sun®
microsystems

# System & Network Administration

Part Number: 800-1733-10
Revision A, of 9 May 1988

Chapters 17 through 22 in the final section of this manual were originally derived from the following works:

*Fsck — The UNIX File System Check Program*
was originally written by T. J. Kowalski, Bell Laboratories, Murray Hill, New Jersey; and was revised by Marshall Kirk McKusick, Computer Systems Research Group, University of California at Berkeley.

*Sendmail — An Internetwork Mail Router*
was originally written by Eric Allman, formerly of Project Ingres, University of California at Berkeley.

*Sendmail — Installation and Operation Guide*
was originally written by Eric Allman, formerly of Project Ingres, University of California at Berkeley.

*Uucp Implementation Description*
was originally written by D. A. Nowitz, Bell Laboratories, Murray Hill, New Jersey.

*3BSD Line Printer Spooler Manual*
was originally written by Ralph Campbell, Computer Systems Research Group, University of California at Berkeley

*Name Server Operations Guide*
was originally published by the Regents of the University of California at Berkeley

# Contents

# Tables

# Figures

# Preface

*System and Network Administration* is a resource manual for managing Sun workstations running Release 4.0. It contains procedures for maintaining all types of Sun equipment configurations—mostly, but not entirely, from a software perspective—and explains system administration and networking concepts for administrators at all levels of expertise.

This manual is a companion to *Installing the SunOS* The text assumes that you have already installed Release 4.0 on new or already-in-use Sun workstations, which you now are going to maintain and manage.

## Types of Configurations

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

□ A workstation without a disk, also called a *diskless client*, that will rely on a network for essential services.

□ A workstation with a disk, also called a *dataless client*, that will rely on a network for essential services.

□ A *standalone* workstation with a disk and tape drive that does not depend on a network for essential services, however, you can attach it to a network.

□ A *network* with at least one *server* machine exporting its services, and other machines using these services.

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

## How to Use This Manual

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

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

□ Part One, *System Administration for Beginners* explains basic concepts relevant to administering all types of Sun equipment configurations, including general networking concepts.

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

□ Part Two, *System Administration Procedures*, explains concepts and contains instructions for maintaining all types of Sun configurations. Topics include the Release 4.0 file systems, boot up and shutdown procedures, backing up and restoring file systems, system security issues, and adding new users and equipment.

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

□ Part Three, *Network and Communications Administration*, explains how to set up and maintain a local area network from a software perspective, including administering the network file system, yellow pages, and electronic mail. It also explains how to set up a modem and install the `uucp` wide area network.

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

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

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

## Supporting Documentation

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

□ *Installing the SunOS* for Release 4.0

□ *SunOS Reference Manual*

□ *Network Programming*

□ *Writing Device Drivers* (for advanced administrators)

## Documentation Conventions

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

□ Prompts and error messages from the system are printed in `listing font like this.`

□ Information that you type as a command or in response to prompts is shown in **`boldface listing font like this`**. Type everything shown in boldface exactly as it appears.

□ Where parts of a command are shown in *italic text like this*, they refer to a variable that you have to substitute from a selection; it is up to you to make the proper substitution.

□ Dialogues between you and the system are enclosed in gray boxes like the following:

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

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

```
int test[100];

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

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

# Part One: System Administration for Beginners

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

- The system administrator's tasks
- UNIX† Operating system concepts and disk formatting concepts
- Introduction to networking and Sun network services

Part One concludes with a glossary of terms.

## Who Should Read This Part

You should read Part One if you are:

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

## Prerequisite Knowledge

All procedures in this manual assume that you have read *Installing the SunOS* for Release 4.0, and have already installed this release using the `format` and `suninstall` utilities. The text also assumes that you have a basic familiarity with SunOS or UNIX system commands. If you are not familiar with these commands, it is suggested that you read two beginners guides, *Getting Started with SunOS: Beginner's Guide* and *Doing More with SunOS: Beginner's Guide.*

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

† UNIX is a registered trademark of AT&T.

# 1

# The System Administrator's Role

# 1

# The System Administrator's Role

This chapter briefly explains the types of tasks you will perform as a Sun system administrator. It also points out the basic knowledge you need for managing each type of configuration, and refers you to places in this manual and others in your Sun docubox where you can find this information.

## 1.1. System Administration--The Big Picture

This section provides a broad overview of the equipment you will manage and the tasks you need to perform as a system, and possibly network, administrator.

### Sun Equipment Configurations

As system administrator, you will be responsible for managing one or more of four types of equipment configurations. These configurations, completely described in *Installing the SunOS* are briefly listed below:

□ Server

Any computer system that provides a network service, such as disk storage, file service, and electronic mail is considered a *server*. Machines that receive these network services are called *clients*. Servers are often classified by the services they provide. For example, a server that enables a client to share its files is called a *file server*. Other categories of servers are described in Chapter 3, "Introducing Networks."

File servers are often described as either homogeneous or heterogeneous, depending on the architectures of the machines on the network they serve. The term *architecture* refers to the type of CPU chip used in the machine

| *Product Type* | *Architecture* |
|---|---|
| Sun-2 | 68010 |
| Sun-3 | 68020 |
| Sun-4 | SPARC |

A *homogeneous server* supports client machines with the same architecture as it has. A *heterogeneous server* supports client machines that have both the same and different architectures than the server. Additionally, a heterogeneous server may support equipment from manufacturers other than Sun—a feature of Sun's "open systems" approach to networking.

□ *Diskless client*

This is a computer that does not have its own disk. To fully operate, it must be attached to a local area network. The diskless client relies on the network server for disk storage and other services. Most significantly, it relies upon the server for the programs that enable it to boot up.

- *Dataless client*

  This computer system has its own local disk, which contains its root and `swap` partitions. It must be attached to a network because although it can independently boot, it cannot come up single or multiuser until receiving major file systems, such as `/usr`, from the server.

- *Standalone system*

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

- *Time-sharing system*

  This is a standalone system with terminals attached to its serial ports. The terminals rely on the standalone not only for disk storage and files, but also CPU time, since terminals do not have their own CPUs. The time-sharing system may or may not be on a network.

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

## The System Administrator's Tasks

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

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

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

## Becoming the Superuser

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

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

You can become superuser in either of two ways:

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

The expressions "superuser" and "root" (implying the root user name) are often used interchangeably in SunOS and other UNIX documentation. For example, certain instructions in this manual will tell you to log in as superuser. Depending on the screen you are using, this simply means type either `su` or `root` to become superuser.

### Assigning a Password to the Root User Name

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

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

1. Type the following to log in as superuser.

```
Name: root
Password:
```

2. Press [RETURN] at the password prompt if you are logging in to a brand new system. If your system is already in use, type the password its previous administrator gave to you. The system then displays a pound sign (#) prompt, signifying that you are now logged in as root.

3. Change the password as shown below:

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

where *your password* is the password you want for the root user name. The password should be at least five characters long; do not use your name or another obvious word as the root password. The system asks you to type the new password twice as a security measure. If you do not type the same letters in response to the retype prompt, the system will not let you change the password.

### Logging in as Superuser from a Shell

For safety's sake, you should not log in as root to perform any operations other than system administrative ones. Therefore, you will constantly log in under your regular user name, move to root for system administration tasks, then go back to your regular shell during the course of a day. To do this, you use the `su` command from the shell as follows:

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

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

## Road Map through System Administration Documentation

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

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

| *Activity* | *Where It's Described* |
|---|---|
| Understanding boot and shutdown | Chapter 5 |
| Modifying `/etc/passwd` | Chapter 8 |
| Modifying `/etc/fstab` | Chapter 13 |
| Where to make mount points | Chapter 13 |
| Handling overloaded file systems | Chapters 7 & 8 |
| Adding and maintaining printers | Chapter 11 |
| Setting up mail aliases | Chapter 15 |
| Adding additional boards | Chapter 11 |
| Diagnosing and fixing problems | Chapter 8 |

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

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

| *Activity* | *Where It's Described* |
|---|---|
| Maintaining disk subsystems | Chapter 10 |
| Installing software | Installing the SunOS, software manuals |
| Doing remote backups/restores | Chapter 7 |
| Understanding yellow pages | Chapter 14 |
| Modifying `/etc/hosts.equiv` | Chapter 13 |
| Files and commands in `/etc` | Chapter 6 |

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

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

| *Activity* | *Where It's Described* |
|---|---|
| Understanding boot & shutdown | Chapter 5 |
| Installing software | Installing the SunOS, software manuals |
| Doing local backups/restores | Chapter 7 |
| Maintaining disk subsystems | Chapter 10 |
| Knowing files and commands in `/etc` | Chapter 6 |
| Modifying permissions | Chapter 8 |
| Handling overloaded file systems | Chapters 7 & 8 |
| Adding and maintaining printers | Chapter 11 |
| Adding additional boards | Chapter 11 |
| Adding modems and terminals | Chapter 11 |
| Diagnosing and fixing problems | Chapter 8 |
| Reconfiguring the Kernel | Chapter 9 |

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

| *Activity* | *Where It's Described* |
|---|---|
| Understanding networks | Chapter 12 |
| Modifying `/etc/fstab` | Chapter 13 |
| Where to make mount points | Chapter 13 |
| Setting up network aliases | Chapter 15 |
| Understanding yellow pages | Chapter 14 |
| Modifying `/etc/hosts.equiv` | Chapter 13 |

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

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

| *Activity* | *Where It's Described* |
|---|---|
| Setting up NFS | Chapter 13 |
| Setting up yellow pages | Chapter 14 |
| Setting up the mail router | Chapter 15 |
| Setting up a wide area network | Chapters 12 & 15 |

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

## 1.2. Developing Your Administration Procedures

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

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

□ Keep a notebook describing the layout of the system, including a history of any changes you have made. In particular, you should save hard copy records of your disk label(s).

□ Because you are not experienced with the Sun environment, start with a "generic" system, using defaults provided by `suninstall`, `format`, and other files and programs. Customize your environment only after you've gained experience and some expertise.

□ Make backup tapes regularly. This may be the system administrator's most crucial tasks. Without backup tapes, lost files are gone forever.

□ If you are going to make a major change to a file system, do full backups first. This is in addition to the incremental backups that you should do regularly.

□ Plan any changes completely before implementing them. If you forget a step or do something out of order, you might introduce big problems.

□ If you run into trouble and are not sure what to do, call Sun Technical Support. Note that if the warranty period has expired on your system, you will be charged for the call unless you have a support contract.

# 2

# Overview of SunOS

# 2

# Overview of SunOS

## 2.1. Some Basic Terms

The Sun operating system (SunOS) consists of the *kernel* and many hundreds of other files containing data and programs. At boot time the *kernel* is loaded into memory; it resides there until the system is shut down. Other files are loaded into memory as needed.

The kernel manages all of the physical resources of the workstation. Some of the kernel's functions include:

1. Implementing the *file system*, which is a tree-structured hierarchy of directories and files residing on local disk or remote disk (and accessed with the Network File System), and permits processes to create, read, write, and remove these files.
2. *Virtual memory*, which allows unsegmented address space for each process by automatically moving "pages" of information from disk to main memory as needed.
3. The *scheduler*, which keeps track of all active processes and decides which process gets to run next.
4. *Device drivers*, which are software routines that control physical devices such as graphics display, mouse, keyboard, disk, tape, RS-232 serial ports, and Ethernet.
5. *Networking software* which implements network functions such as the TCP/IP protocols.
6. *Interprocess communication* facilities such as signals and sockets.
7. Facilities for creating, examining, and modifying processes.
8. System management functions, such as halting, booting, and error handling.
9. Miscellaneous functions which make system resources (like memory, timers, etc.) available to processes.

The kernel resides on the disk in a single file, typically known as `/vmunix`. (There may be other kernels with other names). There are many hundreds of other files comprising the UNIX system. Most of these are *commands*, also called *utilities*, which are programs you invoke directly. Other files contain *libraries*. These are collections of software routines which may be selected from the library and incorporated into another program. Libraries form an extension of the basic system features implemented by the kernel. Many files contain *data* used by the programs in other files.

## 2.2. Using the *SunOS Reference Manual*

The *SunOS Reference Manual* provided in your docubox is a comprehensive reference tool, describing all commands, system programs, and system files comprising the Sun Operating System. It is divided into eight sections with alphabetically listed entries, much like an encyclopedia. Each command, program, or file has an individual entry, refered to as a manual page, or, informally, as a *man page.*

The sections contain the following information:

- Section 1 describes the user commands, sometimes refered to as applications.
- Section 2 describes system calls.
- Section 3 describes library routines.
- Section 4 describes the special files used for devices.
- Section 5 explains major system files, many of which are used for system administration.
- Section 6 describes games and demonstration programs available from Sun.
- Section 7 describes what are known as "public files." These files include various tables and macro packages used with the `nroff` and `troff` formatting programs.
- Section 8 describes the system administration and maintenance commands.

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

`passwd`(5)

This means that the entry is in Section 5 of the *SunOS Reference Manual* and its title is `passwd`(5).

Release 4.0 allows you during `suninstall` to select the online version of the man pages as an option. Should you choose this option, you can display a formatted version of a particular man page on your screen. The man pages are located in the directory `/usr/share/man`. Subdirectories either begin with the word `man`*x* or `cat`*x* where *x* is a number from 1 to 8, indicating a section in the *SunOS Reference Manual* The `man`*1-8* directories contain all man pages in `troff` source format. The `cat`*1-8* directories contain all man pages as formatted output.

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

