work file* for a file transfer between the local machine and the “res45” machine.

The scan for work is done by looking through the spool directory for *work files* (files with prefix “C.”). A list is made of all systems to be called. *Uucico* then calls each system and processes all *work files*.

### 5.2. Call Remote System

The call is made using information from several files that reside in the *uucp* program directory (usually `/usr/lib/uucp`). At the start of the call process, a lock is set to prevent multiple conversations between the same two systems.

The *L.sys* file contains information required to make the remote connection:

[1] system name,
[2] times to call the system (days-of-week and times-of-day) and the minimum time delay before retry,
[3] device or device type to be used for call,
[4] line class (this is the line speed on almost all systems),
[5] phone number if field [3] is ACU or the device if not ACU,
[6] login information (zero or more fields),

The time field is checked against the present time to see if the call should be made. The phone number may contain abbreviations (for example, mh, py, boston) that get translated into dial sequences using the L-dialcodes file.

The L-devices file is scanned using fields [3] and [4] from the L.sys file to find an available device for the call. The program tries each devices that satisfy [3] and [4] until a call is made, or no more devices can be tried. If a device is successfully opened, a lock file is created. If the call is completed, the login information (field [6] of L.sys) is used to login.

The conversation between the two uucico programs begins with a handshake started by the called (SLAVE) system. The SLAVE sends a message to let the MASTER know it is ready to receive the system identification and conversation sequence number. The SLAVE verifies the response from the MASTER and if acceptable, protocol selection begins. The SLAVE can also reply with a "call-back required" message, in which case the current conversation is terminated.

5.3. Line Protocol Selection

The remote system sends a message:

\[ \text{Pproto-list} \]

where proto-list is a string of characters, each representing a line protocol.

The calling program checks proto-list for a letter corresponding to an available line protocol and returns a use-protocol message. The use-protocol message is

\[ \text{Ucode} \]

where code is either a one character protocol letter or "N", which means there is no common protocol. The only protocol which is currently implemented is "g", which uses the packet driver.

5.4. Work Processing

The MASTER program does a work search similar to the one used in the Scan For Work section described above (the MASTER has been specified by the "-rI" uucico option). Each message used during the work processing is specified by the first character of the message:

\[ \text{S send a file,} \]
\[ \text{R receive a file,} \]
\[ \text{C copy complete,} \]
\[ \text{X execute a uucp command,} \]
\[ \text{H hangup.} \]

The MASTER sends R, S or X messages until all work for the remote system is complete, at which point an H message is sent. The SLAVE replies with SY, SN, RY, RN, HY, HN, XY, or
XN, corresponding to yes or no for each request.

An N response can be followed by a number giving the reason for the failure:

- **N0**
  - Copy failed (reason not given by remote system).

- **N1**
  - Local access to file denied.

- **N2**
  - Remote access to path or file denied.

- **N3**
  - System error – bad uucp command generated.

- **N4**
  - Remote system cannot create temporary file.

- **N5**
  - Cannot copy to file or directory – file left in `pubdir/user/file`.

- **N6**
  - Cannot copy to file or directory – file left in `pubdir/user/file`.

The send and receive replies are based on permission to access the requested file or directory using the `USERFILE` and read/write permissions of the file or directory.

After each file is copied into the spool directory of the receiving system, a copy-complete message is sent by the receiver of the file. The message `CY` is sent if the file has successfully been moved from the spool directory to the destination. If the file was not successfully moved from the spool directory to the destination, a `CN` message is sent, the file is put in the public directory (usually `/usr/spool/uucppublic`), and the requester is notified by mail.

The requests and results are logged on both systems.

The hangup response is determined by a work scan of the SLAVE’s spool directory. If work for the remote system exists an `HN` message is sent and the programs switch roles. If no work exists, an `HY` response is sent.

### 5.5. Conversation Termination

When the MASTER receives a `HY` message, the MASTER echoes the message back to the SLAVE and the protocols are turned off. Each program sends a final “OO” (Over and Out) message to the other. The original SLAVE program cleans up and terminate. The MASTER then proceeds to call other systems unless a “-s” option was specified.

### 6. UULOG – UUCP Log Inquiry

When a `uucp` program can not make a log entry directly into the `LOGFILE` an individual log file is created: a file with prefix `LOG`. This sometimes occurs when more than one `uucp` process is running. Periodically, `uulog` may be executed to append these files to the `LOGFILE`.

The `uulog` program may also be used to request the output of `LOGFILE` entries. The request is specified by the use of the options:

- `-sys sys` Print entries where `sys` is the remote system name,
- `-user user` Print entries for user `user`.
The intersection of lines satisfying the two options is output. A null system names or users respectively.

Debugging options available are:

- `xnum`        Num is a level number between 1 and 9; higher numbers give more debug output.
- `nsecs`       Time out lock on log file if older than `secs` seconds.

7. **UUCLEAN – UUCP Spool Directory Cleanup**

_Uuclean_ is typically started by the _uucp_ daily daemon. Its function is to remove files from the spool directory that are more than three days old. These are usually files for work that cannot be completed. The requester of this work is notified that the files have been deleted.

There are several options:

- `ddir` The directory to be scanned is `dir`.
- `m` Send mail to the owner of each file being removed. Note that most files put into the spool directory are owned by the owner of the _uucp_ programs since the setuid bit will be set on these programs. This mail is sometimes useful for administration.
- `nhours` Change the aging time from 72 hours to `hours` hours.
- `ppre` Examine files with prefix `pre` for deletion. Up to 10 of these options may be specified.
- `xnum` This is the level of debugging output desired.

8. **Security**

> The uucp system, left unrestricted, will let any outside user execute any commands and copy out/in any file that is readable/writable by a uucp login user. It is up to the individual sites to be aware of this and apply the protections that they feel are necessary.

There are several security features available aside from the normal file mode protections. These must be set up by the administrator of the _uucp_ system.

- The login for _uucp_ does not get a standard shell. Instead, the _uucico_ program is started so that all work is done through _uucico_.
- The owner of the _uucp_ programs should be an administrative login. It should not be one of the logins used for remote system access to _uucp_.
- All _uucp_ logins should have passwords.
- A path check is done on file names that are to be sent or received. The _USERFILE_ supplies the information for these checks. The _USERFILE_ can also be set up to require call-back for certain login-ids. See the "Files Required For Execution" section for the file description.
- A conversation sequence count can be set up so that the called system can be more confident of the caller's identity.
- The _uuzqt_ program reads a file containing a list of commands that it will execute. A "PATH" shell statement is prepended to the command line as specified in the _uuzqt_ program. The installer may modify the list or remove the restrictions as desired.
The L.sys file should be owned by the uucp administrative login and have mode 0400 to protect the phone numbers and login information for remote sites.

The programs uucp, uucico, uux, uzq, uulofl, and uuclean should be owned by the uucp administrative login, have the setuid bit set, and have only execute permissions.

9. UUCP Installation

It is assumed that the login name used by a remote computer to call into a local computer is not the same as the login name of a normal user or the uucp administrative login. However, several remote computers may use the same login name. It is suggested that the installer follow the convention of using the letter “U” followed by the system name as the login name for each system. For example, use login name Uuseg for the useg system.

Each computer should be given a unique system name that is transmitted at the start of each call. This name identifies the calling machine to the called machine. The login/system names are used for security as described later in the USERFILE section.

There are several source modifications that may be required before the system programs are compiled. These relate to the directories, local system name, and attributes of the local environment.