```
% man entry_name
```

where *entry_name* is the name of the entry you want to display:

3

# Introducing Networks

# 3

# Introducing Networks

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

□ Types of networks

□ Entities on the network

□ The network administrative domain

A network is a group of machines connected together so that they can transfer information among each other. Machines attached to the network are called *hosts*.

Networks consist of both hardware connections and software interfaces that allow transfer of information to and from a communications line. Network hardware includes various types of controllers, cabling, and data conversion devices used by servers and clients. Network software includes high-level services, such as the Network File System, lower level software that translate information so that it can be handled by the higher level services, and network interface drivers, that manage the activities of the communications controllers.

## 3.1. Performing Network Administration Tasks

Network administration is a complex subject; this manual does not cover all its aspects. You may need to refer to other documentation depending on the networking tasks that you wish to perform.

For a basic understanding of networking, familiarize yourself with the terms presented in this chapter. This is recommended even if your machine is a diskless client or a non-networked standalone. Part Three, "Network and Communications Administration," contains instructions for setting up and maintaining servers and clients on a network, from the software perspective. It assumes you are familiar with the concepts in Chapter 3. If you want to learn more about network programming, refer to the *Network Programming* manual in your docubox.

System administrators also are responsible for managing the network hardware. The extent of your hardware responsibilities depends on your system configuration and your site's requirements.

□ If your machine is a client on network, you probably will not be responsible for network assembly and maintenance. For instructions for attaching your machine to the network, refer to the hardware manuals that came with the machine.

- □ If you are the new administrator of an existing network server, you probably will have to maintain machines and cabling on the network and add new equipment as needed. Therefore, read the hardware manuals that came with your equipment, including manuals for peripherals like printers and modems. Remember that you will have to fix network hardware problems as they arise.
- □ If you are the administrator of an entirely new network, you have a different set of considerations. Your site may have individuals who plan and assemble network hardware; in that case, your hardware responsibilities will most likely be at the maintenance level. If you are responsible for assembling the network, refer to the hardware manuals shipped with your server, workstations, cabling, and controller boards for instructions.

## 3.2. Types of Networks

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

### Local Area Networks

Local area networks handle data communications over a physically limited area, such as single buildings. LANs consist of machines, including some that provide networking services, connected together by media such as cabling. Because the length of a local area network is typically limited by its media to a short distance, it is usually owned by the company that purchases the networking equipment. For example, if your Sun workstation is part of a local area network, your company most likely owns this network.

All Sun computers except standalones need to be part of a network in order to operate, but this network does not have to be a wide or medium area network. Therefore, in this manual, the generic term "network" assumes a local area network, unless otherwise specified. Refer to Chapters 12 through 15 for information about setting up a local area network. Chapters 8 and 11 also contain information about local area network maintenance.

### Wide Area Networks

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

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

SunOs Release 4.0 provides your machine with the ability to connect to two types of wide-area networks: `uucp` and Internet. `uucp`, which stands for *UNIX-to-UNIX copy*, is actually a program that uses the telephone network. `uucp` was originally designed as a facility for copying files from one computer running the UNIX operating system to another also running UNIX. Today, the term uucp refers not only to this file copy capability but also to a loose connection of systems with modems that have known phone numbers. This organization is sometimes referred to as the uucp network, but it is not a network in the sense that Internet is. This is because uucp is not based on networking protocols such as those developed by the International Standards Organization (ISO), as described in Chapter 12.

All Sun workstations have uucp capabilities. If you want to set up Sun machines at distances remote from your site, for example, in employees' homes, you can use uucp. Chapter 11, "Adding Hardware to Your System," explains how to set up a modem. Chapters 15, "Electronic Mail and Communications," and 21, "UUCP Implementation Description," explain how to set up and maintain uucp.

The Internet a registered wide-area network originally developed by DARPA (Defense Advanced Research Projects Agency). Today the Network Information Center (NIC) at SRI International maintains the Internet. Sun provides Internet capability for all machines.

Internet supports inter-communication between running programs through a series of protocols, one of which uses the IP address, also called the IP number. You must obtain an IP number from the NIC before you attach your machine to a local area network. You need to do this regardless of whether or not your site will join the actual Internet. Chapter 12 has complete information regarding who to contact at NIC and what information you need to supply to obtain an Internet number.

In addition to uucp and the Internet, your system can become a member of other wide-area networks. You can separately purchase communications software from Sun, such as X.25, that enables you to communicate with different wide area networks.

## 3.3. Entities on a Network

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

The next subsections briefly describe the hardware and software that comprise Sun networks.

### Communications Media

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

**Server**

As mentioned in Chapter 1, a machine is considered a server if it provides a network service, such as disk storage, file service, and electronic mail. Additionally, a server can be an entity such as a process or the kernel itself.

**Client**

In a Sun network environment, client workstations fall into three categories: diskless, dataless, or networked-standalone, as described in Chapter 1. In addition, the term client also applies to a SunOS process that uses resources provided by a server.

**Modem**

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

**Network Services**

Sun server machines are categorized by the services they perform, for example:

- File service, via the Network File System (NFS), and disk storage. A server that provides this service is called an *NFS server*. Furthermore, the NFS server also provides the files that diskless and dataless clients require to boot up and operate. Chapter 13, "The Sun Network File System," explains how NFS works and how to set up and maintain this service on your system.

- Yellow pages (YP) name-lookup services. A network may have a *YP master server* and, if needed, one or more *YP slave servers*. YP provides system-wide information about users, host machines, and other networks. YP also provides network-work wide security features. Chapter 14, "The Sun Yellow Pages Service," explains how to set up and maintain YP.

- Electronic Mail forwarding. *Mail servers* provide this service. Electronic mail service enables users on a network to exchange messages. Chapter 15 explains how to set up and maintain this service.

- Printer controls. This service is provided by a *print server*. Chapters 11, "Adding Hardware To Your System," and 19, "4.3BSD Line Printer Spooler," explain how to add printers to a machine, thus making it a print server.

Typically, one machine acts as NFS, YP, mail, and print server, though on large networks, this is not always the case.

## 3.4. The Concept of Administrative Domains

The *domain* is a naming technique that allows a collection of machines to be administered as a single entity. Within a domain, all named objects, such as machines or users, must have different names. However, two host machines on different domains can have the same name, just as two files in different directories can have the same name. Just as a full pathname can uniquely distinguish two files with the same basename, thus a unique hostname can be obtained by adding all or part of the domain name to the base hostname.

For example, assume company A has a domain named "compA.COM," and company B has a domain named "compB.COM." If a user at company A has a host machine called "raks," a user at company B can also have a machine called raks.

This is permissible, because when you add their domain names to the identical basename raks, these machines have the unique names

```
raks.compA.com
```

and

```
raks.compB.com
```

However, no two machines within compA.COM can have the name raks.

The domain concept has different implications for different services such as NFS, electronic mail, and YP. A YP domain is an entity that each machine must be part of in order to access a particular set of YP maps. A domain can consist of a single machine, in which case YP is not necessary. But usually YP is used to look up names within a YP domain.

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

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

# 4

# Glossary of Terms

# 4

# Glossary of Terms

**architecture**

The specific components of a computer system and the way they interact with one another. From a Sun kernel perspective, architecture refers to the type of CPU chip in the computer. A Sun-2 machine is said to have 68010 architecture, because it contains this CPU chip. A Sun-3 has 68020 architecture. A Sun-4 is considered to have SPARC architecture, because it contains a SPARC chip.

**binding**

The process during which a client finds out where a server is so that the client can receive services. NFS binding is explicitly set up by the user and remains in effect until the user terminates the bind, for example by modifying the `/etc/fstab` file. YP binding occurs when a client's request is answered by a server and is terminated when the server no longer responds.

**boot block**

The first 8 Kbyte block on a disk. It contains the disk label.

**booting**

The process of powering up the computer, testing to determine which attached hardware devices are running, and bringing the operating system kernel into memory and operation.

**bus**

A cable or circuit used for the transfer of data or electrical signals among devices.

**client (dataless)**

A machine on a network that has its own disk and individual root and `swap` partition, but relies on the NFS server for booting single and multiuser, and for other services.

**client (diskless)**

A machine on a network that does not have a disk and relies on the NFS server for file storage and other basic services.

**configuration, equipment**

The combination of CPU, peripherals, and software, and the way they are interconnected to form a system.

**controller**

An integrated circuit board that controls the operation of another device or system, such as a graphics board that controls a graphics color monitor.

**daemon**

A process that handles system-wide functions, such as network administration or line printer spooling operations.

**device**

A hardware component acting as a unit that performs a specific function, such as formatting and printing output (a printer) or reading and writing information on a disk (a disk drive). SunOS treats all devices as files.

**device driver**

A program within the kernel that controls the operation of devices. For example, the operation of the SCSI disk controller is handled by a disk device driver.

**domain**

A name given to a group of machines administered together. In YP terminology, a domain is a group of machines that access the same YP maps.

**dump**

The process of copying directories onto a tape for offline storage, using the `dump` command.

**Ethernet**

A commonly-used local area network technology originally developed by Xerox Corporation.

**export**

The process by which a server advertises the file systems that it allows hosts on a network to access.

**file system**

A hierarchical arrangement of directories and files.

**gateway**

A device that enables networks using different protocols to communicate with each other.

**global file**

A file containing information such as user, host, and network names, that is network wide in scope.

**head**

The mechanism on a disk drive that reads and writes information on a disk.

**heterogeneous server**

An NFS server that has clients of its own architecture and other architectures. For example, a Sun-4 server that has Sun-2 and Sun-3 clients is a heterogeneous server.

**homogeneous server**

An NFS server that has clients only of its own architecture.

**host**

A computer attached to a network.

**inode**

An entry in a pre-designated area of a disk that describes where a file is located on that disk, the file's size, when it was last used, and other identification information.

**interface**

A connection to a network.

**internetwork**

A group of networks interconnected with routers, which uses the IP protocol.

**Internet**

A world-wide wide area network using internet protocol (IP) that was originally sponsored by the Defense Advanced Research Project Agency (DARPA).

**kernel**

The master program set of SunOS software that manages all the physical resources of the computer, including file system management, virtual memory, reading and writing files to disks and tapes, scheduling of processes, printing, and communicating over a network.

**label**

Information written by the `format` program on sector 0 of a disk. The disk label describes the size and boundaries of the disk's partitions.

**library routine**

A series of SunOS functions that can be called by user programs written in C and other compatible programming languages.

**local file**

A file containing information specific to the machine where it resides. When using YP, the local file is checked first before a corresponding global file is checked.

**map**

A file used by the Yellow Pages service that holds information of a particular type, for example, the password entries of all users on a network or the names of all host machines on a network.

**makefile**

A file used by the `make` command, which describes files that `make` must process and programs that `make` must run.

**modem**

An electronic device that you can use to connect a Sun workstation to a telephone line.

**mount**

The process of accessing a directory from a disk attached to the machine making the mount request (4.2 mount) or remote disk on a network (NFS mount).

**netgroup**

A network-wide group of machines granted identical access to certain network resources for security and organizational reasons.

**network (local area)**

A network consisting of a number of machines that share resources such as files and mail. The network covers a physically limited area no greater than two miles.

**network (wide area)**

A network consisting of one or more large computers providing services such as file transfer, and a large number of client computers that use the services. This network may cover a large physical area, sometimes spanning the globe.

**packet**

A group of information in a fixed format that is transmitted as a unit over communications lines.

**platter**

A flat disk made of magnetic media that is mounted on a spindle. A disk such as those used by Sun computers actually is composed of a number of platters.

**page**

A standard unit of memory with an architecture-dependent size that is the smallest entity manipulated by the kernel's virtual memory system.

**partition**

A discrete portion of a disk, configured during installation, and assigned to a specific file system.

**process**

A program in operation. For example, a daemon is a system process that is always running on the system. (If it stops running, you have to start it up.)

**protocol**

A set of formal rules explaining how hardware and software on a network should interact in order to transmit information.

**pseudo-device**

Software subsystems or drivers with no associated hardware.

**remote procedure calls**

Routines that enable communication between two remote programs.

**root user name**

SunOS user name that grants special privileges to the person who logs in with that ID. If the user can supply the correct password for the root user name, he or she is given superuser privileges for the particular machine.

**router**

A device that forwards information of a certain protocol type from one network to another.

**sector**

A segment of a disk track, which, on Sun systems, holds 512 bytes of data.

**server**

A machine that provides a network service, such as disk storage and file transfer, or a program that handles such a service.

**standalone machine**

A workstation with its own disk and tape drives that does not rely on a server in order to boot.

**subnet**

A networking scheme that divides a single logical network into smaller physical networks to simplify routing.

**superblock**

A block on the disk that contains information about particular file systems, such as the file system name, size in blocks, and so on.

**superuser**

A user with special privileges granted if he or she supplies the correct password when logging in as root or using the `su` command. For example, only the superuser can change the password file and edit major system administration files in `/etc`.

**symbolic link**

A file that consists of a reference to the name of another file. The kernel translates accesses to the symbolic link into accesses to the file it refers to.

**time-sharing system**

A standalone Sun workstation with dumb terminals attached to its serial ports. The terminals rely on the workstation for processing power as well as file service and disk storage.

**track**

A concentric ring on a disk that passes under a single stationary disk head as the disk rotates.

**virtual memory**

A memory management technique used by SunOS for programs that require more space in memory than can be allotted to them. The kernel moves only pages of the program currently needed into memory, while unneeded pages remain on the disk.

# Part Two: System Administration Procedures

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

## Who Should Read This Part

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

## What Is in This Part

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

Part Two discusses the following:

- Booting up and shutting down the system.
- Using the Release 4.0 file system and understanding important system administration files.
- Performing regular maintenance, such as checking resource use, backing up and restoring file systems, and setting up the disk quota system.
- Diagnosing and fixing (non-network) problems, performing periodic maintenance and record keeping, such as crash recovery, maintaining a system log, and setting up accounting.
- Reconfiguring the kernel, including an annotated description of the GENERIC kernel configuration file for each Sun model.
- Maintaining disks with the `format` program.
- Adding hardware: boards, printers, terminals, modems.
- Adding new users and groups, setting up a local permission scheme.

# 5

# Booting Up and Shutting Down Your System

# 5

# Booting Up and Shutting Down Your System

This chapter explains what actually happens during the booting process and during system shutdown. The text is directed to advanced system administrators who need to know the booting process in detail. Topics discussed include:

- Powering up the machine from the monitor that controls the system before the kernel has taken control.
- Booting up the system through the automatic boot process.
- Using an alternative boot procedure when automatic boot has been interrupted.
- The `init` daemon and system initialization scripts.
- Stopping the operating system in a safe fashion, including steps to take if a power loss occurs while the system is operating.

If you are a beginning administrator, try to get an overall understanding of the automatic booting process. When you are more familiar with the system, review this chapter to learn booting in greater depth. You should also learn the commands needed for booting from various devices, for aborting the booting process, and for shutting down the system in an orderly fashion. Finally, pay particular attention to the remote booting process, during which an NFS server boots a client over the network. This is an important concept for administrators of any networked Sun computer to understand.

For more information about booting, refer to the *PROM User's Manual* manual in regard to the boot PROM and EEPROM, and to the man pages in regard to programs used during the booting process, particularly `boot`(8s), `installboot`(8s), and `init`(8).

## 5.1. Changes to `/boot` from Previous Releases

This section explains changes to the booting process made for Release 4.0. You should read it if you have an existing machine running earlier releases of SunOS. If your Sun computer is brand new with this release, you can skip this section if you wish.

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

☆ SEE page 50 for shutdown & reboot

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

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

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

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

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

would read /vmunix from /pub.MC680x0 on the ND server with which this client workstation was registered.

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

□ ND code has been eliminated, since servers no longer support ND operations.

□ The program /boot now understands how to perform NFS file operations over the network to a server with which the client workstation is registered.

□ Note that the boot program for a client is now contained in /tftpboot/boot.sun*x* where *x* is 2, 3, or 4, depending on the client's architecture.

## 5.2. Powering Up Self-Test Procedures

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

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

□ Critical errors are found. The screen remains dark.

□ On a Sun-2 machine, firmware checks if a keyboard is attached to the system and there is a frame buffer. If the monitor cannot find these, it sends its output to the system's serial port A.

For Sun-3 and Sun-4 machines, firmware also checks for the existence of a keyboard. However, if the keyboard cannot be found, the monitor checks the values set in the EEPROM also on the CPU board. If no EEPROM setting specifies where to receive input when no keyboard is found, the monitor then defaults to the serial port.

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

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

## 5.3. The Automatic Boot Process

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

- typing the **b** command at the monitor prompt (>)
- running the `fastboot` command from the shell (as superuser)
- running the `reboot` command from the shell (as superuser)

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

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

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

In Sun-3s and Sun-4s, this sequence of events is controlled by settings in the EEPROM. If desired, you can program the EEPROM so that the monitor automatically tries to boot over the network, skipping the SMD and SCSI stages.

### Booting from a Local Disk

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

1. It loads in a small executable called the *bootblock code* from a known location on the disk There is bootblock code at the beginning of each file system on the disk from which the machine can be booted. (The root file system partition `sd0a` or `xy0a` contains bootblock code, but sometimes others do, as well.)

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

2. The bootblock code reads in `/boot`. The bootblock code knows which blocks in that file system contain the code of `/boot` itself. This block list is placed in the bootblock code by the `installboot` program, so that the boot block code does not need to interpret the file system structure. As a consequence, any time that you change `/boot` on that file system, you must run `installboot` to reinstall the bootblock code. This is because the list of blocks occupied by `/boot` will typically change every time it is modified. For more information, refer to the `installboot`(8s) man page.

3. After it is read in, `/boot` first checks to see what device it was read in from. It then attempts to read the kernel file `/vmunix` from the same device.

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

Figure 5-1 *Autoboot Output When Booting from Disk*