The uucp system uses several directories:

- sources (/usr/src/cmd/uucp) – This directory contains the uucp system source files.
- program (/usr/lib/uucp) – This is the directory used for some of the executable system programs and the system files. Some of the programs reside in /usr/bin.
- spool (/usr/spool/uucp) – This is the uucp system spool directory.
- xqtdir (/usr/lib/uucp/XQTDIR) – This directory is used during execution of the uux scripts.

The names in parentheses above are the default values for the directories. The italicized names sources, program, xqtdir, and spool are used in the following text to represent the appropriate directory names.

There are two files that may require modification, namely the Makefile file and the uucp.h file. In addition, the uuzqt.c program may be modified as indicated in the section entitled Security, above. The following sections describe the modifications.

9.1. uucp.h Modification

Several manifests in uucp.h may need modification for the local system environment:

- UNAME should be defined if the “uname” function is available.
- ACULAST is the character required by the ACU as the last character. For most systems, it is a “-”. This is only relevant if you are using a DN11 autodialler.
- DIALOUT should be defined if the C library routine “dialout” is available.
- UUDIR should be defined if the uucp subdirectory syskludge is being used.
- UUNAME should be defined if the system name should be read from the SYSTEMNAME file.
- UUSTAT should be defined if you need the uustat program.
9.2. Makefile Modification

There are several make variable definitions that may need modification:

- **INSDIR** is the *program* directory (for example, INSDIR=/usr/lib/uucp). This parameter is used if "make cp" or "make install" is used.
- **PUBDIR** is a public directory for remote access. This is also the login directory for remote uucp users. It should be the same as that defined in uucp.h.
- **SPOOL** is the uucp spool directory. This should be the same as that defined in uucp.h.
- **XQTDIR** is the directory for uuxqt to use for command execution. It is also defined in uucp.h.
- **OWNER** is the administrative login for uucp.
- **LIBS** should include syskludge/syskludge.a if the syskludge library is used. **UUDIR** should be defined in uucp.h.
- **CFLAGS** add -DVMUNIX if on a VMUNIX system.

9.3. Compiling the System

The command:

```
make install
```

makes the required directories, compiles all programs, sets the proper file modes, and copies the programs to the proper directories. This command should be run as root. The command:

```
make
```

compiles the entire system.

The programs uucp, uuz, and uulog should be put in /usr/bin. The programs uuxqt, uucico, and uuclean should be put in the *program* directory.

9.4. Files Required for Execution

Six files are required for execution. They should reside in the *program* directory. The field separator for all files is a space. The required files are summarized here, and the following subsections describe them in detail.

- **L-device** contains call unit information.
- **L-dialcodes** contains dialcode abbreviations.
- **USERFILE** contains user accessibility and constraint information.
- **L.sys** contains information about the systems which local uucp programs can call.
- **L.cmds** contains commands which uuxqt is allowed to execute.
- **SYSTEMNAME** contains the name of the system.
9.5. L-devices – Call Unit Information File

$L$-devices contains call-unit device and hardwired connection information. The special device files are assumed to be in the /dev directory. The format for each entry in the $L$-devices file is:

type line call-unit speed

$\text{type}$ is a device type such as ACU or DIR. The currently supported device types are described later. The field can also be used to specify particular ACUs for some calls by using a suffix on the ACU field (ACUDF03wats, for instance). This name should be used in $L\_sys$.

$\text{line}$ is the device for the line (for example, cul0 if using a DN11, otherwise it is the same as the call-unit field).

$\text{call-unit}$ is the automatic call unit associated with line (for example, cua0). Hardwired lines have a number "0" in this field.

$\text{speed}$ is the line speed.

For example, an entry in the $L$-devices file like this:

ACU cul0 cua0 300

would be set up for a system that has a DN11 device "/dev/cul0" wired to a call-unit "/dev/cua0" for use at 300 baud. An entry like:

ACUDF03 cua0 cua0 1200

would be set up for a system that has a DF03 device "/dev/cua0" for use at 1200 baud.


$L$-dialcodes contains the dialcode abbreviations used in the $L\_sys$ file (for example, py, mh, boston). The entry format is:

\textit{abb} dial-seq

\textit{abb} is the abbreviation,

\textit{dial-seq} is the dial sequence to call that location.

For example, a line in the $L$-dialcodes file that looks like:

py 165-

would be set up so that entry py7777 would send 165-7777 to the dial-unit.

9.7. USERFILE

USERFILE contains user accessibility information. It specifies four types of constraint:

[1] which files can be accessed by a normal user of the local machine,

[2] which files can be accessed from a remote computer,
[3] which login name is used by a particular remote computer,
[4] whether a remote computer should be called back in order to confirm its identity.

Each line in USERFILE has the format:

```
login,sys [c] path-name [path-name] ...
```

- `login` is the login name for a user or the remote computer,
- `sys` is the system name for a remote computer,
- `c` is the optional call-back required flag,
- `path-name` is a path-name prefix that is acceptable for `sys`.

The constraints are implemented as follows.

[1] When the program is obeying a command stored on the local machine (MASTER mode) the path-names allowed are those given on the first line in the USERFILE that has the login name of the user who entered the command. If no such line is found, the first line with a null login name is used.

[2] When the program is responding to a command from a remote machine (SLAVE mode) the path-names allowed are those given on the first line in the file that has the system name that matches the remote machine. If no such line is found, the first one with a null system name is used.

[3] When a remote computer logs in, the login name that it uses must appear in the USERFILE. There may be several lines with the same login name but one of them must either have the name of the remote system or must contain a null system name.

[4] If the line matched in ([3]) contains a “c”, the remote machine is called back before any transactions take place.

**Examples**

The line:

```
  u,m /usr/xyz
```

allows machine `m` to login with name `u` and request the transfer of files whose names start with `/usr/xyz`.

The line:

```
  dan, /usr/dan
```

allows the ordinary user `dan` to issue commands for files whose name starts with `/usr/dan`.

(Note that this type of restriction is seldom used.)

The lines:

```
  u,m /usr/xyz /usr/spool
  u, /usr/spool
```

allows any remote machine to login with name `u`. If its system name is not `m`, it can only ask to transfer files whose names start with `/usr/spool`. If it is system `m`, it can send files from path `/usr/xyz` as well as `/usr/spool`.

The lines:
allow any user to transfer files beginning with /usr, but the user with login root can transfer any file. Note that any file that is to be transferred must be readable by anybody. The USER-FILE is normally set up as follows:

,myname /
 ,/usr/spool/uucppublic

where myname is the name of the current system. These lines allow any user on the current machine to access any file (subject to the normal permissions on the file) for uucp transfer, whereas users on other machines can only access files in /usr/spool/uucppublic.

9.8. L.sys

Each entry in L.sys represents one system that can be called by the local uucp programs. More than one line may be present for a particular system. In this case, the additional lines represent alternative communication paths that are tried in sequential order. The fields are described below.

system name
   The name of the remote system.

time
   This is a string that indicates the days-of-week and times-of-day when the system should be called (for example, MoTuTh0800-1730).
   The day portion may be a list containing some of
      Su Mo Tu We Th Fr Sa
   or it may be Wk for any week-day or Any for any day.
   The time should be a range of times (for example, 0800-1230). If no time portion is specified, any time of day is assumed to be okay for the call. Note that a time range that spans 0000 is permitted, for example, 0800-0600 means all times are ok other than times between 6 and 8 am.
   A time specification of "None" is often used for a passive system, that is, one which cannot call the specified system. In this case the following fields may be omitted.
   Note that the string "None" has no special significance, but is merely a string that is not a correct time specification.
   Several day-time specifications may be present, separated by a | character.
   An optional subfield is available to indicate the minimum time (minutes) before a retry following a failed attempt. The subfield separator is a ",". For example, Any,9 means call any time but don't allow another call until at least 9 minutes after a failure has occurred.

device
   This fields either starts with ACU, or is the hardwired device to be used for the call.
   For the hardwired case, the last part of the special file name is used (tty0, for instance).
class

This is usually the line speed for the call (for example, 300). The exception is when the C library routine "dialout" is available in which case this is the dialout class.

phone

The phone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation should be one that appears in the L-dialcodes file (for example, mh5900, boston995-9980). For the hardwired devices, this field contains the same string as used for the device field.

login

The login information is given as a series of fields and subfields in the format

[ expect send ] …

where expect is the string expected to be read and send is the string to be 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:

login--login

expects to see the word login; if it gets it, the program proceeds to the next field; if it does not get login, it sends null followed by a new line, then expects login again.

There are two special names available to be sent during the login sequence. The string EOT sends an EOT character and the string BREAK tries to send a BREAK character. The BREAK character is simulated using line speed changes and null characters and may not work on all devices and/or systems. A number from 1 to 9 may follow the BREAK. For example, BREAK1 sends 1 null character instead of the default of 3. Note that BREAK1 usually works best for 300/1200 baud lines.

The following escape sequences are also recognized:

\r        send a carriage-return.
\n        send a newline (linefeed) character.
\d        delay for 1 second.
\c        suppress newline at end of send string.
\s        send a space.

A typical entry in the L.sys file would be:

sys Any ACU 300 mh7654 login:-EOT-login:-BREAK-login: uucp ssword: word

The expect algorithm matches all or part of the input string as illustrated in the password field above. Very complex expect-send sequences are often required if the called system is connected to a terminal concentrator or a front end.

9.9. L.cmds

L.cmds contains a list of commands which uuzqt is allowed to execute. The commands should be one per line. At a minimum, L.cmds should contain the command "rmail". Other commands often allowed are "rnews", "usend", and "lpr".

16 7 January 1984
9.10. Device Types

The currently supported device types are:

<table>
<thead>
<tr>
<th>Type Code</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACU</td>
<td>DEC DN11</td>
</tr>
<tr>
<td>ACUDN11</td>
<td>DEC DN11</td>
</tr>
<tr>
<td>ACUDF02</td>
<td>DEC DF02 (300 baud only)</td>
</tr>
<tr>
<td>ACUDF03</td>
<td>DEC DF03 (1200 baud only)</td>
</tr>
<tr>
<td>ACUVENTEL</td>
<td>Ventel MD212 Plus</td>
</tr>
<tr>
<td>DIR</td>
<td>Hardwired Line</td>
</tr>
</tbody>
</table>

The DN11 is only available on DEC PDP-11 and VAX systems. The DEC DF02, DF03, and Ventel MD212 Plus can be connected to any system using a standard RS232 terminal interface. When connecting one of these devices to a terminal line, the device node (/dev/ttyz) should be renamed (to /dev/cuaO for instance) to emphasize that the line is for dialout only and to prevent accidentally starting a login process for that line.

The device type specified in the L-devices file should be one of those listed above, optionally qualified further by additional characters after the device type. The device type specified in the L.sys file should be a prefix of one of the devices specified in the L-devices file. For example, assume you have two DF03's, one connected to a local telephone line and the other connected to a WATS line. The L-devices file could be set up as follows:

```
ACUDF03local cuao cuao 1200
ACUDF03wats cuao cuao 1200
```

To call a system using only the WATS line, specify ACUDF03wats in the device type field. Similarly, to call a system using the local telephone line, specify ACUDF03local. To call a system using either DF03, specify ACUDF03 in the L.sys file.

Note that the telephone numbers specified in the L.sys file will have a format dependent on the ACU device type. This is a deficiency which may be corrected in the future.

10. Administration

This section indicates some events and files that must be administered for the uucp system. Some administration can be accomplished by shell files initiated by crontab entries. Others may require manual intervention. Some sample shell files are given toward the end of this section.

10.1. SQFILE – Sequence Check File

SQFILE is set up in the program directory and contains an entry for each remote system with which you agree to perform conversation sequence checks. The initial entry is just the system name of the remote system. The first conversation adds the conversation count and the date/time of the most resent conversation. These items are updated with each conversation. If a sequence check fails, the entry will have to be adjusted manually. Note that this feature is rarely used.
10.2. TM – Temporary Data Files

These files are created in the spool directory while a file is being copied from a remote machine. Their names have the form

TM.pid.ddd

where pid is a process-id and ddd is a sequential three digit number starting at zero. After the entire file is received, the TM file is moved/copied to the requested destination. If processing is abnormally terminated the file remains in the spool directory. The leftover files should be periodically removed; the uuclean program is useful in this regard. The command

program/uuclean -pTM

removes all TM files older than three days.

10.3. LOG – Log Entry Files

During execution, log information is appended to the LOGFILE. If the LOGFILE is locked by another process, the log information is placed in individual log files with a with a LOG prefix. These individual files should be combined into the LOGFILE by using the uulog program. Uulog appends the contents of the individual log files onto the end of the LOGFILE. The command:

uulog

accomplishes the merge. Options are available to print some or all the log entries after the files are merged. The LOGFILE should be removed periodically.

The LOG files are created initially with mode 0222. If the program that creates the file terminates normally, it changes the mode to 0666. Aborted runs may leave the files with mode 0222 and the uulog program will not read or remove them. To remove them, either use rm, uuclean, or change the mode to 0666 and let uulog merge them into the LOGFILE.

10.4. STST – System Status Files

These files are created in the spool directory by the uucico program. They contain information such as login, dialup or sequence check failures or will contain a TALKING status when two machines are conversing. The form of the file name is

STST.sys

where sys is the remote system name, truncated to seven characters.

For ordinary failures, such as dialup or login, the file prevents repeated tries for about 55 minutes. This is the default time; it can be changed on an individual system basis by a subfield of the time field in the L.sys file. For sequence check failures, the file must be removed before any future attempts to converse with that remote system. Retries are accomplished by starting uucico from crontab, usually hourly.
10.5. LCK – Lock Files

Lock files are created for each device in use (e.g., automatic calling unit) and each system conversing. This prevents duplicate conversations and multiple attempts to use the same device. The form of the lock file name is:

LCK..str

where str is either a device or system name. The files may be left in the spool directory if runs abort (usually only on system crashes). They are ignored (reused) after 1.5 hours. When runs abort and calls are desired before the time limit, the lock files should be removed.

10.6. Shell Files

The uucp program spools work and attempts to start the uucico program, but uucico will not always be able to execute the request immediately. Therefore, the uucico program should be periodically started. The command to start uucico can be put in a “shell” file with a command to merge LOG. files and started by a crontab entry on an hourly basis. The file could contain the commands:

```
/usr/bin/uulog
program/uucico -rl -sinter
program/uucico -rl
```

The “-rl” option is required to start the uucico program in MASTER mode. The “-s” option can be used for polling as illustrated in the second line where machine inter is being polled. The third line processes all other spooled work.