```
> b                    unecessary when booting from power-up

Boot: xy(0,0,0)vmunix          xy controller
Boot: auto-mount               No arguments followed >b
root on xy0a fstype 4.2        Mount by /boot
Boot: vmunix           default kernel name

Size: 320808+80276+41728 bytes
```

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

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

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

If no other filename is provided, the file /vmunix is booted from this disk, by default from Partition A on Drive 0 on Controller 0 for this disk type. /vmunix is booted in the example above. Normally /vmunix contains a SunOs kernel, though it can instead contain any standalone program that you like, for example, a standalone diagnostic program, as long as the disk contains a standard SunOS file system that /boot can mount.

## Booting over the Network

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

### Booting a New Client over a Heterogeneous Network

When you first boot a client of a heterogeneous NFS server, you have to perform certain operations. These are described below and in *Installing the SunOS.* These procedures are necessary before normal TFTP booting can occur.

To initially boot a Sun-2 client served by a Sun-3 or Sun-4 server, do the following:

1. On the server, move the boot file boot.sun2 from /export/exec/sun2/stand to /tftpboot directory.

2. Power up the client and type the following:

```
> b ie() -a
```

   Do not use the address of the server within the parentheses following ie. This causes the boot to hang.

3. Type the following when the boot prompt appears:

```
> Boot:  vmunix
```

4. Press [RETURN] in response to the prompt for the root name.

5. Type the following in response to file system type:

```
> ???? file system type (????):  nfs
```

The Sun-2 machine should then boot up.

To boot a Sun-3 or Sun-4 client served by an NFS server of a different architecture, you use the steps above, except for Step 2. For this step, power up the client and type the following:

```
>b ie(0,server_id,0) -a
```

where *server_id* is the Internet host number of the server in hexadecimal representation. The client is a Sun-3/50 or Sun-3/60, type **`le`** instead of **`ie`**. Then proceed with Steps 3-5 above.

## The TFTP Boot Process

Here is a summary of the normal TFTP boot process, which occurs after you have initially booted a client over the network.

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

2. A RARPD server on the network recognizes the client and responds by sending the client's Internet address. This server is the one originally given the IP address for the client during `suninstall` or later, through the `setup_client` program.

3. Upon receiving its IP address, the client sends a TFTP request for its boot program from the server.

4. The server looks in its `/tftpboot` directory for a filename that is the hexadecimal representation of the client's IP address, plus a suffix representing the client's architecture, except for Sun-3s. For example, these filenames might look like:

| *Filename* | *Architecture Type* |
|---|---|
| C0095A2E | Sun-3 |
| C0095A09.SUN2 | Sun-2 |
| C0095A2B.SUN386 | Sun-386 |
| C0095A3A.SUN4 | Sun-4 |

This file is a symbolic link to the client's boot program, typically `boot.sun`*x*, where *x* is either 2, 3, or 4, depending on the client's architecture. The `boot.sun`*x* file contains the boot program for a client of architecture *x*.

6. The server then sends the appropriate /tftpboot/boot.sun program over the network to the client.

7. The client boots up using this boot program.

   Sun-2 client machines use an extra step. Instead of performing Steps 1 and 2 above, they broadcast an ND request, which is intercepted by the ndbootd(8) server program. It returns a standalone program, which carries out the same TFTP request sequence as for Sun-3s and Sun-4s. After Step 3 above, the boot process is then identical for a Sun-2 machine.

   Servers that have no entry for the client in their /tftpboot directories will wait a short while before replying, so as not to overwhelm the client with error messages.

8. After the monitor reads it in, /tftpboot/boot.sun*x* first checks to see what device it was read in from. It then attempts to read the kernel file /vmunix from the same device.

Heterogeneous Sun-3 and Sun-4 servers need to do the following before they can boot their Sun-2 clients.

1. On the server, move the boot file boot.sun2 from /export/exec/sun2/stand to /tftpboot.

2. Power up the client and type the following:

```
    > b ie() -a
```

   Do not use the address of the server within the parentheses following ie. This causes the boot to hang.

3. Type the following when the boot prompt appears:

```
   > Boot:  vmunix
```

4. Press RETURN in response to the prompt for the root name.

5. Type the following in response to file system type:

```
   > ???? file system type (????):  nfs
```

## The bootparamd Daemon and /etc/bootparams File

The bootparamd daemon and /etc/bootparams file assist client machines in finding their IP addresses and the paths of their root and swap directories. After being loaded, the client's boot.sun program broadcasts a REVARP request to find its own IP address, then broadcasts a whoami request. A bootparamd daemon running on a server responds to this request. The bootparamd daemon looks up the client's IP address and returns it to its client's hostname. In the example above, the client's IP address is 192.9.1.91 and its hostname is solntze. The client then sends a getfile request along with its hostname to the bootparams server, which should have an entry for this client in the file /etc/bootparams or in a corresponding YP map.

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

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

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

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

If /etc/bootparams has been set up correctly, then in response to the getfile request, boot.sun*x*, running on the client, receives the following information:

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

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

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

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

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

Figure 5-2 *Autoboot Output When Booting from NFS*

```
> b                              unnecessary if booting from power-up.
Boot: ie(0,0,0)                       Network Interface ie0
hostname: solntze                    From Bootparam server
domainname: sweng.sun.com            From Bootparam server
server name: estale                  From Bootparam server
server path: /export/root/solntze        From Bootparam server
root on estale:/export/root/solntze fstype nfs  Mount in /boot
Boot: vmunix                 Default kernel name

Size: 357072+83604+53036             load of /vmunix begins
```

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

## 5.4. Aborting a Booting Sequence

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

The specific abort key sequence depends on your keyboard type. For a Sun-4 keyboard, the sequence is [Stop-A]. For Sun-2 and Sun-3 keyboards, the sequence is [L1-A]. For a standard terminal keyboard (console only!) the [BREAK] key generates an abort. For a Sun Model 100U or 150U, the sequence is [SET UP-A].

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

```
Abort at xxxxxxxx
```

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

## 5.5. Booting from Specific Devices

Your Sun machine can be booted from

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

As mentioned previously, the monitor automatically attempts to boot `vmunix` from a default local disk partition or NFS file system. If you wish to boot a different file, or a file from another device than the default, you must abort the automatic boot process and then issue the appropriate version of the monitor boot command.

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

### The Standard Booting Syntax

The boot command has the syntax:

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

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

To determine which devices your monitor PROM is able to boot from, you can use the command:

```
> b ?
```

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

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

```
> b
```

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

### Booting Up in Single User Mode

When booting multiuser fails, booting single user will sometimes work. To reboot `/vmunix` from the default file system and come up in single user mode, type:

```
> b -s
```

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

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

If you have successfully booted /vmunix in single user mode, you can then bring the system up in multiuser mode by typing the CTRL key and the D key simultaneously:

```
# ^D
```

If for some reason your machine cannot access its local file system, you may have to boot a program from the network or from a tape. For example, if your local disk becomes corrupted, you can boot the format(8) disk formatting program from tape or from an NFS server, then use this copy of format to fix your disk.

### Booting from an Alternative Disk

In certain circumstances, you may wish to boot from an alternative device, primarily if the file system on your default booting device becomes corrupted. For disk drives, you can use two alternative ways to boot. If you need to obtain both /boot and /vmunix from a disk, use the following syntax:

```
> b controller(address, drive, partition)pathname args
```

The arguments in the syntax are explained below.

*controller* The disk controller which runs the specific disk: xy for the Xylogics 450 or 451 controller, xd for the Xylogics 7053 controller, or sd for the SCSI disk controller.

*address* This is a small number, such as 0 or 1, indicating the *n'th* standard controller board, or the physical address of the controller on the bus.

*drive* The unit number of the disk on the specified controller.

*partition* A number corresponding to the logical partition on the disk where the file specified by pathname can be found. Zero (0) corresponds to Partition a, 1 to Partition b, and so on.

*pathname* The name of the file to boot. This can be the name of a file or a *symbolic link* pointing to a file in the same partition.

The symbolic link cannot point to a file actually resident on another disk partition, since only the root partition is mounted by /boot. For example, you can have a link /vmunix pointing to a file /vmunix.new, but not a link /vmunix pointing to /usr/vmunix.

*args* Optional arguments that will be passed on to *pathname*.

You can use the -a (for "ask") option of boot either to boot from a non-default local file system or a non-default file, as follows:

You can also obtain /boot over the network and /vmunix from a local disk. If you specify the **-a** option, boot itself asks where you wish to boot from. If you do not specify the **-a** option, /boot tries to boot /vmunix from the same device where it is located, in this case, is the network.

```
> b  sd()-a
Boot: sd(0,0,0)vmunix -a
root filesystem type (4.2 nfs):    4.2
root device (xy%d[a-h] sd%d[a-h] ): sd0g
root on sd0g fstype 4.2
Boot: vmunix.test

Executable
Size: 380928+87344+69580 bytes
```

In this example you boot /vmunix.test from a file system /dev/sd0g, (which need have no /boot), by using the /boot file obtained from /dev/sd0a.

An extra complication arises when you obtain /vmunix from a non-default file system or a non-default file. Some user programs, notably ps and rpc.rstatd, assume that the object of the running kernel can be read from a file called /vmunix in the root file system. You can arrange this by creating a symbolic link in the root file system called /vmunix that points to the kernel that was really booted.

The following example shows how to boot using a /boot file obtained over the network and the vmunix file on your machine's local disk.

```
> b  ie()-a

Boot: ie(0,0,0)vmunix -a
Using IP Address 192.9.1.90 = C009015A.
Booting from tftp server at 192.9.1.3 = C0090103.
Downloaded 132443 bytes from tftp server.
Boot V1.0
root filesystem type ( 4.2 nfs  ): 4.2
root device (xy%d[a-h] sd%d[a-h] ): sd0a
root on sd0a fstype 4.2
Boot: vmunix

Executable
Size: 380928+87344+69580 bytes
```

Here your machine's monitor issues a TFTP request and subsequently obtains /boot from an NFS server on the network. The **-a** option causes /boot to ask from which device the root file system should be mounted, with 4.2 (disk) and NFS as alternative choices. The /boot code then lists the available block devices, xy, xd and sd, then finally mounts sd0a as the root file system. It then reads the default program /vmunix from this file system, and boots it, passing the -a option to it also.

## Booting from an NFS-Mounted Partition

If you have a dataless client machine or standalone system on a network, you may want to boot it from an NFS mounted partition. Note that booting from a local disk does *not* involve the `bootparamd` daemon. However, you can set up a machine with a local disk to boot from an NFS-mounted file system. You do this by placing an entry for the machine in the `/etc/bootparams` file, or its corresponding YP map, on the NFS or YP server. Specifically, you can obtain `/boot` from a local disk, and subsequently obtain `/vmunix` from an NFS server on the network, as shown below:

```
> b -a

Boot: xy(0,0,0)vmunix -a
root filesystem type ( 4.2 nfs  ): nfs
Requesting Internet address for 8:0:20:1:3:36
Internet address is 192.9.1.90
hostname: guppy
domainname: wseng.sun.com
root name: root
server name 'estale'
server_path '/export/root/guppy'
root on estale:/export/root/guppy fstype nfs
Boot: vmunix

Executable
Size: 380928+87344+69580 bytes
```

In this example, the monitor obtains `/boot` from the local `xy` disk. Since it was invoked with the `-a` option, `/boot` does not attempt to locate the default device (the device from which it itself was obtained). Instead, it asks what kind of file system to mount. If the response had been **`4.2`**, indicating a local file system, `/boot` would have located the default device and mounted the required local file system. Since the response was **`nfs`**, `/boot` locates a network interface, and completes the boot by mounting an NFS partition from a server. Since `-a` was used, it also asks for the name of the file to boot. The kernel also reacts to the `-a` by asking where to mount its root partition from.

Machines with local disks can also obtain `/boot` and `/vmunix` from an NFS server, even if this is not the default location for these files. To perform this type of boot, use the following syntax:

> \> **b** *controller(address, hostnumber, partition)pathname args*

The parameters in the command are explained below:

*controller* — The device abbreviation for the Ethernet controller: `ec` for a 3Com Ethernet controller, `ie` for a Sun-2, Sun-3 or Sun-4 Ethernet controller, or `le` for a LANCE Chip controller.

*address* — A small number, such as 0 or 1, indicating the *n'th* standard controller board, or the physical address of the controller on the bus.

*hostnumber* Should be zero.

*partition* Should be zero.

*pathname* Pathname of the file to boot.

*args* Optional arguments.

### Booting from Tape

You can boot a Sun computer from 1/2 inch nine-track magnetic tape, from a 1/4 inch cartridge tape controlled by a Sun 1/4-inch tape controller, or from a 1/4-inch tape controlled by a SCSI tape controller. Use the following command:

```
> b tape(controller, unit, filenum) -a
```

where

*tape* The device abbreviation for the tape controller: `mt` for 1/2-inch tape with a Tapemaster controller, `xt` for 1/2-inch tape with a Xylogics controller, `ar` for a Sun 1/4-inch tape controller, or `st` for a SCSI tape controller.

*controller* A small number, such as 0 or 1, indicating the *nth* standard magnetic tape controller in the system, or the bus address of the controller.

*unit* Specifies which tape drive on the controller is to be used.

*filenum* Specifies which file of the tape contains the boot program. (On a SunOS release tape, this is always File #0.) By convention, boot commands number the first file on the tape #0, the second #1, and so on.

### Booting an Alternate Kernel from the Default Device

To boot any file from the default device, enter:

```
> b pathname args
```

In this manner you can boot an alternate version of the kernel by supplying its name as *pathname* in the example above. This is also useful for booting standalone utility programs like `format` after your disk or network is set up.

## 5.6. The `init` Daemon and System Initialization Scripts

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

1. `init` runs `rc.boot`.

2. `rc.boot` sets the machine's name. Then, if the system is to come up multiuser, it invokes `fsck` with the preen option ( `-p`).

3. `fsck` checks the disks for inconsistencies.

4. If `fsck` does not report problems, `init` invokes `rc`. If `fsck` does detect a serious problem, `init` brings the system up in single user mode. When you press [Control-D] to leave single user mode, `rc` is invoked.

5. `rc` mounts file systems on the machine's local disks, if any (4.2 mounts). Then it passes control to `rc.local`.

6. `rc.local` starts daemons on the local machine that handle NFS, YP, and mail requests. It mounts file systems that the machine accesses over the network (NFS mounts). Finally, it returns control to `rc`.

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

8. When `rc` finishes running, the system comes up multiuser.

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

## 5.7. Shutdown Procedures

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

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

You must be superuser to run any of them.

### The `shutdown` Command

HUTDOWN & REBOOT:
shutdown -r now

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

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

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

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

### The `halt` Command

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

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

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

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

`halt` writes out information to the disks and halts the processor — no warning, no delay. It takes you back to the monitor, out of the operating system. Using `halt` instead of `shutdown` on a server when an immediate system halt is not necessary can be very disturbing to users.

### The `reboot` Command

When SunOS is running and you want to reboot, you normally use `shutdown` on a server or time-shared standalone. However, you might prefer to use `/usr/etc/reboot` to reboot a machine with only one user (diskless and dataless client, or standalone that is not time shared) or a server or time-shared standalone currently operating in single user mode.

The syntax of `reboot` is:

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

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

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

### The `fastboot` and `fasthalt` Commands

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

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

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

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

You should use these commands very carefully because they do not run `fsck` to check file system consistency

6

# The SunOS File System

# 6

# The SunOS File System

This chapter discusses the file system organization in SunOS Release 4.0. It briefly describes major directories and introduces the important system administration files that you'll learn about in the rest of this manual.

## 6.1. The Release 4.0 File Systems

The directory structure of SunOS was reorganized for Release 4.0. This was done to make it easier for a single file server to support many clients of different architectures, while retaining most of the historical UNIX and BSD structure. The philosophy behind this reorganization is described in *Installing the SunOS*. You encountered its major aspects when you ran `suninstall` and used the Disk Form.

SunOS requires the existence of two file systems: / (root) and `/usr`. The directories installed by `suninstall` on an NFS server are:

```
/ (root)
/usr
/home
/export/root
/export/swap
/export/exec
```

and on a standalone machine are:

```
/ (root)
/usr
/export
```

### Importance of Symbolic Links

The SunOS file system, like all UNIX- based file systems, is organized as a tree-structured hierarchy of directories, device nodes, symbolic links, and ordinary files. Although these files and devices could reside anywhere in the hierarchy, many previously-existing commands expect certain files and devices to reside in specific directories, and will not function properly unless they do. The file system reorganization in Release 4.0 involved moving certain files and directories, particularly executable files, from their traditional UNIX directories to other directories. Because programs (and users' programs, as well) expect to find files in their traditional directories, these directories now contain *symbolic links*, rather than the actual files.

A symbolic link is a pointer in one location that the kernel reads and uses to find a file in a different location. For example, commands such as chown and fsck have been moved from their standard locations in /etc to the directory /usr/etc. In Release 4.0, if you ask for a long listing of the /etc directory, you find the following entry for the fsck command:

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

The *l* displayed before the list of permissions (rwxrwxrwx) indicates that this is a link, not an actual file. The phrase fsck -> /usr/etc/fsck indicates that the fsck command file is actually in the directory /usr/etc.

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

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

## The root File System

The root file system (/) is the first required SunOS file system, located at the top of the hierarchal file system tree. It contains the files and directories crucial for system operation, such as the kernel binary image, a device directory, and programs used for booting. On Sun machines with disks, suninstall assigns / to Partition A of the first disk (disk 0), by default. The root directory also contains *mount points*. These are empty directories where the file systems that suninstall (and you) mount are attached to the root file system tree. Note that you can make mount points anywhere in the file system, not just in the root directory. See Chapter 14 for more information about mount points.

/ contains the following files and directories:

```
bin        home        sbin       var
boot       kadb        sys        vmunix
dev        lib         tftpboot
etc        lost+found  tmp
export     mnt         usr
```

They are described below.

/bin In traditional UNIX file systems, this directory contains user commands. However, for Release 4.0, the /bin directory is actually a symbolic link to /usr/bin, which is described in the section, "The /usr File System."

/boot On machines with a disk, this file contains a program for booting the system.

/dev This is the "device" directory, which contains all of the *device special files*, also known as *device nodes*. Among the devices listed are tape drives (for example, mt0), printers (such as lp0), disk partitions (for example, sd0a), and terminals (such as tty0). Note that the contents of /dev differ for each configuration, depending on its equipment. /dev also contains a shell script called MAKEDEV. You will use MAKEDEV to create device nodes for all Sun-supported devices that you add to your configuration, as explained in Chapter 11, "Adding Hardware to Your System."

/etc — This "et cetera" directory contains data files and subdirectories used in system administration. Because of its importance, /etc and its companion directory /usr/etc are described fully in a later section, "The /etc and /usr/etc Directories."

/export — This directory contains the directories that clients access from a server.

/home — This directory contains home directory trees for the server and its clients.

/kadb — This is the kernel debugger program.

/lib — This is a symbolic link to the directory /usr/lib, which is described in the section, "The usr File System."

/lost+found

Normally this directory is empty. However, if a file system becomes damaged, the fsck (file system check) program places in lost+found links to any files that it cannot link elsewhere in the file system in an acceptable fashion.

/mnt — This is an extra mount point that you can use to temporarily or permanently mount a file system.

/sbin — This directory contains the executable files necessary for bringing up the /usr file system at boot time: hostname, ifconfig, init, mount, and sh.

/sys — This is a symbolic link to the directory /usr/share/sys.

/tmp — This is a directory for holding temporary working files. Various SunOS utilities, such as cc, the C compiler, and ar, create temporary data files in /tmp. Users may also use /tmp as a temporary workspace. Every time the system is rebooted, the script /etc/rc removes all files in /tmp except for subdirectories.

/tftpboot — This directory contains files used to boot diskless clients over the network.

/usr — This is the mount point for the usr file system. /usr is fully described in the next section.

/var — This directory contains subdirectories with data files that tend to grow. /var/adm holds system accounting files. /var/spool contains files being processed for printing, and electronic mail, among many others. Spool directories hold files pending further processing, typically by a program other than the one that placed the file into the spool directory. For example, `/var/spool/lpd` holds files sent by the `lpr` command that are waiting to be output to the printer. `/var/spool/mail` contains users' *system mailboxes*, which contain incoming mail waiting to be read. `/var/spool/mail` can be NFS mounted from a mailbox server. `/var/spool/uucp` contains `uucp` data files and work files in transit to or from other machines. `/var/crash`, which holds an image of the contents of memory when there is a crash, and `/var/preserve`, which holds files saved by the `vi` and `ex` editors if the system should crash.

If you wish, you can use `suninstall` to assign `/var` to its own partition on the disk, thus making it a separate file system.

The `/var/tmp`. directory is similar to `tmp` in the root directory. However, `/var/tmp` is available for users as temporary work space. Unlike `tmp`, files are not cleared out of `/var/tmp` when you reboot. This directory replaces `/usr/tmp`

`/vmunix` This is the SunOS kernel.

## The `usr` File System

**Note:** If you have a standalone, the user's home directories will also be located in `/usr`.

`/usr` is the second required file system in Release 4.0. On systems with disks, `/usr` is located on Partition g by default. In addition to the information usually found in a traditional UNIX `/usr` file system, Release 4.0 `/usr` also contains executables that formerly were found under `/`. Executables that used to be in `/etc`, `/bin`, and `/lib` are, in general, now in `/usr/etc`, `/usr/bin`, and `/usr/lib`.

Many files in the `/usr` file system are executable commands, system programs, and library routines. These files are completely described in the *SunOS Reference Manual*. This text contains the complete set of SunOS reference material, informally referred to as the manual or "man" pages. It is in the docubox that came with your Release 4.0 system tapes.

The major directories in the `/usr` file system are:

```
5bin        dict        lost+found      stand
5include    etc         man             sys
5lib        games       mdec            ucb
bin         hosts       pub
boot        include     sccs
demo        lib         share
diag        local       src
```

They are described below.

`/usr/bin` This major directory contains the basic SunOS commands you use every day. In Release 4.0, `/usr/bin` contains the commands formerly found in `/bin` , such as `ls`, `cat`, and `mkdir`.

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

System V Directories

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

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

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

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

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

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

/usr/demo This directory contains graphics demonstration programs. If you want to run these programs, refer to Section 6 of the *SunOS Reference Manual* and the manual *Installing the SunOS*.

/usr/dict This directory contains English language spelling lists used by the spell(1) spelling checker.

/usr/etc This directory contains commands used for system administration and maintenance. This directory is more fully described in the next section, "The /etc and /usr/etc Directories."

/usr/games This directory contains games. See Section 6 of the *SunOS Reference Manual* and *Installing the SunOS* for instructions on installing games from the distribution tapes.

/usr/hosts This directory contains a script called MAKEHOSTS, which creates a symbolic link to the rsh command for each host in the /etc/hosts file. (/etc/hosts is introduced later in this chapter.) For example, after running MAKEHOSTS and adding /usr/hosts to your $PATH shell variable, you can just type sundial instead of typing rlogin sundial. The remaining files in /usr/hosts are the actual symbolic links for every host on the network, for example, andromeda -> /usr/ucb/rsh.

/usr/include
This directory contains all the standard "include" files or "header" files used in C programs. These files, whose names conventionally end with .h, contain definitions of useful constants and macros. Individual .h files are explained in Sections 2, 3, and 5 of the *SunOS Reference Manual.*

/usr/lib
This directory contains the files originally in /lib, which, in Release 4.0, is a symbolic link to /usr/lib. /usr/lib has become a catch-all for SunOS utility files. These include: libraries (names ending in .a,) supporting SunCore, Sunwindows, and other system functions; programs used by various system utilities (for example, lint, lex, spell, etc.); troff macros (tmac, me, ms); line printer filters; and more.

The most important files in /usr/lib, from a system administrator's perspective, are the sendmail files, which are used in the electronic mail system described in Parts Three and Four.

/usr/local
This directory is empty when you first install Release 4.0. You can use it for holding commands, programs, or other files, such as those you might obtain from third parties (like the Sun User Group) and add after installation. To access these files, the users on your system should add /usr/local (and/or /usr/local/bin) to their PATH environment variables.

/usr/lost+found
Every file system has a lost+found directory, which serves the same purpose as /lost+found in the root file system.

/usr/mdec
This directory contains boot programs and a script for installing them. You normally don't use these programs after system installation, unless you need to restore a damaged root file system.

/usr/pub
The pub directory contains files used in printing.

/usr/sccs
This directory contains commands used by sccs, the Source Code Control System. See sccs(1) and the *SCCS* tutorial in *Programming Utilities and Libraries* for more information.

/usr/share
This directory contains files that can be shared across all architectures. It holds the online man pages in the directory /usr/share/man, moved there from /usr/man. The subdirectory /usr/share/sys holds kernel object modules moved from /usr/sys. The subdirectory /usr/share/lib holds the termcap file and other files pertaining to terminals; it also holds macro packages used by the troff command. /usr/share/src/sun/suntool has

files used by suntools.

/usr/src This directory is not present on all configurations. If your machine is licensed to contain source code, it will reside in this directory.

/usr/stand This directory contains programs that must be run as standalone programs from the PROM monitor, rather than run as SunOS processes.

/usr/sys The "system" directory contains all the files necessary to build or reconfigure the kernel. Kernel configuration is explained in Chapter 9, "Reconfiguring the System Kernel."

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

## The export Directory Tree

This directory tree contains directories that the server makes available, or "exports," to its clients: These directories are:

```
/export/root
/export/swap
/export/exec/architecture-name
```

/export is present, but empty, on standalone systems.

The directories that comprise the /export directory tree are defined below:

exec This directory contains the native executables (the /usr file systems) of each Sun workstation architecture you want on your server's disk.

When you first install the server, the /usr file system for the server's architecture is placed in /usr and a symbolic link to that location is placed in /export/exec/*architecture_name*, where *architecture_name* is either sun2, sun3, or sun4, depending on the server's architecture. The /usr file systems for all other architectures are installed into /export/exec/*architecture_name*, where *architecture* is sun2, sun3, or sun4, according to the client's architecture. A heterogeneous server will have in /exec a copy of the /usr file system for each architecture it supports. A homogeneous server will only have a symbolic link to /usr in its /export/exec directory.

root This directory contains the root directories for all clients of a server. Each client machine has its own root, with the same files as in the server's root. Administrators of diskless and dataless clients can change files in their machines' root directories, to affect their local environments.

`swap` This directory contains the individual swap areas for each client on a server.

### The `home` Directory Tree

This file system contains the home directories for users.

### The `/etc` and `/usr/etc` Directories

The `/etc` and `/usr/etc` directories contain most of the files, commands, and subdirectories that you need for system administration. `/etc` contains the major files that make up the administrative databases of your system. Note that it no longer actually contains the commands used for system administration; symbolic links exist in place of each command. `/usr/etc` contains the system administration commands that previously were in `/etc`. You use these commands for performing basic administrative tasks, such as shutting down the system and backing up files. This directory also contains most daemon programs.

The files in `/etc` are listed below.

```
adm->
aliases
aliases.dir
aliases.pag
bootparams
chown->
chroot->
clri->
config->
cron->
dkinfo->
dmesg->
dump->
dumpdates->
ethers->
exports->
fastboot->
fasthalt->
format.dat
fsck->
fstab
gettytab
group
halt->
hosts
hosts.equiv
ifconfig->
inetd.conf
link->
magic->
mkfs->
mknod
motd->
mount->
mtab
ncheck->
netgroup
netid
networks
newfs->
passwd
preserve->
printcap
protocols
pstat->
publickey
rc
rc.boot
rc.local
rdump->
reboot->
remote
renice
restore->
rmt->
rmtab->
rpc
rrestore->
sendmail.cf->
services
shutdown->
sm
sm.bak
state
syslog.conf
syslog.pid
termcap->
tmp->
ttys
ttytab
umount->
unlink->
update
utmp
vipw->
xtab->
yp
```

Files followed by the characters "->" are symbolic links to other locations, principally under `/usr/etc` and `/var`. The exception is:

| File | New Location |
|---|---|
| termcap | /usr/share/lib/termcap |

The section, "Major System Administration Files," later in this chapter introduces many files in `/etc`. You can also find reference material about them in Chapter 5 of the *SunOS Reference Manual*

The following files are in `/usr/etc`:

```
ac
accton
arp
audit
auditd
biod
catman
chown
chroot
clri
config
cron
dcheck
devnm
dkinfo
dmesg
dump
dumpfs
edquota
eeprom
etherfind
exportfs
extract_release
extract_unbundled
fastboot
fasthalt
format
fpa
fsck
fsck-
fsirand
gettable
getty
gpconfig
grpck
halt
htable
icheck
ifconfig
implog
implogd
in.comsat
in.fingerd
in.ftpd
in.named
in.rexecd
in.rlogind
in.routed
in.rshd
in.rwhod
in.talkd
in.telnetd
in.tftpd
in.timed
in.tnamed
inetd
init
keyenvoy
keyserv
kgmon
link
lpc
mailstats
mc68881version
mconnect
mkfile
mkfs
mknod
mkproto
mount
ncheck
ndbootd
newfs
nfsd
nfsstat
nstest
pac
ping
portmap
praudit
pstat
pwck
quot
quotacheck
quotaoff
quotaon
rarpd
rdump->
reboot
renice
repquota
restore
rmt
route
rpc.bootparamd
rpc.etherd
rpc.lockd
rpc.mountd
rpc.rexd
rpc.rquotad
rpc.rstatd
rpc.rusersd
rpc.rwalld
rpc.sprayd
rpc.statd
rpc.yppasswdd
rpc.ypupdated
rpcinfo
rrestore->
rwall
sa
savecore
setup
setup.files
showmount
shutdown
spray
swapon
syslogd
termcap->
trpt
tunefs
umount
unlink
update
upgrade
vipw
yp
ypbind
ypserv
```

Three files are symbolic links:

| File | New Location |
|---|---|
| rdump | /usr/etc/dump |
| rrestore | /usr/etc/restore |
| termcap | /usr/share/lib/termcap |

/usr/etc only contains executable files, which are described in greater detail throughout this manual and in the man pages. It does have a few subdirectories, the most significant of which is yp. This directory contains commands for setting up and maintaining the yellow pages (YP) service on your configuration. (YP is explained in more detail in Chapter 14, "The Sun Yellow Pages Services.")

## 6.2. Major System Administration Files

As system administrator, you will constantly update or refer to a series of system administration files in order to keep your configuration running smoothly. Most, though not all, of these files are in /etc . Some are significant to you only if you are administering a network server. Others are important for all configurations: servers, standalones, dataless, and diskless clients.

This next section introduces the major administrative files, organized according to the configuration type(s) that typically uses them. Refer to the section that applies to your configuration for information about the file you need to learn about.

### Files for Administering Diskless Clients

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

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

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

They are introduced below.

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

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

### crontab

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

Use crontab for scheduling commands, shell scripts, and other programs that you want to run at regular intervals, such as every day, once a week, once an hour, and so forth. The /usr/etc/cron daemon consults the list of commands to be run in a crontab file and makes sure they are executed at the appropriate times.

You can create or edit a crontab file using the crontab -e command. Do not directly edit the system /var/spool/cron/crontabs/root file.

Here is a typical crontab file:

```
0 0 * * * calendar -
15 3 * * * find / -name .nfs -mtime +7 -exec rm -f {} ; -o -fstype nfs -prune
15 4 * * * find /var/preserve/ -mtime +7 -a -exec rm -f {} ;
30 3 * * * sh /usr/local/GETNAMES; sh /usr/local/lib/distnameslist
10 2 * * * sh /etc/yp/doaliases; cd /etc/yp; make
0 18 * * * cd /etc/yp; make bootparams ethers hosts netgroup passwd; date >/var/tmp/yppushed
20 1 * * * sh /usr/local/update.h
```

Each crontab entry has six fields, which you can edit using your text editor.

Refer to Chapter 8, "Special Maintenance," and the crontab(5) and cron(8) man pages in the *SunOS Reference Manual* for more information.

### fstab

The /etc/fstab file shows all file systems that your client machine mounts when it boots. Every time you boot your system, fstab is read by commands that mount and unmount file systems, and check file system consistency. It is also read by the system when providing additional swap space. Here is an example of a client's fstab file.

```
raks:/export/root/stefania / nfs rw,hard,bg 0 0
raks:/export/swap/stefania / nfs rw,hard,bg 0 0
raks:/usr /usr nfs ro,soft,bg 0 0
raks:/home/raks /home/raks nfs rw,hard,bg 0 0
```

suninstall creates a default fstab file for each machine on a network. To have your client mount additional file systems upon boot up, you must log in as superuser, and, using your text editor, add these file systems to fstab. The fstab file is more fully described in Chapter 13, "The Sun Network File System," and in the mntent(5) man page in the "SunOS Reference Guide."

### passwd

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

```
root:jWDWag4CibZJE:0:1:Operator:/:/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
uucp:*:4:4::/var/spool/uucppublic:
news:*:6:6::/var/spool/news:/bin/csh
sync::1:1::/:/bin/sync
stefania:ZRVQ6OQsLUYdM:3747:20:Stephanie:/home/raks/stefania:/bin/csh
```

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

The /etc/passwd file is your system's first line of security. Its fields are explained fully in in the passwd(5) man page. The YP password map and how it affects access to machines on a network is explained in Chapter 14.

### group

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

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

```
wheel:*:0:
daemon:*:1:
bin:*:3:
operator:*:5:operator
news:*:6:
syslog:*:8:karen,bill,bnow
bundled:*:12:rog,bill,bob,bb,dan,gale,judy,mat,nat,sh,sp,smb,tom,tommy,yvonne
```

If you are on a network running YP, placing a plus sign (+) at the end of /etc/group will include the YP group categories on your system.

Like /etc/passwd, /etc/group is another security mechanism. Its fields and its significance to a network client are explained fully in Chapter 8 and in the group(5) man page. YP group policies are discussed in Chapter 14.

### hosts.equiv

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

A typical hosts.equiv file has the following structure:

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

Chapter 12 explains how hosts.equiv helps to keep your configuration secure in a network environment. Also refer to the hosts.equiv(5) man page for more information.

### .rhosts

The .rhosts file, which is in your home directory, is used for additional permission checking when trying to access a remote machine. It has the same format as /etc/hosts.equiv, that is

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

Chapter 12 contains more information about .rhosts.

### aliases

The /etc/aliases file is used by the sendmail mail routing program. You use /etc/aliases to give sendmail at least one alternative name, or *alias*, for a user, which sendmail then recognize as having the same machine address as the user's official name. Such an entry would look like:

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

where the first entry,

```
jgoodyear: amina@raks
```

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

```
amina: jgoodyear
```

tells sendmail that the address jgoodyear is to be rewritten as amina@raks.

You can also use /etc/aliases to create a mail distribution list for a group of people. For example,

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

where samba is the name of a distribution list and the names following are the machine names to receive all mail addressed to "samba."

Note that you have to use the newaliases command in order for these changes to take affect. Chapter 15, "Electronic Mail and Communications," explains /etc/aliases and sendmail in detail, as does the aliases(5) man page.

### printcap

The /etc/printcap file is a database describing the types of printers used by your machine. It is refered to by the printer spooling system. You may attach a printer directly to your workstation or send your files to printers elsewhere on your network.

Below is a printcap file that includes four remote printers:

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

You will want to modify the printcap file to describe the printer attached to your workstation, to add additional printers on your network, and to indicate your default printer. Chapter 11 explains how to administer printers, including how to modify /etc/printcap. Also refer to the printcap(5) man page for more information.

### Files for Administering a Dataless Client

As administrator of a dataless client, you have some files on your local disk but rely on the server for your / and /usr file systems. In most respects, your configuration functions on the network similarly to a diskless client. Therefore, you should learn about the files described above for administering a diskless client:

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

However, because you probably will be responsible for backing up the files on your local disk, you should also understand the /etc/dumpdates file described in the subsection, "Files for Administering a Server."

### Files for Administering a Standalone System

Your standalone configuration has its / and /usr file systems on the local disk. Therefore, if it has a local tape drive, your machine may function as a single user standalone or as a time-sharing system with terminals attached. Depending on your needs, you can attach either of these standalone configurations to a network.

If you administer a non-networked standalone configuration, you should learn about the following files, which were previously described in the subsection "Files for Administering a Diskless Client."

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

You should also learn about the following files, especially if you have a time-sharing standalone.

| | |
|---|---|
| /etc/dumpdates | (for backing up files) |
| /etc/ttytab | (if you plan to attach terminals) |
| /etc/aliases | |

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

If your standalone configuration is on a network, also learn about the following files:

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

These were described previously in "Files for Administering a Diskless Client." You should also learn about the exports file, which is described in the next subsection.

## Files for Administering a Server

As system administrator of a server, you need to learn about *local* files that pertain specifically to your machine, such as:

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

These files are described above in "Administering a Diskless Client." Other files that affect your system locally are

| | |
|---|---|
| `dumpdates` | (for backing up files) |
| `ttytab` | (if you plan to attach a terminal) |

These are described in this subsection.

You also need to learn about files for managing machines and file systems over the network. These files are:

```
bootparams
ethers
exports
fstab
hosts
netgroup
networks
```

They also are introduced in this subsection.

### dumpdates

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

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

/etc/dumpdates is fully described in the next chapter, "Regular File System Maintenance."

### ttytab

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

```
console "/usr/etc/getty Console-9600"  sun     on secure
ttya    "/usr/etc/getty std.9600"       unknown         off secure
ttyb    "/usr/etc/getty std.9600"       unknown         off secure
ttym0   "/usr/etc/getty std.9600"       unknown         off secure
ttym1   "/usr/etc/getty std.9600"       unknown         off secure
ttym2   "/usr/etc/getty std.9600"       unknown         off secure
ttym3   "/usr/etc/getty std.9600"       unknown         off secure
ttym4   "/usr/etc/getty std.9600"       unknown         off secure
ttym5   "/usr/etc/getty std.9600"       unknown         off secure
ttym6   "/usr/etc/getty std.9600"       unknown         off secure
ttym7   "/usr/etc/getty std.9600"       unknown         off secure
ttyp0   none                    network         off secure
ttyp1   none                    network         off secure
ttyp2   none                    network         off secure
ttyp3   none                    network         off secure
```

ttytab is fully explained in Chapter 11. Also refer to the ttytab(5) man page in the *SunOS Reference Manual*

### bootparams

The /etc/bootparams file is a database maintained on the server that clients use during booting. Here is a fragment of a typical /etc/bootparams file.

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

The file shows the directories that clients samba and ballet need to mount from NFS server dancers in order to boot: root and swap. For more information about /etc/bootparams, refer to Chapter 5.

### exports

The /etc/exports file lists the file systems that your server makes available to host machines on its network. You modify it by using a text editor. When you add a new client machine, /etc/exports is automatically updated.

Here is a fragment of a typical /etc/exports file:

```
/       -access=engineering
/usr    -access=engineering
/home   -access=engineering:snail
#
/export/root/raks -root=raks,access=raks
/export/swap/raks -root=raks,access=raks
/export/dump/raks -root=raks
#
/export/root/samba -root=samba,access=samba
/export/swaps/samba -root=samba,access=samba
/export/dump/samba -root=samba
```

The file lists the names of exported file systems and machine names or netgroup names allowed access to them. A *netgroup* is a network-wide group of users allowed access to certain files for security and organizational reasons. /etc/exports is fully described in Chapter 13. Also refer to the man page exports(5) for more information.

### hosts

The /etc/hosts file lists the names and addresses of the host machines on the network that the server knows about.

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

```
# If the yellow pages is running, this file is only consulted
# when booting
#
# These lines added by the Suninstall Program
#
192.9.200.1        ballet
192.9.200.2        raks
192.9.200.3        samba

192.9.200.100      dancer

127.0.0.1 localhost loghost
#
# End of lines added by the Suninstall Program
#
```

The numbers in the first column are the Internet numbers for each host. The next column contains the machine names of the hosts, and shows which, if any, hosts are also servers. The third and subsequent columns contain nicknames for the host. Also, if the host runs YP, its /etc/hosts file is used only during the booting process until YP comes into operation.

Chapter 12 explains /etc/hosts in detail, as does the hosts(5) man page.

### fstab

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

```
   /dev/xd0a / 4.2 rw,nosuid 1 1
   /dev/xd0h /usr 4.2 rw 1 2
   /dev/xd4c /home 4.2 rw 1 3
   /dev/xd0g /export/dump 4.2 rw 1 4
   /dev/xd0e /export/swap 4.2 rw 1 5
   /dev/xd0f /export/root 4.2 rw 1 6
```

Note that the default fstab file on a server lists what are called 4.2 or UFS mounts--directories that are mounted in standard BSD fashion, directly from the disk. Diskless client's file systems are NFS mounted over the network. Single-user machines that have disks—dataless clients and networked standalones—will contain both 4.2 and NFS mounts in their /etc/fstab files. The fstab file is more fully described in Chapter 13 and in the mntent(5) man page in the *SunOS Reference Manual.*

### ethers

The /etc/ethers file is a database containing the Ethernet address of hosts on the network. Below is a fragment from an /etc/ethers file

```
8:0:20:1:d5:bb  alexandria
8:0:20:1:29:7   athens
8:0:20:1:5e:e3  bagdad
8:0:20:1:1f:73  carthage
8:0:20:1:51:a1  constantinople
8:0:20:1:36:3   jerusalem
8:0:20:1:70:ad  rome
```

The first column contains the Ethernet address. The second column is the machine name. /etc/ethers is not consulted if you run YP. Refer to Chapter 12 and the ethers(5) man page for more information.

### networks

The /etc/networks file lists the names and addresses of the networks that are part of your larger network. Here is a fragment of an /etc/networks file.

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

Refer to Chapter 12 and the networks(5) man page for more information.

### netgroup

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

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

Refer to Chapter 14 and the netgroup man page for more information.

7

# Regular Maintenance

# 7

# Regular Maintenance

This chapter explains two major maintenance tasks that you need to perform on a regular basis:

- Backing up file systems (and restoring files on an as-needed basis).
- Checking disk usage to ensure that file systems aren't exceeding allocated space on the disk.

You should develop a schedule for these procedures that is appropriate for your type of configuration and your site's requirements.

## 7.1. Backing Up File Systems

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

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

- SunOS file systems that you should back up
- Planning a backup strategy
- Tapes and tape controllers used for backup
- The `dump` command syntax
- How to dump various system configurations

### File Systems That Should Be Backed Up

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

SunOS provides two commands for backing up files: `tar` and `dump`. You used `tar` when you loaded the 4.0 operating system. However, because `tar` performs backup on a file-by-file basis, it is not recommended for regular back ups. `dump` backs up files on a per-file-system basis, as you will see in the subsection discussing `dump` syntax. Therefore, when determining your backup strategy, you need to determine which file systems should be dumped and on what schedule.

### Who Needs to Back Up?

You have to perform dumps if you administer either an NFS server or a standalone system. If you administer a diskless or dataless client, you typically rely on the administrator of the NFS server to dump your file systems

### File Systems to Dump on a Standalone

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