Another shell file may be set up on a daily basis to remove TM, ST and LCK files and C. or D. files for work that can not be accomplished for reasons like bad phone number, login changes, and so on. A shell file containing commands like:

```
program/uuclean -pTM -pC. -pD.
program/uuclean -pST -pLCK -n12
```

can be used. Note that the “-n12” option causes the ST and LCK files older than 12 hours to be deleted. The absence of the “-n” option uses a three day time limit.

A daily or weekly shell should also be created to remove or save old LOGFILE’s. A shell like:

```
cp spool/LOGFILE spool/o.LOGFILE
rm spool/LOGFILE
```

can be used.

The shell files in program/uucp.* do a more extensive job than that described here. They should be started by entries in crontab. Read the shell files for more information.

10.7. Login Entry

Two or more logins should be set up for uucp. One should be an administrative login: the owner of all the uucp programs, directories and files. All others are used by remote systems to access the uucp system. Each of the /etc/passwd entries for the access logins should have program/uucico as the shell to be executed. The login directory should be the public directory (usually /usr/spool/uucppublic) for both the administrative login and the access logins. The
various access login names are used in the USERFILE to restrict file access.

10.8. File Modes

The programs uucp, uuz, uucico, uulog, uuclean, and uuzqt should be owned by the uucp administrative login with the "setuid" bit set and only execute permissions (mode 04111). The L.sqs, SQFILE, and the USERFILE, which are put in the program directory should be owned by the uucp administrative login and set with mode 0400. The mode of spool should be 0755. The mode of sqtdir should be 0777. The L-dialcodes and the L-devices files should have mode 0444.
USENET Installation and Maintenance

Table of Contents

USENET Installation and Maintenance ................................. 1
1. Files .............................................................................. 1
   1.1. File Modes and Permissions .................................... 2
   1.2. Format of the *sys* File ......................................... 3
2. Setting up Links ............................................................ 4
   2.1. Non-mail Links ..................................................... 4
   2.2. Mail Links ............................................................ 4
3. Posting Methods ............................................................ 5
4. Control Messages .......................................................... 5
5. Maintenance ................................................................. 6
6. Creating New Newsgroups ................................................ 7
7. Differences between Version 2.9 and 2.10 ....................... 7
8. Version A Versus Version B ............................................. 9
USENET Installation and Maintenance

This paper addresses a few (slightly disjunct) topics relevant to system administrators installing, converting, or maintaining a network news system. The basic installation and conversion procedures are discussed in the body of the System Installation and Maintenance Guide; here we give more detailed information on usage and maintenance.

1. Files

The files in /usr/lib/news and their functions are:

**active**  Lists active newsgroups. Each line of the *active* file contains two fields: the newsgroup name, and the highest local article number (for the most recently received article within the newsgroup). The fields are separated by a space. Local article numbers begin at 1 and increment as articles are received. They do not usually correspond to local article numbers at other sites. The article number is always stored as a 5 digit number (with leading zeros) to allow updating of the file in place. *Active* is automatically updated as new newsgroups come in.

The order of *active*’s list of newsgroups dictates the order in which news will be presented by *readnews*, so you might want to edit *active* and arrange the newsgroups accordingly. Here is a recommended order for *active*:

```
general
local.general
net.general
net.followup
local newsgroups, in alphabetical order
net.all newsgroups, in alphabetical order
junk
fa.all, in alphabetical order
test
all.test
to.all
control
```

**caesar**  Does caesar decoding of rotated text on a line-by-line basis. *Caesar* copies standard input to the standard output, rotating each line according to a static single letter frequency table. If an integer argument is given (13, for example, may be used to encode material for posting), every line is rotated by that argument, without regard for letter frequencies. Call up *caesar* by using the D *readnews* command.
help Displays a list of legitimate commands when an illegal command is typed to readnews.

history Lists every article received by your system. Used to reject articles that come in for the second time (presumably via a different path). The history file will grow, but may be cleaned out with the expire command.

history.dir, history.pag These two files are used as a hashed version of history; they contain the message ID's of all articles in the history file. Both files are updated by inews and expire.

log If present, maintains a log of articles processed and error conditions encountered. The log file can grow without limit, so it is cleaned out periodically by running news.week from cron.

ngfile Lists newsgroups that you can legally post to locally. Actually, it's a pattern, so if you include all it will allow everything. You probably want to forbid fa.all here, for example, since fa.groups are not posted to directly. There is also a mechanism for controlling which newsgroups you will accept from other sites — see the section on the format of the sys file, below.

notify Names the user to notify in case of a problem. If the file is empty, nobody is notified. The notify file is especially useful if one person administers several systems, and does not want multiple copies of control message notifications.

organization Contains (on one line) the name and address of your organization. This information is inserted in the articles you post.

recnews Arranges to send mail to post news.

recording Lists newsgroup classes and file names to display recordings for. Recordings on certain newsgroups are intended to remind the user of the rules for the newsgroup; in the case of a certain companies, recordings may be used to remind news authors to protect proprietary information.

Recording contains one line per recording. Each line contains two fields, separated by a space. The first field is the newsgroup class (net.all, for example); the second field is the name of the file containing the recorded message. If the file name does not begin with a slash, it is searched for in /usr/lib/news.

sendnews Sends news internally from one computer to another. Useful if you use mail links to transmit articles.

seq Contains the current sequence number for your system. Used to generate unique article ID's.

sys Lists all your neighbors, which newsgroups they subscribe to, and how to send news to them. Sys's format is documented below.

users Lists users who read news on your system.

uureq Receives news sent by sendnews.

1.1. File Modes and Permissions

The current intended state of affairs is that inews runs set-user-id (suid) news. The readnews program does not need to be set-user-id. This makes it possible to write your own interface to read news instead of using readnews.
All the files in /usr/spool/news should be writable by the “news” user.

1.2. Format of the sys File

To set up a link to another site (the two types of links are discussed below), you edit /usr/lib/news/sys. Sys is similar to the UUCP L.sys file. Each line contains four fields; fields are separated by colons:

(1) The system name of a site to which you forward news. Normally, you should include a line for all systems you have links to, as well a line for your own system.

(2) The newsgroups to be forwarded to them. This is a pattern in the sense of a subscription. Generally, you will list classes of newsgroups, that is, using all for everything. A typical forwarding list for a new site would be:

    net.all, fa.all, to.sysname

where sysname is the name of your contact site. Note that you should not forward all, in particular, since local newsgroups (those without dots) should not be sent. In the line describing your own system, this field describes the newsgroups your site will accept from contact sites. Thus, if another site insists on sending you a newsgroup you don’t want, say net.jokes, include !net.jokes here.

(3) Flags describing the connection between sites. These are:
A Indicates that the contact site is running an A version of netnews, or
B The contact site is running a B version. If neither A nor B is indicated here, default is B. If you are running the latest release of Sun software, you have a B version. If you aren’t sure which version your contact site is running, ask them before proceeding.
F Indicates that the fourth field is the name of a file. The full path name of a file containing the article in /usr/spool/news will be appended to this file.
L Prevents transmission unless the article was created on this site. Feeding an L link to a backbone site ensures that your submissions will be more likely to get to the entire network, even in the event of a local problem. Please make sure that a mail link exists too, so you can get replies.
U Arranges that the parameter to the optional %s in the command field be filled in with a permanent file name from /usr/spool/news instead of a temporary customized file name.