```
/           Partition a
/usr        Partition g
```

The / file system on a standalone contains your SunOS kernel and other important files. It also contains the directory /var/spool/mail, where mail files are kept. Therefore, you will want to back it up at regular intervals, especially if your standalone is on a network.

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

During suninstall, you may have assigned additional file systems relevant to your site to other available partitions. Be aware of the partitions where your file systems are located, because you specify the file system to the dump command by its partition location.

### File Systems to Dump on a Server

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

These file systems are, by default:

```
/                 Partition a
/usr              Partition g
/home             Partition h
/export/root      Partition d
```

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

The diskless and dataless clients' individual root directories are kept in the /export/root file system. Because they contain the same information as the server's root directory in Partition a, they don't change too frequently. If your site sends mail to the clients' root directories, you should back up /export/root more frequently.

The file system you need to back up most frequently is /home. Because /home contains the home directories and subdirectories of all clients on the system, its files are very volatile. Depending on your site, you should back up /home frequently—at the very least, once a week.

## Planning a Dump Strategy

The dump command lets you do two kinds of dumps: full dumps and incremental dumps. dump provides a system of *levels*, through which you can specify whether all files should be dumped or only files that have been altered since a dump of a lower level. The dump levels are 0-9. When you specify Level 0 to the dump command, it does a *full dump*—a backup of the contents of an entire file system. When you specify Levels 1-9 to dump, it does an *incremental dump*—a backup only of files that have changed since the last dump of a lower level. (You will see how to actually specify dump levels in the subsection, "The dump Command Syntax.")

Plan your dump strategy on paper. You should consider the following items:

- Which file systems need to be dumped for your particular configuration (standalone or server)?
- How often should you back up? This depends on how many people use your configuration and the type of work that they do. If users constantly create and modify files, consider making daily dumps of /home on a server or /usr on a standalone.
- What levels of incremental dumps are necessary so that you can recover files in case a user deletes them or the system crashes? Sample dump levels are described in the next subsections.
- How many tapes are needed to implement your backup scheme? If cost-effectiveness is a consideration at your site, you might want to devise a strategy that efficiently uses a smaller number of tapes.

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

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

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

- It is recommended that you save a week's worth of daily, level 9 dumps for at least one week after they have been made. This will require at least four tapes.
- You should probably do a level 0 dump of your root file system once a week to once a month, depending on your site's requirements.

Remember to tailor your strategy for your configuration. Then, and most importantly, you must implement a schedule and make sure the dump tapes are made. In addition, make sure stored tapes are kept cool and clean and are well labeled. Tapes that cannot be read due to deterioration or lack of a label are useless.

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

### Backup Strategies for a Server

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

Table 7-1 *Schedule of Backups for a Server*

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

With this scheme, you use *n* tapes (the number of tapes needed for a full backup of /, /usr, /export/root, and /home) plus eight additional tapes for the incremental dumps of /home.

Here is how such a scheme works:

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

dump— Friday's Level 5 dump.

5. Do the Wednesday and Thursday Level 9 dumps onto Tapes D and E.

6. At the end of the week, use Tape F for a Level 5 dump of /home. This tape will contain all changes made to files since the Level 0 dump, approximately a week's worth of changes. Save this tape until the second Friday of the next month, when you will use it again.

7. Repeat Steps 3-5 for the next week, and so on until the end of the month, when you once again take full dumps of all file systems.

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

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

Table 7-2 *An Alternative Backup Schedule for a Server*

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

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

## Backup Strategies for a Standalone

As a system administrator of a standalone, you can use a strategy similar to that pictured in Table 7-1 for the server. Depending on system usage, you may not want to back up as frequently as recommended in Table 7-2. Remember that `/usr` contains the SunOS commands as well as the users' home directories. Therefore, a Level 0 dump of `/usr` will be large. However, the incremental dumps will only include changes made to home directories, since you probably won't change SunOS files in `/usr` that frequently.

If your standalone is on a network, you might also want to back up / more frequently, perhaps once a week. This is because / includes `/var/spool/mail`, the directory that holds electronic mail.

Table 7-3 *Schedule of Backups for a Standalone*

| | | | |
|---|---|---|---|
| / | end-of-month | Level 0 | *n* tapes |
| /usr | end-of-month | Level 0 | " |
| /usr | 1st Friday | Level 5 | Tape A |
| " | 1st Monday | Level 9 | Tape B |
| " | 1st Tuesday | Level 9 | Tape C |
| " | 1st Wednesday | Level 9 | Tape D |
| " | 1st Thursday | Level 9 | Tape E |
| /usr | 2nd Friday | Level 5 | Tape F |
| " | 2nd Monday | Level 9 | Tape B |
| " | 2nd Tuesday | Level 9 | Tape C |
| " | 2nd Wednesday | Level 9 | Tape D |
| " | 2nd Thursday | Level 9 | Tape E |
| /usr | 3rd Friday | Level 5 | Tape G |
| " | 3rd Monday | Level 9 | Tape B |
| " | 3rd Tuesday | Level 9 | Tape C |
| " | 3rd Wednesday | Level 9 | Tape D |
| " | 3rd Thursday | Level 9 | Tape E |
| /usr | 4th Friday | Level 5 | Tape H |
| " | 4th Monday | Level 9 | Tape B |
| " | 4th Tuesday | Level 9 | Tape C |
| " | 4th Wednesday | Level 9 | Tape D |
| " | 4th Thursday | Level 9 | Tape E |

With this scheme, you use *n* tapes (the number of tapes needed for a full backup of / and `/usr`) plus eight additional tapes for the incremental dumps of `/usr`.

Here is how such a scheme works:

1. At the end of the month, you do a full backup (Level 0) of / and `/usr.` and save these tapes for a month.
2. On the first Friday of the month, you do a Level 5 dump of `/usr`, which copies all files changed since the previous lower level dump, in this case the Level 0 dump you did at the end of the month. You use Tape A for this

dump, store it for a month, then use it the first Friday of the next month.

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

### Tapes and Equipment Used for Backup

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

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

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

Table 7-4 *Sun Tape Controllers*

| *Tape Controller* | *Device Abbrev* | *Width* |
|---|---|---|
| Xylogics | `mt` | 1/2 inch |
| Tapemaster | `mt` | 1/2 inch |
| SCSI | `st` | 1/4 inch |
| Archive | `ar` | 1/4 inch |

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

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

### Specifications for 1/2 Inch Tapes

Devices with 1/2 inch tapes should always have the abbreviation rmt*n* or nrmt*n*, whether you use a Xylogics or Tapemaster controller. Here is what the letters in rmt*n* mean:

r — This abbreviation tells dump to use the raw device. The term *raw* means, consider the tape as a single unit, not as a series of physically blocks, or a block device. You should always specify the raw device when using dump.

mt — This tells dump that you are using reel-to-reel tape.

*n* — This represents an actual number, such as 0, 8, and so on. The number you supply depends on the type of tape device you have and whether or not you want the tape to rewind when files are closed.

When planning your backup, you may decide to store more than one file system on a reel of tape. If so, you need to consider how much data your tape will hold and compare that to the size of the file systems you want to dump. When you copy more than one file system onto a tape, you need to use dump's no rewind option, as described in the next subsection. The tables below show tape format in terms of bpi, which means bits-per-inch of data that you can store on the tape. 1600 bits per inch is considered a *low density* format. A 2400 foot tape at 1600 bpi holds approximately 30 Mbytes of information. 6250 bits-per-inch is considered a *high density* format. A 2400 foot tape at 6250 bits-per-inch holds approximately 150 megabytes of information.

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

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

Table 7-5 *Specifications for a Xylogics Tape Controller with Fujitsu Tape Unit*

| *Device Abbrev.* | *Format* | *Capacity* |
|---|---|---|
| rmt0 | 1600 bpi (rewind) | 30 Mbytes |
| nrmt0 | 1600 bpi (no rewind) | 30 Mbytes |
| rmt8 | 6250 bpi (rewind) | 140-150 Mbytes |
| nrmt8 | 6250 bpi (no rewind) | 140-150 Mbytes |

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

| *Device Abbrev* | *Tape Length* | *Capacity* |
|---|---|---|
| rmt0 | 1600 bpi (rewind) | 35-40 Mbytes |
| nrmt0 | 1600 bpi (no rewind) | 35-40 Mbytes |

## Specifications for 1/4 Inch Tapes

Two types of tape controllers are available on Sun equipment: various models of SCSI tape controllers and Archive controllers, which are used on older Sun-2 workstations. Device abbreviations for 1/4 inch tape drives are either `rst`*n* and `rar`*n*, or `nrst`*n* and `nrar`*n*. Here is what the letters in these abbreviations mean:

`r` This character tells `dump` to use the raw device. The term *raw* means, consider the tape as a single unit, not as a series of physically blocks, or a block device. You should always specify the raw device when using `dump`.

`st` or `ar` This is the abbreviation indicates tape device, either `st` for a tape device on the SCSI controller or `ar` for a tape device on the Archive controller.

*n* This represents an actual number, such as 0, 8, representing the QIC format of the tape. QIC-11 means the tape is in four-track format. QIC-24 means the tape is in nine-track format. The table below fives the calue of `n` that corresponds to each of these formats.

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

Tapes are formatted in what is called QIC format standard; on Sun equipment, tape drives are either in QIC-11 or QIC-24 format. Usually QIC-11 capable tape drives are 4 track and QIC-24 capable drives are 9 track, but that is not always the case. The tape format, equipment used, tape length, and number of tracks determine how many megabytes of information you can store on the tape. The table below depicts this information.

Table 7-7 *Devices and Tapes on a SCSI Tape Controller*

| *Device Abbrev* | *Description* | *Format* | *Tracks* | *Tape Length* | *Capacity* |
|---|---|---|---|---|---|
| `rst0` | Sun-2 Multibus | QIC-11 | 4 | 450 ft | 20 Mbytes |
| `rst8` | Sun-2 Multibus | QIC-24* | 4 | 450 ft | 20 Mbytes |
| `rst0` | Sun-2 VME, Sun-3, Sun-4 | QIC-11 | 9 | 450 ft | 45 Mbytes |
| `rst0` | Sun-2 VME, Sun-3, Sun-4 | QIC-11 | 9 | 600 ft | 60 Mbytes |
| `rst8` | Sun-2 VME, Sun-3, Sun-4 | QIC-24 | 9 | 450 ft | 45 Mbytes |
| `rst8` | Sun-2 VME, Sun-3, Sun-4 | QIC-24 | 9 | 600 ft | 60 Mbytes |

Table 7-8 *Devices and Tapes on an Archive Controller*

| *Device Abrev* | *Description* | *Format* | *Tracks* | *Tape Length* | *Capacity* |
|---|---|---|---|---|---|
| `rar0` | Sun-2 Multibus | QIC-11 | 4 | 450 ft | 20 Mbytes |

* Not all Sun-2 Multibus SCSI tape hardware is capable of QIC-24 operation.

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

## Using the dump Command

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

The syntax of dump is:

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

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

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

0-9 Dump level you want to use.

c Dump to cartridge tape. (The default is reel-to-reel.)

d Tape density. The default is 1600 bpi for a 1/2 inch tape and 1000 bpi for a 1/4 inch tape.

f Dump file system to the device indicated later on the command line.

s Tells dump the size of the tape.

u Write the date the dump was successfully completed to the file /etc/dumpdates. (/etc/dumpdates is described in the section "Restoring Files.")

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

```
    /dev/rmt8
```

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

```
    /dev/nrst8
```

to indicate "no rewind." The advantage of specifying no rewind is that you can copy more than one file system onto the tape during a backup procedure. From the arguments you supply, dump automatically calculates how much data it can put onto the tape.

Suppose you load a new tape, then issue a dump command. This starts the copy of the file system at the beginning of the tape. If the file system is too large to fit on one tape, dump will wait for you to load another tape, then continue the copying process. However, if you select no rewind," dump does not prompt you to load a second tape when a tape ends in the middle of a file system. Therefore, before dumping multiple file systems onto one tape of any type, you must calculate the sum, in megabytes, of all the file systems to be dumped. Then, organize the order that you issue commands so that the file systems being dumped will fit on the same tape. You can get the size, in kilobytes, of the file system(s) you plan to dump by using the df(1) command.

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

Table 7-9 *Sun Disk Controllers*

| *Disk controller* | *Abbreviation* |
|---|---|
| Xylogics 7053 | xd |
| Xylogics 450 & 451 | xy |
| SCSI Disk | sd |

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

```
/dev/rxd0d
```

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

```
/dev/rsd0g
```

## Procedures for Dumping Your Configurations

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

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

□ Log in as superuser, then run shutdown, as described in Chapter 5, "Booting Up and Shutting Down Your System." Then, reboot and come up single user, as shown below:

```
# /usr/etc/shutdown

Various prompts from shutdown appear

>b -s
```

**Note:** With Release 4.0, clients no longer have to shut down their systems while you use the NFS server for dumps. They just will not have access to whatever file system you are currently backing up.

If you have a *server*, you might prefer to use this method, because `shutdown` gives you the option to send messages regarding the imminent shutdown of services to others on the network.

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

```
# kill 1
```

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

2. If you are doing a Level 0 dump on any file system, you should, at single user level, run **`/usr/etc/fsck`**, the file system checking program. `fsck` checks the file system for integrity and readability, and makes minor repairs. This action ensures that your full dump will be reasonably clean.

The following subsections are divided into:

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

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

## Procedures for Dumping to Half Inch Tapes

The next instructions are actually examples for dumping files to a local 1/2 inch tape unit. The examples assume that you are backing up an NFS server with a 1/2 inch tape unit (`mt8`) that uses high density, 6250 bits-per-inch tape, and has a disk attached to a Xylogics (`xy`) controller unit. (If your NFS server has a different disk controller or tape unit, simply substitute the appropriate device abbreviations for those shown in in the examples.) The examples also assumes that your tape has a length of 2400 feet. They also assum that `/`, `/usr`, `/home`, and `/export/root` are on the default disk partitions.

### A Sample Full Dump

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

1. While the system is in single user mode (and you, of course, are logged in as superuser), load the tape into the tape device.

2. Dump the root file system by typing