(4) The command to be run to send news to the remote site. The article will be on the standard input. A %s in the command will be replaced with the name of the file that contains the article. Leaving this field blank means an ordinary UUCP link is being used, that is, the command defaults to:

    uux --r -s sysname!rnews

Options here are:
- Tells uux to expect input on stdin.
--r Tells uux not to start up a daemon right away. This eases the load on the system, and makes news transmission only a bit slower. News is sent when the next daemon is started, usually the next time uucico is invoked by cron.
--s Shuts off the annoying message you would otherwise get mailed to you telling you that your article was successfully broadcast. The --s option is nonstandard; the remote system may need to be modified to understand it. To avoid using --s, put the uux command in the fourth field.
Here is a sample *sys* file for a site "zenith" with connections to "nadir". "Nadir" also passes news on to "lowerreaches". We assume that "zenith" and "lowerreaches" exchange a local newsgroup class *lng.all* as well as the network wide newsgroups. News to "lowerreaches" is batched.

```
zenith:net.all,fa.all,lng.all::
nadir:net.all,fa.all::
lowerreaches:net.all,fa.all,lng.all:F:/usr/spool/batch/lowerreaches
```

2. Setting up Links

There are two basic types of links for exchanging news: those that use mail and those that don't. The ones that use mail are more indirect, yet more versatile; the ones that don't are simpler. The default type is without mail, so we discuss it first.

2.1. Non-mail Links

The basic theory behind a non-mail link is that the *rnews* program is invoked on the remote system with the article being transmitted as the standard input. This is possible on various networks, but the most common implementation is via the UUCP network. Using the *uux(1C)* command, the command which is forked to the shell looks like:

```
uux --rc --r -z remotesys!rnews < article
```

This is the default transmission method. In order to set up such a link, obviously a UUCP link with the remote system must be in effect. In addition, *rnews* must be available and executable by *uuzqt* on the remote machine. In most cases, this means that *rnews* must be in /usr/bin so *uux* can find it.

2.2. Mail Links

When using mail to transmit articles, two intermediary programs are necessary: *sendnews(8)* and *uurec(8)*. The mail link works as follows. First, the folks at system A arrange to run *sendnews* on articles they wish to send to system B by placing an entry like the following in the *sys* file on system A:

```
/usr/lib/news/sendnews -a rnews@B
```

The -a option specifies that the mail should be formatted for the arpanet. When someone at system A sends an article to system B, *sendnews* packages the article and mails it to *rnews@B*. Somehow, system B must make sure that all mail to user "rnews" is fed as input to the *uurec* program. The best way to make this happen is to use *sendmail* or *delivermail*, if you are on a system running them. So the system B administrator creates an alias in /usr/lib/aliases like this:

```
rnews: "!/usr/lib/news/uurec"
```

When *uurec* receives the article, it unpackages the article and invokes *rnews*. 
3. Posting Methods

There are three ways to post news. The basic method is to use the `inews` command:

```
% inews -t title -n newsgroups < bodyfile
```

This is the primitive used by other programs, and is not very suitable for humans.

A somewhat friendlier front end is `postnews`. `Postnews` first prompts for article header lines: title, newsgroups, and distribution, and then places you in the editor so you can enter the text of your article. The system default EDITOR (`/usr/uclvi`) is used unless the environment variable EDITOR is set to override the system default. The header lines you entered at the beginning of the session remain available for editing at the top of the buffer; other header lines can also be added, such as expiration date. When you write out the file and exit from the editor, your article is posted.

Another method is to use mail. This is possible with Sun systems. To use mail, set up an alias such as the following for each newsgroup you subscribe to (adding new groups as they are created):

```
net.general: "/usr/lib/news/recnews net.general"
```

Now, whenever you send mail to `net.general` this starts up the given shell command which calls `recnews` with one argument: the name of the newsgroup. `Recnews` will in turn invoke `inews`.

Note that there are limits to `recnews`. There is no way to use it to post to multiple newsgroups without creating separate articles (something frowned upon because it forces people to read the same thing more than once). Nor is there any way to make the recording feature work when `recnews` is used (see the `Files` section above).

4. Control Messages

Some news systems send articles that are not for human consumption. These articles are messages to your news system called "control messages."

A control message begins with a "Control:" header, the subject of the article contains a command and zero or more arguments (much like a UNIX program), and the body of the article may be used for additional text. A list of commands follows.

Older systems use newsgroups matching `all.all.ctl`, rather than the "Control:" header, and this will still work, although the "Control:" header is preferred. Since the newsgroup name is used for distribution only, and is not checked to ensure that it's in the active file, such newsgroup names can still be used. This makes it possible to post network-wide control messages with `net.msg.ctl` (or restricted broadcast such as `btl.msg.ctl`) or messages for a particular system: `to.ucbvax.ctl`. Messages are cancelled with a "Control:" line in a message to the same newsgroup(s) as the original message.

Control messages are not stored in `/usr/spool/news`; rather they are acted on and discarded at once.

Control message commands are:

- `newgroup` Allows special action to be taken locally when a new newsgroup is created. The group itself is created with the `inews` command with the `-C` option, and this generates the message. By default, the newsgroup is added to the `active` file, a directory is created for it, and mail is sent to the local contact advising that this has happened. `newgroup` takes one argument: the name of the newsgroup to be
Control Messages

created.

rmgroup Cancels a newsgroup network-wide. *Rmgroup* takes one argument: the name of the newsgroup to be removed. There is also a shell script, "*rmgrp*" that cancels a newsgroup locally. We recommend that when you receive mail advising you of the demise of a newsgroup, you run *rmgrp* by hand; this prevents accidental or malicious removal of a good newsgroup.

cancel Cancels a given article. *Cancel* should be broadcast to the same newsgroup as the original article. *Cancel* takes one argument: the message ID of the article to cancel.

sendsys Mails the *sys* file to the originator of the message — used for making maps. *Sendsys* takes no arguments.

senduuname Runs *uuname(1)* and mails the output to the originator of the message — used for making UUCP maps. *Senduuname* takes no arguments.

version Mails the local version name/number of the netnews software to the originator of the message.

Other Messages

Any unrecognized message will cause an error message to be mailed to the originator. Additional messages may be defined as time goes on, such as messages to automatically update directories or maps.

5. Maintenance

There are a few things you should set up at the outset, and a few that you must do periodically, to keep your news system running smoothly. We hope to eventually automate all or most of this, but right now some of it must be done by hand.

To get articles to expire automatically, put the following line in *crontab*:

```
0 7 * * 2 su news < /usr/lib/news/news.week
```

This runs a shell script which runs *expire(8)* to delete all expired news. The --option to *expire* archives all expired news under /usr/spool/oldnews.

Sometimes news has not expired when it should have. If this seems to be happening, make sure that *expire* has permissions to unlink files, that it runs as a user that has a .newsrc, and that it is properly suid. You can manually invoke expire with the -v (verbose) option to find out what it's doing. Adding levels of verbosity (for example, "--v6") generates more and more output.

The *history* and *log* files in your /usr/lib/news directory will grow and must be cleaned out periodically. The /usr/lib/news/expire program removes lines from *history* corresponding to deleted articles, but it is a good idea to check the file every few months to make sure it is not going wild. Be sure not to completely lose your history file when you clean it up, in case a neighbor tries to send you an article you received recently. If you only get news from one site, it is safe to clean *history* out completely.

The *log* file is not automatically cleaned out by any netnews software, and will grow quickly. The news.week entry to /usr/lib/crontab noted above, however, will take care of cleaning the log file in addition to deleting expired articles.