```
# /usr/etc/dump 0uf /dev/nrmt8 /dev/rxy0a
```

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

0uf The *0* specifies a full, Level 0 dump; the *u* tells dump to update the /etc/dumpdates file; the f tells dump to write to a device to be named as the next arguement on the command line.

/dev/nrmt8 Gives dump information about the device it is dumping to. /dev tells dump that it should write to a special (device) file with the following name, in this case, nrmt8. The actual name nrmt8 gives instructions about the tape unit. *n* indicates "use the no rewind option." In this way, dump will not rewind the tape after it is finished copying the file system, so you can copy another file system onto the tape. *r* says treat this unit as a raw device. *mt8* indicates that the device is a reel-to-reel, magnetic tape unit using high density, 6250 bpi tape.

/dev/rxy0a Gives dump information about the file system you want to copy. /dev tells dump that the file system to copy is in a special (device) file whose name follows, in this case, rxy0a. The actual name rxy0a gives information about the device where the file system is located. *r* says treat this unit as a raw device. *xy* indicates that the device is attached to a Xylogics 450 or 451 disk controller. *0a* indicates that the file system is on the first disk (disk 0) on that controller, and is in Partition a, the usual location for a server's / file system.

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

```
# /usr/etc/dump 0uf /dev/nrmt8 /dev/rxy0g
```

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

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

```
# /usr/etc/dump 0uf /dev/nrmt8 /dev/rxy0d
```

The only difference between this and the previous commands is in *rxy0d*, which tells dump to back up Partition d, which usually holds the /export/root file system.

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

```
# /usr/etc/dump 0uf /dev/nrmt8 /dev/rxy0h
```

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

6. Type the following:

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

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

### A Sample Incremental Dump

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

1. While the system is in single user mode (and you, of course, are logged in as superuser), load the tape into the tape device.

2. Dump the /home file system by typing:

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

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

3. Rewind the tape by typing:

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

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

## Procedures for Dumping to Quarter-Inch Tapes

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

Note that since you are using cartridge tape, you have to specify the blocking factor option to the dump command. Tapes are divided into areas that hold data, separated by tape gaps—blank areas about 3/4 inch long that the tape drive uses for alignment. The area for holding data is divided into smaller segments called *physical blocks*, which hold 512 bytes of data each. The *blocking factor* is the number of physical blocks that occur between the tape gaps for this particular type of tape. The default blocking factor of 126 is appropriate for 1/4 inch tapes.

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

### A Sample Full Dump

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

1. While the system is in single user mode (and you, of course, are logged in as superuser), load the tape into the tape device.

2. Dump the /usr file system first, because, on a standalone, this is the largest file system and may exceed more than one tape. Type

```
# /usr/etc/dump 0ucf /dev/nrst8 /dev/rsd0g
```

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

0ucf — The *0* specifies a full, Level 0 dump; the u tells dump to update the /etc/dumpdates file. The *c* indicates that you are using cartridge tape. The *f* tells dump to write to the device to be named later on the command line.

/dev/nrst8 — Gives dump information about the device it is dumping to. /dev tells dump that it should write to a special (device) file with the following name, in this case, nrst8. *n* indicates "use the no rewind option." Note that /usr on a standalone may exceed one tape cartridge. *r* says treat this unit as a raw device. *st8* indicates that the device is a cartridge tape unit using nine-track tape attached to a SCSI tape controller.

/dev/rsd0g — Gives dump information about the file system you want to copy. /dev tells dump that the file system to copy is in a special (device) file whose name follows, in this case, rsd0g. *r* says treat this unit as a raw device. *sd* indicates that the device is attached to a SCSI disk controller. *0g* indicates that the file system is on the first disk (disk 0) on that controller, and is in Partition g, the usual location for /usr.

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

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

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

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

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

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

*If your system is an NFS server,*
continue with the following steps.

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

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

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

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

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

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

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

### A Sample Incremental Dump

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

1. While the system is in single user mode (and you, of course, are logged in as superuser), load the tape into the tape device.

2. Dump the /usr file system by typing:

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

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

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

3. Rewind the tape by typing:

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

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

## Remote File System Dumps Over the Local Area Network

You can dump file systems from an NFS server or standalone on a local area network, onto a tape mounted on another (remote) machine's tape drive. This is particularly useful if the local tape drive is down, though it, of course, requires that another tape drive be on your network. The name of the machine you are dumping from must be in the `/.rhosts` file on the machine with the remote tape drive. This is because you have to be superuser in order to run the remote dump commands.

The command you use is `rdump`. Its syntax is

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

The `rdump` command is a version of `dump` used specifically for accessing a remote machine over the network. The arguments to the command are explained below:

*options* These options are similar to those used for `dump`. They include 0-9, for dump level, `c` to indicate a cartridge tape is to be used, `d` *bpi* for reel-to-reel tape density, `f` for dump file, and several others, as discussed in the `rdump` man page in the *SunOS Reference Manual*

*remote_host:* This is the name of the remote machine with the tape drive you want to dump to. Be sure to include the colon (:) after the machine name, with no spaces before or after it.

*/dev/tape_dev* This is the device name of the tape drive on the remote host. Refer to the tables in the subsection, "Tapes and Equipment for Backing Up" for the proper device abbreviation for the remote host's tape drive.

*/dev/file_system_to_dump*
This is the local disk partition containing the file system you want to dump. Refer to the table at the beginning of this subsection for the device abbreviation for your disk controller.

### A Sample Remote Incremental Dump

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

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

1. Log in to the NFS server as superuser.
2. If you are not certain whether the server is in the `.rhosts` file of the machine with the tape drive, try to remote log in to that machine. If you cannot `rlogin`, your NFS server is not in the remote machine's `.rhosts` file.

Contact the person responsible for the remote machine and have him or her add you to its .rhosts file.

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

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

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

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

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

## 7.2. Restoring Files and File Systems

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

Topics covered in this section include:

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

### Using the /etc/dumpdates File

/etc/dumpdates keeps a record of each dump you take, provided that you have specified the u option of dump (as in the procedures in the section, "Backing Up Your File Systems"). Here is a typical /etc/dumpdates file from an NFS server:

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

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

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

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

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

### Using the restore Command

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

The syntax of restore is:

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

Here *key* refers to one or more of a number of keys available through restore. These keys are composed of one *function letter* and possibly one or more *function modifiers*. For a complete discussion of restore's keys and other arguments, see the `restore`(8) man page in the *SunOS Reference Manual.*

The most frequently used function letters are:

t — Verifies that the files you specified on the command line are on the tape. If `restore` finds the files, it displays their names on your screen. If you run `restore` with just the `t` option and no file name, `restore` lists all files on the tape.

i — Runs the interactive version of `restore`. The program runs an interactive environment that lets you select specific files to be restored.

r — Tells `restore` to do a recursive restore; that is, copy everything on the tape.

x — Restores only the files named on the command line.

help — Display a "help" screen.

The most frequently used function modifiers are:

v — Displays the names of each file as it is restored—the verbose option.

f — Indicates that you are going to specify on the command line the tape device you want `restore` to read from.

The *[names ...]* argument is the name of the file, files, or file systems that you want to restore.

## Restoring Individual Files

Frequently, a user will ask you to restore a file, files, or entire file system he or she has lost through carelessness, hardware failure, a faulty program, or by using a program improperly. Cases like this require you to be able to restore individual files or small groups of files.

Here are some actions you might want to take before actually using `restore` to make the restore process easier and more accurate.

1. Ask the user the date when the file was lost (in most cases, it's the present date), or the approximate date when the file last was in the state he or she wants to recover.
2. Refer to the `/etc/dumpdates` file to find the date of the last dump that would have the requested version of the file on it. Note that this is not necessarily the most recently backed up version of the file.
3. Retrieve the tape containing the dump made on that date.

   You should always be aware of the storage organization of backup tapes at your site, so that you can locate tapes that are months or years old. This is particularly important when you must restore a subdirectory containing many files worked on over a long period of time. In such cases, the needed versions of files are often not on a single dump tape.

4. Typically, you restore files from the dump tape into the `/var/tmp` directory, though this is certainly not required. If you want to create a special directory to receive restored files, log in to your system as superuser and create this directory. You should almost never restore a file to its original directory.

5. Change the ownership of that directory to the owner's name by using the `chown` command. (An example of `chown` usage is given in the next procedure. Refer to the `chown(1)` man page for more information about this command.)

Once you have taken these precautions, you are now ready to use `restore`.

## Procedures for Restoring a File

In most cases, the following instructions apply to both server and standalone configurations, and to both 1/2 inch and 1/4 inch tape drives. The instructions include special steps for configuration-dependent or tape-size-dependent cases.

**How to Verify a File, Then Restore It**

1. Log in as superuser and load the proper dump tape into the tape unit.

2. Go to the directory where you want to copy the restored file:

```
# cd /var/tmp
```

If you don't want to restore into `/var/tmp`, substitute the name of the directory to receive the restored file.

3. Type the following to verify that the file exists:

```
# /usr/etc/restore tf /dev/rmt8 file_name
```

The arguments to `/usr/etc/restore` are

`tf` The *t* function letter tells `restore` to verify whether the file, files, or directory given as the *file_name* argument are on the tape. *f* tells `restore` to find the file on the tape device named on the command line.

`/dev/rmt8` This is the name of the tape device used. In this case it is a reel-to-reel tape drive using 6250 bpi tape. If you are using a different tape device, substitute the proper abbreviations for your device in this command.

*file_name* This is the name of the file, files, or directory that you want to restore.

4. Now, restore the file by typing the following:

```
# /usr/etc/restore x /dev/rmt8 file_name
```

The x argument tells `restore` to copy the file, files, or directory *file_name*. If *file_name* is a directory, `restore` then recursively extracts the directory.

5. Rewind the tape and take it offline by typing:

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

where *remote_host* is the name of the remote machine.

6. Change ownership of the restored file by typing:

```
# chown owner file_name
```

where *owner* is the owner's login name and *file_name* is the name of the file you just restored.

It is suggested that you let the user change the restored file to whatever name he or she chooses and move the file to its desired location.

**How to Interactively Restore Files**

The next procedure uses the interactive version of `restore`.

1. Log in as superuser and load the proper dump tape in the tape unit.
2. Go to the directory where you want to copy the restored file:

```
# cd /var/tmp
```

If you don't want to restore into `/var/tmp`, substitute the name of the directory to receive the restored file.

3. Type the following:

```
# /usr/etc/restore ivf  /dev/rst8
```

Here the *i* function letter invokes the interactive version of `restore`. `/dev/rst8` is the name of the tape device you are going to use. In this example, it is a nine-track cartridge tape device. If your system has a different tape device, substitute the proper device abbreviation here. Remember, you do not have to specify the blocking size for a tape cartridge device, since this is already written on the tape label.

The system invokes `restore` in interactive mode. You will see the following on your screen:

```
Verify tape and initialize maps
Tape block size is 64
Dump   date: Tue Aug 18 09:06:43 1987
Dumped from: Fri Aug 14 08:25:10 1987
Extract directories from tape
Initialize symbol table.
restore >
```

You type instructions after the `restore >` prompt shown above. Interactive `restore` understands a limited set of SunOS commands.

4. Display a directory using the `ls` command to verify that it has the files you want to restore:

```
restore > ls directory_name
```

5. Go to the directory with the files you want to retrieve and type the following:

```
restore > cd directory_name
restore > add file_name
```

   The `add` command tells `restore` to add *file_name* to the list of files to be restored. You can also give `add` the name of directories to be restored.

6. Restore the list of files by using the `extract` command:

```
restore > extract
```

7. Specify "volume 1" when asked for a volume number. If you are doing a multi-volume dump, you should start by specifying the last tape first. This will speed up the restore process.

8. Type **q** to leave interactive `restore`.

9. Complete Steps 4 and 5 of the previous procedure to rewind the tape and change ownership of the file.

## Restoring Files over a Local Area Network

You can restore file systems from a remote tape drive to an NFS server or standalone on a local area network. In order to do this, the name of the NFS server or standalone from which you are issuing the `restore` commands must be in the `/.rhosts` file on the machine with the remote tape drive.

The command you use is `rrestore`. Its syntax is

```
# /usr/etc/rrestore keys remote_host:/dev/tape_dev file_name
```

`rrestore` is a version of `restore` used specifically for accessing a remote machine over the network. The arguments to the command are explained below:

*keys* — These keys are the same as those used for `restore`. Note that you must use the `f` key with `rrestore`. Refer back to the subsection, "Using the `restore` Command," for more information.

*remote_host:* — This is the name of the remote machine with the tape drive you want to use.. Be sure to include the colon (:) after the machine name, with no spaces before or after it.

*/dev/tape_dev* — This is device name of the tape drive on the remote host.

*file_name* — This is the name of the file or files you want to restore.

### A Sample Remote Restore

This example assumes that you are using an NFS server and accessing a remote machine with a cartridge tape drive attached to a SCSI disk controller.

1. Log in to the NFS server as superuser.

**Note:** If you log in as a member of the operator group, your machine does not have to be in the server's `.rhosts` file. Refer to Chapter 13, "The Sun Network File System," for more information about the `/etc/group` file and the operator group.

2. If you do not know whether the server is in the `.rhosts` file of the remote machine, try to remote log in to that machine. If you cannot `rlogin`, your NFS server is not in the remote machine's `.rhosts` file. Contact the person responsible for the remote machine and have him or her add you to its `.rhosts` file.
3. Load the tape in the remote tape drive. (Or, have its administrator do this for you if the remote machine is not easily accessible from the server.)
4. Go to the directory that you want to receive the copied file.