You should also clean out old newsgroups that are no longer active. To remove a newsgroup, use the /usr/lib/news/rmgrp shell script.
Another task you'll probably have to undertake is clearing up UUCP constipation. If you have more than one connection, chances are that UUCP will get clogged up when one of your neighbors goes down for more than a few hours. Various spooling schemes are being worked on to help make the news/UUCP system more robust. Right now, UUCP is the weak link in netnews distribution, and you should certainly keep an eye on it.

6. Creating New Newsgroups

As system news administrator, you can create newsgroups. Before creating a newsgroup, first make sure this is the right thing to do. Normal etiquette runs as follows: a suggestion is first posted to net.general, net.news.group for a net newsgroup, followups are made to net.news.group, it is established if there is general interest in such a group, and a name is agreed on. Then you (as user "news") can create the newsgroup network-wide by typing the command:

```
% inewa -C newsgroup
```

This creates the newsgroup directory and active entry locally. It also prompts you for a paragraph describing the group, and starts up inews to post a newgroup control message announcing the group. This control message is sent out on net.mail.ctl; other sites may have configured their systems to do something with these messages. A human readable announcement is not made — you can post one to net.news.group if you wish.

You should make sure a first article is posted to the new newsgroup immediately. If this is not done, checknews will see the empty newsgroup directory and believe there is unread news (as each user lacks a "newsrc" line for the newsgroup).

7. Differences between Version 2.9 and 2.10

Both versions 2.9 and 2.10 are 'B' Versions of USENET format (just to confuse you). Version 2.9 was released with the Sun 0.4 and earlier software distributions; version 2.10 is released with Sun 1.0 and the current distributions. Differences between versions 2.9 and 2.10 follow; the section after this one discusses the differences between versions A and B of the USENET software.

New File Storage Format

The file storage format has been changed. Rather than storing news in /usr/spool/news/net.games.rogue/123, an article now goes in /usr/spool/news/net/games/rogue/123. This allows newsgroup names to be longer than 14 characters and still have subgroups. It also makes directories smaller, resulting in faster performance. The dot files are gone: rather than saving the next article number in /usr/spool/news/.net.games.rogue as the length of the file, it goes in the active file on the same line as net.games.rogue. Thus, your active file contains lines like

```
net.games.rogue 00123
```

where the newsgroup name and the max article number are separated by a space. The article numbers are ALWAYS 5 digits long and include the leading digits to do this (this is so they can be updated in place without growing the active file).

This conversion of directory tree formats has an extra benefit. You'll find that readnews is now considerably faster than in version 2.9. The movement of the dot files accounts for much of this, since it is no longer necessary to "stat" every dot file. Additionally, a routine to find a newsgroup in your .newsarc has been modified to keep the file sorted in the same
Differences between Version 2.9 and 2.10

order as active, and a "last found" pointer is used to reduce the find time algorithm from quadratic complexity (on the number of newsgroups) to linear complexity. This makes the total number of newsgroups less of a factor in determining speed, and also keeps everyone's .newsrc cleaned out. It is important that people not store extra junk in their .newsrc, because it will be deleted when they run readnews.

New Hashed History
The method used to determine if an incoming article has already been seen locally has been changed. On V7 systems, profiling showed that rnews spent about half of its time in fgets reading the history file. It now uses the dbm(3) library to hash the message ID of each article. To avoid incompatibilities between 2.9 and 2.10, if you have more than one incoming news feed, run the provided evt.hist program (see the Conversion section for the script), which will enter all the message ID's from your 2.9 format history file into the hashed database. The text history file is maintained as it was in 2.9, for human use.

New Message Header Format
Message headers now meet the USENET format standard* and RFC822. Headers stored will look verbose, and contain much more information. Headers transmitted to other systems will work with old B news systems or new ones. The format of dates has been changed to conform to RFC822.

New User Interface
The user interface is roughly compatible, but you will notice a few differences, and there are a few extensions. One major difference is that postnews now prompts for a "Distribution". This defaults to the same as the newsgroup (and is omitted in this case), but allows you to conveniently enter a Distribution header line (and makes you think twice about where your message is going to). Any newsgroup name(s) can be typed here, but one normally types either nothing or a class distribution ("net", "btl", "nj", or "world", for example), restricting the distribution to that class of sites.

A more minor change: headers are now displayed in a format which is more compact, but more information is displayed than before. If you want to see the article ID or full path, or the date or newsgroups, use the h command for a display. The H command can be used for a full, verbose header dump.

Changed Control Messages
Control messages are slightly different. In particular, 2.10 now requires that a newsgroup be created with a newsgroup control message (generated by inews -C) before it can be used. If an incoming article is in some newsgroup that is not in the local active file, the article will be stored locally in newsgroup junk and not forwarded to other systems. This will prevent the accidental creation of typographical errors by systems running older versions of news.

Also, the rmgroup control message has been made less dangerous. When an rmgroup message comes around, you will be sent mail asking you to remove the group, but the group will not actually be removed. This prevents someone from accidentally or deliberately removing an important newsgroup such as net.general.

---

Archiving

Expire used to automatically archive news in /usr/spool/oldnews if that directory existed; it now archives news only if you supply the --a option.

Sharing News Software

When you connect to a new site, you must now send a copy of your active file, so that the appropriate newsgroup directories and active file can be built. This was not necessary in 2.9 because unrecognized incoming newsgroups were automatically created, but they are thrown away in 2.10.

New Programs

Some new programs have been added to /usr/lib/news. These include recmail and caesar. Recmail takes a mail message on stdin, figures out who the To: and Cc: lines refer to, and invokes /bin/mail with those arguments (not changed) and with the file on stdin. Recmail is used by the reply command. Caesar decodes rotated jokes; it can also be used to create rotated jokes.

New Mail Batching Provisions

Version 2.10 has some experimental batching provisions in it. See the batch and unbatch programs, and the F and U options in the sys file, for more details. All this is very new, and while it appears to run satisfactorily, caution is advised in installing batching software. You must also make sure your neighbor is prepared to receive batched news; this is normally true if the neighbor is running at least 2.10.

8. Version A Versus Version B

Version B automatically understands incoming news in either version A or B format. Version B generates either format, depending on the flag in the third field of the description line of the sys file. Thus, it would be possible for two version B sites to communicate using version A format — though it would not be a good idea, since the translation from B to A results in a permanent loss of some header information (such as expiration date).

Version A does not understand version B format.

News from versions A and 2.9 B do not conform to the USENET interchange standard. 2.10 supports the standard and will communicate with either A or 2.9 B news. 2.10 will write out headers with both standard (Date, Message-ID) and 2.9 (Posted, Article-I.D.) lines so that either B system will properly handle the article. Incoming news is recognized by the first letter ("A" for A news), or the lack of an "@" in the From line (2.9). Missing fields are constructed as well as possible from the available information.
# Line Printer Spooler Manual

## Table of Contents

1. Overview ........................................................................................................................................ 1
2. Commands ....................................................................................................................................... 1
   2.1. Ipdl line printer daemon .......................................................................................................... 1
   2.2. Ipq show line printer queue ................................................................................................. 2
   2.3. Ipdm remove jobs from a queue ............................................................................................ 2
   2.4. Ipc line printer control program .......................................................................................... 2
3. Access Control ................................................................................................................................ 3
4. Setting Up .................................................................................................................................... 3
   4.1. Creating a Printcap File .................................................................................................... 3
      4.1.1. Printers on Serial Lines .............................................................................................. 3
      4.1.2. Remote Printers ......................................................................................................... 4
   4.2. Output Filters ...................................................................................................................... 4
5. Output Filter Specifications ......................................................................................................... 5
6. Line Printer Administration .......................................................................................................... 5
7. Troubleshooting ............................................................................................................................. 6
This document describes the structure and installation procedure for the line printer spooling system implemented in this release of the Sun UNIX* operating system.

1. Overview

The line printer system supports:
- multiple printers,
- multiple spooling queues,
- both local and remote printers, and
- printers attached via serial lines which require line initialization such as the baud rate.

Raster output devices such as a Varian or Versatec, and laser printers such as an Imagen, are also supported by the line printer system.

The line printer system consists mainly of the following files and commands:

- `/etc/printcap` — printer configuration and capability data base
- `/usr/lib/lpd` — line printer daemon, does all the real work
- `/usr/ucb/lpr` — program to enter a job in a printer queue
- `/usr/ucb/lpq` — spooling queue examination program
- `/usr/ucb/lprm` — program to delete jobs from a queue
- `/etc/lpc` — program to administer printers and spooling queues
- `/dev/printer` — socket on which lpd listens

The file `/etc/printcap` is a master data base describing line printers directly attached to a machine and, also, printers accessible across a network. The manual page entry `printcap(5)` provides the ultimate definition of the format of this data base, as well as indicating default values for important items such as the directory in which spooling is performed. This document highlights the important information which may be placed in `printcap`.

2. Commands

2.1. lpd — line printer daemon

The program `lpd(8)`, usually invoked at boot time from the `/etc/rc` file, acts as a master server for coordinating and controlling the spooling queues configured in the printcap file. When `lpd` is started it makes a single pass through the `printcap` database restarting any printers which have
jobs. In normal operation \textit{lpd} listens for service requests on multiple sockets, one in the UNIX domain (named `"/dev/printer"`) for local requests, and one in the Internet domain (under the "printer" service specification) for requests for printer access from off machine; see \textit{socket}(2) and \textit{services}(5) for more information on sockets and service specifications, respectively. \textit{Lpd} spawns a copy of itself to process the request; the master daemon continues to listen for new requests.

Clients communicate with \textit{lpd} using a simple transaction oriented protocol. Authentication of remote clients is done based on the "privilege port" scheme employed by \textit{rshd}(8C) and \textit{rcmd}(3X). The following table shows the requests understood by \textit{lpd}. In each request the first byte indicates the "meaning" of the request, followed by the name of the printer to which it should be applied. Additional qualifiers may follow, depending on the request.

<table>
<thead>
<tr>
<th>Request</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A printer\n</td>
<td>check the queue for jobs and print any found</td>
</tr>
<tr>
<td>B printer\n</td>
<td>receive and queue a job from another machine</td>
</tr>
<tr>
<td>C printer [users ...] [jobs ...]\n</td>
<td>return short list of current queue state</td>
</tr>
<tr>
<td>D printer [users ...] [jobs ...]\n</td>
<td>return long list of current queue state</td>
</tr>
<tr>
<td>E printer person [users ...] [jobs ...]\n</td>
<td>remove jobs from a queue</td>
</tr>
</tbody>
</table>

The \textit{lpr}(1) command is used by users to enter a print job in a local queue and to notify the local \textit{lpd} that there are new jobs in the spooling area. \textit{Lpd} either schedules the job to be printed locally, or in the case of remote printing, attempts to forward the job to the appropriate machine. If the printer cannot be opened or the destination machine is unreachable, the job will remain queued until it is possible to complete the work.

2.2. \textbf{lpq – show line printer queue}

The \textit{lpq}(1) program works recursively backwards displaying the queue of the machine with the printer and then the queue(s) of the machine(s) that lead to it. \textit{Lpq} has two forms of output: in the default, short, format it gives a single line of output per queued job; in the long format it shows the list of files, and their sizes, which comprise a job.

2.3. \textbf{lprm – remove jobs from a queue}

The \textit{lprm}(1) command deletes jobs from a spooling queue. If necessary, \textit{lprm} will first kill off a running daemon which is servicing the queue, restarting it after the required files are removed. When removing jobs destined for a remote printer, \textit{lprm} acts similarly to \textit{lpq} except it first checks locally for jobs to remove and then tries to remove files in queues off-machine.

2.4. \textbf{ipc – line printer control program}

The \textit{ipc}(8) program is used by the system administrator to control the operation of the line printer system. For each line printer configured in \texttt{/etc/printcap}, \textit{ipc} may be used to:

- disable or enable a printer,
- disable or enable a printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
3. Access Control

The printer system maintains protected spooling areas so that users cannot circumvent printer accounting or remove files other than their own. The strategy used to maintain protected spooling areas is as follows:

- The spooling area is writable only by a daemon user and daemon group.
- The lpr program runs setuid root and setgid daemon. The root access is used to read any file required, verifying accessibility with an access(2) call. The group ID is used in setting up proper ownership of files in the spooling area for lprm. Data files are owned by user, group daemon, mode 660.
- Control files in a spooling area are made with daemon ownership and group ownership daemon. Their mode is 0660. This insures control files are not modified by a user and that no user can remove files except through lprm.
- The spooling programs, lpd, lpq, and lprm run setuid root and setgid daemon to access spool files and printers.
- The printer server, lpd, uses the same verification procedures as rshd(8C) in authenticating remote clients. The host on which a client resides must be present in the file /etc/hosts.equiv, used to create clusters of machines under a single administration.

In practice, none of lpd, lpq, or lprm would have to run as user root if remote spooling were not supported. In previous incarnations of the printer system lpd ran setuid daemon, setgid spooling, and lpq and lprm ran setgid spooling.

4. Setting Up

The Sun system release comes with the necessary programs installed and with the default line printer queue created. The real work in setting up is to create the printcap file and any printer filters for printers not supported in the distribution system.

4.1. Creating a Printcap File

The printcap database contains one or more entries per printer. A printer should have a separate spooling directory; otherwise, jobs will be printed on different printers depending on which printer daemon starts first. This section describes how to create entries for printers which do not conform to the default printer description.

4.1.1. Printers on Serial Lines

When a printer is connected via a serial communication line it must have the proper baud rate and terminal modes set. The following example is for a DecWriter III printer connected locally via a 1200 baud serial line.
The `lp` entry specifies the file name to open for output. In this case it could be left out since "/dev/lp" is the default. The `br` entry sets the baud rate for the tty line and the `fs` entry sets `CRMOD, no parity, and XTAB8` (see `tty(4)`). The `tr` entry indicates a form-feed should be printed when the queue empties so the paper can be torn off without turning the printer off-line and pressing form feed. The `of` entry specifies the filter program `lpf` should be used for printing the files; more will be said about filters later. The last entry causes errors to be written to the file "~/usr/adm/lpd-errs" instead of the console.

### 4.1.2. Remote Printers

Printers which reside on remote hosts should have an empty `lp` entry. For example, the following `printcap` entry would send output to the printer named "lp" on the machine "ucbvax".

```plaintext
lp[default line printer:]
 :lp=/dev/lp:br=#1200:fs=#0320:
 :tr=\f:of=/usr/lib/lpf:lf /usr/adm/lpd-err:
```

The `rm` entry is the name of the remote machine to connect to; this name must appear in the `/etc/hosts` database, see `hosts(5)`. The `rp` capability indicates the name of the printer on the remote machine is "lp"; in this case it could be left out since this is the default value. The `sd` entry specifies "~/usr/spool/vaxlpd" as the spooling directory instead of the default value of "~/usr/spool/lpd".

### 4.2. Output Filters

Filters are used to handle device dependencies and to perform accounting functions. The output filter `of` is used to filter text data to the printer device when accounting is not used or when all text data must be passed through a filter. It is not intended to perform accounting since it is started only once, all text files are filtered through it, and no provision is made for passing owners' login name, identifying the beginning and ending of jobs, etc. The other filters (if specified) are started for each file printed and perform accounting if there is an `af` entry. If entries for both `of` and one of the other filters are specified, the output filter is used only to print the banner page; it is then stopped to allow other filters access to the printer. An example of a printer which requires output filters is the Benson-Varian.

```plaintext
valvarian[Benson-Varian:]
 :lp=/dev/va0:sd=/usr/spool/vad:of=/usr/lib/vpf:
 :tr=/usr/lib/rvcat:mx#2000:pl#58:tr=\l:
```

The `tf` entry specifies "~/usr/lib/rvcat" as the filter to be used in printing `troff(1)` output. This filter is needed to set the device into print mode for text, and plot mode for printing `troff` files and raster images (see `vp(4S)`). Note that the page length is set to 58 lines by the `pl` entry for 8.5" by 11" fan-fold paper. To enable accounting, the `varian` entry would be augmented with an `af` filter as shown below.
5. Output Filter Specifications

For most devices or accounting methods, it is probably necessary to create a new filter.
Filters are spawned by lpd with their standard input the data to be printed, and standard output the printer. The standard error is attached to the If file for logging errors. A filter must return a 0 exit code if there were no errors, 1 if the job should be reprinted, and 2 if the job should be thrown away. When lprm sends a kill signal to the lpd process controlling printing, it sends a SIGTERM signal to all filters and descendents of filters. This signal can be trapped by filters which need to perform cleanup operations such as deleting temporary files.

Arguments passed to a filter depend on its type. The of filter is called with the following arguments:

    ofiler -wwidth -llength

The width and length values come from the pw and pl entries in the printcap database. The if filter is passed the following parameters.

    filter [-c] -wwidth -llength -iindent -D login -h host accounting_file

The -c flag is optional, and only supplied when control characters are to be passed uninterpreted to the printer (when the -l option of lpr is used to print the file). The -w and -l parameters are the same as for the of filter. The -n and -h parameters specify the login name and host name of the job owner. The last argument is the name of the accounting file from printcap.

All other filters are called with the following arguments:

    filter -xwidth -ylength -n login -h host accounting_file

The -x and -y options specify the horizontal and vertical page size in pixels (from the px and py entries in the printcap file). The rest of the arguments are the same as for the if filter.

6. Line Printer Administration

The lpc program provides local control over line printer activity. The major commands and their intended use will be described. The command format and remaining commands are described in lpc(8).

abort and start

Abort terminates an active spooling daemon on the local host immediately and then disables printing (preventing new daemons from being started by lpr). This is normally used to force a hung line printer daemon to restart (i.e., lpq reports that there is a daemon present but nothing is happening). It does not remove any jobs from the queue (use the lprm command instead). Start enables printing and requests lpd to start printing jobs.

enable and disable
Enable and disable allow spooling in the local queue to be turned on/off. This will allow/prevent lpr from putting new jobs in the spool queue. It is frequently convenient to turn spooling off while testing new line printer filters since the root user can still use lpr to put jobs in the queue but no one else can. The other main use is to prevent users from putting jobs in the queue when the printer is expected to be unavailable for a long time.

restart
Restart allows ordinary users to restart printer daemons when lpq reports that there is no daemon present.

stop
Stop is used to halt a spooling daemon after the current job completes; this also disables printing. This is a clean way to shutdown a printer in order to perform maintenance, etc. Note that users can still enter jobs in a spool queue while a printer is stopped.

topq
Topq places jobs at the top of a printer queue. This can be used to reorder high priority jobs since lpr only provides first-come-first-serve ordering of jobs.

7. Troubleshooting
There are a number of messages which may be generated by the line printer system. This section categorizes the most common and explains the cause for their generation. Where the message indicates a failure, directions are given to remedy the problem.

In the examples below, the name printer is the name of the printer. This would be one of the names from the printcap database.

LPR

lpr: printer: unknown printer
The printer was not found in the printcap database. Usually this is a typing mistake; however, it may indicate a missing or incorrect entry in the /etc/printcap file.

lpr: printer: jobs queued, but cannot start daemon.
The connection to lpd on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check the local socket /dev/printer to be sure it still exists (if it does not exist, there is no lpd process running). Use

```
% ps ax | fgrep lpd
```
to get a list of process identifiers of running lpd's. The lpd to kill is the one which is not listed in any of the "lock" files (the lock file is contained in the spool directory of each printer). Kill the master daemon using the following command.

```
% kill pid
```
Then remove /dev/printer and restart the daemon (and printer) with the following commands.

```
% rm /dev/printer
% /usr/lib/lpd
```
Another possibility is that the lpr program is not setuid root, setgid spooling. This can be checked with

```
% ls -lg /usr/ucb/lpr
```

lpr: printer: printer queue is disabled
This means the queue was turned off with

```
% lpc disable printer
```
to prevent lpr from putting files in the queue. This is normally done by the system manager when a printer is going to be down for a long time. The printer can be turned back on by a super-user with lpc.

---

**LPQ**

**Waiting for printer to become ready (offline ?)**
The printer device could not be opened by the daemon. This can happen for a number of reasons, the most common being that the printer is turned off-line. This message can also be generated if the printer is out of paper, the paper is jammed, etc. The actual reason is dependent on the meaning of error codes returned by system device driver. Not all printers supply sufficient information to distinguish when a printer is off-line or having trouble (e.g. a printer connected through a serial line). Another possible cause of this message is some other process, such as an output filter, has an exclusive open on the device. Your only recourse here is to kill off the offending program(s) and restart the printer with lpc.

*printer is ready and printing*
The lpq program checks to see if a daemon process exists for printer and prints the file status. If the daemon is hung, a super user can use lpc to abort the current daemon and start a new one.

**Waiting for host to come up**
This indicates there is a daemon trying to connect to the remote machine named host in order to send the files in the local queue. If the remote machine is up, lpd on the remote machine is probably dead or hung and should be restarted as mentioned for lpr.

**Sending to host**
The files should be in the process of being transferred to the remote host. If not, the local daemon should be aborted and started with lpc.

**Warning: printer is down**
The printer has been marked as being unavailable with lpc.

**Warning: no daemon present**
The lpd process overseeing the spooling queue, as indicated in the “lock” file in that directory, does not exist. This normally occurs only when the daemon has unexpectedly died. The error log file for the printer should be checked for a diagnostic from the deceased process. To restart an lpd, use

```
% lpc restart printer
```

**lprm: printer: cannot restart printer daemon**
This case is the same as when lpr prints that the daemon cannot be started.
LPD

The `lpd` program can write many different messages to the error log file (the file specified in the If entry in `printcap`). Most of these messages are about files which cannot be opened and usually indicate the `printcap` file or the protection modes of the files are not correct. Files may also be inaccessible if people manually manipulate the line printer system (i.e. they bypass the `lpr` program).

In addition to messages generated by `lpd`, any of the filters that `lpd` spawns may also log messages to this file.

LPC

could't start printer
This case is the same as when `lpr` reports that the daemon cannot be started.

cannot examine spool directory
Error messages beginning with "cannot ..." are usually due to incorrect ownership and/or protection mode of the lock file, spooling directory or the `lpc` program.