```
# cd /var/tmp
```

If you do not want to copy the files to `/var/tmp`, substitute the name of the directory you want to use in this command.

5. Type the following:

```
# /usr/etc/rrestore xf remote_host:/dev/rst8 file_name
```

6. Rewind the tape when you have restored the files by typing:

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

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

### Restoring an Entire File System

Occasionally a file system becomes so damaged that you have to restore it entirely. Typically, this is due to a disk head crash. Fully restoring a file system such as `/home` can take a lot of time. But rest assured that if you have faithfully backed up your file systems, you can restore to them to their state as of the previous working day—or at least up to the last incremental dump, if your site does not require daily incremental dumps.

When restoring the file system, you use three commands that have not been previously covered in this manual. They are

```
newfs
mount
umount
```

They are described in the next subsections.

## The newfs Command

Before you restore the file system, verify that it is unmounted by displaying the /etc/mtab file. /etc/mtab lists all currently mounted file systems. Here is a sample of this file:

```
/dev/xy0a / 4.2 rw,nosuid 1 1
/dev/xy0h /usr 4.2 rw 1 2
/dev/xy0d /export/swap 4.2 rw 1 4
/dev/xy0f /export/root 4.2 rw,nosuid 1 3
/dev/xy2c /home 4.2 rw 1 3
```

If the file system is NOT listed in /etc/mtab, then you can be sure that it is not mounted.

Next, you need to run the newfs command. newfs creates a new file system on the specified disk partition. Its basic syntax is

```
# /usr/etc/newfs options /dev/rdisk_type_partition
```

where **/dev/r***disk_type_partition* is the disk controller name and disk partition where you want to build the new file system.

newfs has a fair amount of options that are not covered in this manual. Moreover, it is a variation of the more involved mkfs command, which you don't need to use when restoring hard partitions. Refer to newfs(8) and mkfs(8) in the *SunOS Reference Manual* for more information.

## The mount, and umount Commands

As part of your administrative tasks, you will perform two types of mounts. One type of mount is NFS, which is described in Chapter 13. The second type of mount is 4.2, or UFS, which is the type of mount used for a restore procedure. For UFS mounts, you use the mount and umount commands to attach (and remove) a file system to the directory you specify.

In a UFS mount, you tell SunOS to attach a file system located on the specified local disk partition. UFS mounts are listed in the mtab file as 4.2 mounts, because because the UFS/4.2 mount is the traditional mount operation provided by the BSD 4.2 operating system. If you look in the /etc/mtab file for a server, you will note that all file systems that the server mounts from its local disk are listed as 4.2 mounts. Any file systems the server might receive from a standalone or other server on the network would be listed as NFS mounts.

The syntax you use to perform a UFS mount is:

```
# /usr/etc/mount /dev/disk_type_partition /mnt
```

where *disk_type_partition* is the disk controller device abbreviation and the partition letter for the file system you want to mount. **/mnt** is the directory where you want to attach the file system. If required, you can use a different directory than /mnt, but this directory must be an empty directory, or *mount point*. Chapter 13 describes mount points in more detail.

When you want to remove a file system, you use the umount command as shown below:

```
# /usr/etc/umount /dev/disk_type_partition
```

where *disk_type_partition* is the partition containing the file system you want to unmount.

Refer to Chapter 13 and mount(8) and umount(8) in the *SunOS Reference Manual* for more information.

### Determining Which Tapes to Use

Before you perform the restore procedure, you need to determine which dump tapes to use. You always need the most recent Level 0 dump tape, You also need the most recent incremental dump tapes made at each of the higher levels.

For example, if you just make Level 0 and Level 9 dumps, you only need the last Level 9 dump made, since it picked up all changes made since Level 0. If you use a scheme similar to the ones illustrated in "Planning a Backup Strategy," you will need the most recent Level 0, Level 5, and Level 9 dumps.

### Procedures for Restoring an Entire File System

The next procedure shows how to restore the /home file system on an NFS server. You can use this procedure for restoring other file systems, including the /export/root file system on a server, and file systems you may have created for your site.

The procedure shows this type of restore operation by using an example. It assumes that you are restoring the /home file system on Partition h of an NFS server with a Xylogics 450 hard disk controller and a reel-to-reel tape drive using 6250 bpi tape. If the device abbreviations in the examples don't apply to your system, substitute the proper abbreviations, as shown in the device tables in "Backing Up Your File Systems."

**Note:** Do not use these instructions for restoring an entire root (/) or /usr file system on either a standalone or server. Instead, use the instructions in the subsection, "Restoring a Damaged root or /usr File System."

This procedure assumes that a serious software or hardware problem resulted in the loss of the /home file system. It also assumes that you have corrected the problem, have determined the tapes you need to use for the restore, and are now ready to restore the file system from the tapes.

1. Log in to the system as superuser.

2. Recreate the damaged file system by using newfs:

```
# /usr/etc/newfs /dev/rxy0h
```

   where */dev/rxy0h* is the partition where you want newfs to build the new file system. Note that you address the raw device with *r*.

3. Check the file system by typing:

```
# /usr/etc/fsck /dev/rxy0h
```

   Remember to specify the raw device.

Be cautious about running fsck on mounted file systems. It is safer to run it in single-user mode or on unmounted file systems. Keep running fsck until it no longer reports problems. If fsck continues to report problems, something may be wrong with your hardware.

4. Mount the file system on an existing directory before reloading. Typically, you use the mnt directory created in / by suninstall.

**Note:** Here you specify the block device, not the raw device.

```
# /usr/etc/mount /dev/rxy0h /mnt
```

5. Load the Level 0 dump tape; go to the /mnt directory; and restore the file system.

```
# cd /mnt
# restore rvf /dev/rmt8
```

6. Unmount the Level 0 tape, remove it, then load the next lowest level dump. Always restore tapes starting with the lowest level and continuing until you reach the highest level.

7. Use the same version of restore to copy the incremental dump tape. If you have more tapes to load, repeat Steps 6 and 7 for each tape.

8. Remove the file restoresymtable from /mnt (or your current directory).

```
# rm restoresymtable
```

restore creates this directory during its operation.

9. Go to another directory. (You can't unmount a file system while you are still in it.) Then, unmount the file system and check it with fsck. Remember to specify the raw device.

```
# cd /
# /usr/etc/umount /dev/rxy0h
# /usr/etc/fsck /dev/rxy0h
```

10. Do a full dump of the newly restored file system:

```
# /usr/etc/dump 0uf /dev/nrmt8 /dev/rxy0h
```

### Restoring A Damaged / or /usr File System

Restoring a damaged root or /usr file system from a dump tape is unlike other types of restore operations, because the programs you need to run are in the damaged file system. Therefore, you first need to load and boot the mini-root file system from the boot tape, and reload the file systems from there.

## Procedures for Restoring root

The procedures below are appropriate for NFS servers and standalones. The example is for a standalone system using a SCSI disk controller and a SCSI tape cartridge unit with nine-track tape. If the device abbreviations in the example do not apply to your configuration, remember to substitute the appropriate abbreviations when you need to do these procedures.

1. Locate and fix any bad blocks or other hardware problems.

2. Load and boot the mini-root file system from the boot tape:

```
b st()
boot: st(0,0,4)
from: st(0,0,5)
to: sd(0,0,1)
boot: sd(0,0,1)vmunix -as
```

   If you have disk problems, boot `format` from tape and fix the disk. Be sure to check that the disk label is good by using `format`'s `verify` option. (Refer to Chapter 10, "Maintaining Disks with Format," in this manual for more details.)

3. Use the `/dev/MAKEDEV` command to install the entries for the tape unit and disk type onto the mini-root file system. (The `MAKEDEV` command is fully explained in Chapter 11, "Adding Hardware to Your System.")

```
# cd /dev
# /dev/MAKEDEV st0
# /dev/MAKEDEV sd0
```

4. Use the `newfs` command to recreate the root file system:

```
# /usr/etc/newfs /dev/rsd0a
```

   Use `r` for raw device. Be sure to copy down and save the backup superblock numbers that `newfs` displays.

5. Check the root file system:

```
# /usr/etc/fsck /dev/rsd0a
```

6. Mount the file system so it can be restored. The example assumes the `/mnt` directory already exists; use `mkdir` to create it if it does not exist.

```
# /usr/etc/mount /dev/sd0a /mnt
```

7. Change to the `/mnt` directory and restore the level 0 dump tape.

```
# cd /mnt
# /usr/etc/restore rvf /dev/rst8
```

8 Continue to restore incremental dumps in the order of lowest level number first and working up to the most recent, highest level tape last.

9. You must install a new boot block after restoring the root partition. You use the `installboot` command below to create the boot block.

```
# cd /usr/mdec
# installboot /boot bootsd /dev/sd0a
```

10. Remove the `restoresymtable` file that `restore` creates in the working directory during its operation.

```
# rm restoresymtable
```

11. Unmount the file system and check it again with `fsck`.

```
# cd /
# /usr/etc/umount /dev/sd0a
# /usr/etc/fsck /dev/rsd0a
```

Notice that you specify the “r“ to have `fsck` check the raw device.

12. Do a full dump (Level 0) of the newly restored root file system.

```
# /usr/etc/dump 0ucf /dev/rst8 /dev/rsd0a
```

## Restoring the `/usr` File System

The instructions for recovering `/usr` are similar to recovering `/`, but vary from site to site. If you have a server and left `/usr` in the same state as it was when installed from the tape, you only need to use a Level 0 dump of `/usr` to restore the file system. However, if you have moved any site-specific software into `/usr` on a server, particularly into the `/usr/local` directory, the `/usr` file system may be more volatile. Thus, you may have taken incremental dumps of `/usr` that you might want to use in addition to the last Level 0 dump. Furthermore, a standalone also has users' home directories in the `/usr` file system. Therefore you not only have to restore a Level 0 dump of `/usr`, you also have to restore the latest version of each higher level dump taken since the Level 0 dump.

The next instructions refer to the subsection “Procedures for Restoring `root`,” and only explain the steps that are unique to restoring `/usr`.

1. Perform Steps 1-7 as shown in “Procedures for Restoring `root`.” Remember to substitute the g partition for `/usr` (as opposed to the a partition for `/`) when specifying disk controller and tape device names.

2. For Step 8 on a server, you probably do not need to do a higher level dump than 0, since there are no user files in `/usr` initially, and you have probably mounted it read-only. However, if you are on a standalone, this is where you have to load incremental dump tapes.

3. Skip Step 9 in "Procedures for Restoring root." You only have to create a new boot block when you are restoring /.

4. Performs Steps 10, 11, and 12.

## 7.3. Monitoring File System Usage

Besides backing up and restoring files, the other maintenance operation you need to perform on a regular basis is checking file system usage. This section describes various utilities that help you check system usage. Topics discussed include:

- Commands that help you determine file system usage.
- Commands that show how a disk is being used.
- Steps for reducing overloaded systems.
- Suggestions for using the optional quota system.

Note that the information in this chapter applies to all configurations: diskless and dataless clients, standalones, and servers.

As you did for backup, you should devise a schedule for checking file system usage. How often you do this should depend on your configuration type, how many people use it, and how file-intensive the work they do is. For example, if your Sun workstation mainly runs large FORTRAN programs that perform mathematical calculations, this is not as file-intensive as using it to create large application programs or documents.

### Commands for Checking Disk Capacity—Checking Disk Resource Use

Several commands will tell you about the current use of disk resources. Three commands tell you the size of files in a file system: ls(1), du(1) and quot(8).

- ls gives you the size in kilobytes of the files in a directory when you give it the -s option, and recursively lists subdirectories when you give it the -R option. To find the largest files in the current directory, type:

  ```
  % ls -s | sort -nr | more
  ```

  To find out which files were most recently written, use ls -t. It lists files in order of most recently created, or altered, first.

- du gives the number of kilobytes contained in all files and, recursively, directories within each specified directory or file named.
- quot must be executed by the superuser, and gives the number of blocks currently owned by each user in the named file system.

These three commands allow you to quickly find large files. If disk storage space is short, you can move unused large files to a less expensive medium like tape.

## Commands for Checking Free Space

The df(1) command returns information about specific disk partitions, including the file space occupied by each file system, the space used and the space available, and how much of the file system's total capacity has been used. When df gives the percentage of the capacity used on the disk, the number indicates the optimum capacity. When a file system is 90 percent full, df claims that the file system is 100 percent full. If the file system becomes any fuller, file system access slows down. In fact, SunOS permits file systems to become only 90 percent full unless you are the superuser. Ordinary users get an error if they go beyond this limit.

Here is an example of the df command on an NFS server:

```
Filesystem              kbytes    used   avail capacity  Mounted on
/dev/xy0a                 7735    6200     761    89%    /
/dev/xy0h               232507   74366  134890    36%    /usr
/dev/xy2c               548367  486737    6793    90%    /home
/dev/xy3g                76111   28728   39771    42%    /export/exec
/dev/xy0d                57335   42274    9327    82%    /export/root
```

Note that this server has three disks mounted and has divided file systems among them.

## Making Room on File Systems

You should always monitor disk usage so that you are not caught in a situation where a file system has run out of space. But that is not always possible. This subsection shows how to free up space on an overloaded file system. The instructions apply to all configurations: diskless and dataless clients, servers, and standalones. Remember that even if you have a diskless client, you still need to check usage of the /home file system on your server, and, if necessary, delete unneeded files.

You know file system space has been exceeded when you get the following message on your workstation console:

```
file system full
```

Before trying anything else, suspend or kill the currently executing programs and try to make more room on the affected file system.

For systems that are filling up, or that become full suddenly, delete some files or move some files to another file system. You might want to try the following:

1. Check your directory for files named core. When certain programs crash, they leave a core file.

2. If you enable crash dumps, you should check the /var/crash directory. At the next reboot after a crash, savecore, a program which runs from /etc/rc.local, puts a copy of the core dump in /var/crash. This copy will stay in /var/crash until you remove it. If a second crash happens, savecore will put a second core dump in /var/crash, making the contents of this file system even greater. /var/crash is another good place to delete files.

As distributed, the section of /etc/rc.local that creates the crash dump file is commented out. Should you wish to get the crash dumps without using up too much disk space, you can create a file called minfree in /var/crash. savecore reads from this file, and if the file system contains less free space in kilobytes than the ASCII number contained in minfree, the core file will not be saved.

3. Check the directory /var/adm. It contains accounting files that increase in size as you use your system. You can disable login accounting by removing the file /var/adm/wtmp. You can disable process execution accounting by removing the file /var/adm/acct. Re-enable accounting by recreating the file in question with length zero. For example, type **touch** *filename*.

4. Look for obsolete files in /tmp and /var/tmp. Be careful not to delete currently active temporary files from /tmp, such as those file names starting with Ex or Rx, which the editors use. Also look for obsolete files in the subdirectories of /var/spool, such as /var/spool/mail and /var/spool/uucppublic.

## The Disk Quota System

Disk space is always a limited resource. The disk quota system provides a mechanism to control how much disk space each user uses. Typically, the quota system is used for an NFS server, but you can also use it for a standalone. Sites that have limited amounts of disk space or that require tight security might want to set up a quota system. Also, sites that have high disk usage, such as universities, might want to set up a disk quota system. Otherwise, you probably do not want to use the quota system.

**Note:** If you are the administrator of a diskless or dataless client upon which the NFS has imposed quotas, refer to the section, "Quotas and NFS Clients," at the end of this chapter.

You can set quotas for each individual user on any or all file systems. The quota system warns users when they exceed their allotted limit, but allows some extra space for current work. A user who remains over quota longer than a specified time-limit will get a fatal over-quota condition.

The quota system is an optional part of the kernel, that you can specify when the system is configured.

### Setting Up the Quota System

You must perform several steps to set up the disk quota system.

1. Log in to the NFS server machine as superuser, and verify that the system is configured to include the disk quota subsystem.

2. Go to the /usr/sys/sun[*2,3,4*]/conf file, and edit the kernel configuration file, as described in the instructions in Chapter 9, "Reconfiguring the System Kernel." If it isn't already there, add the line:

```
        options    QUOTA
```

Note that if you add or enable the quota option, you must also make sure that the line

```
        options    UFS
```

is enabled in the kernel configuration file.

3. Rebuild the kernel, as described in Chapter 9.

4. Decide which file systems need to have quotas imposed. Usually, only the /home file system on an NFS server and /usr on a standalone need quotas. (If you have created other file systems that have users' files on them, you may also want to impose quotas on them.) You may also want to include /usr on a server. If possible, /tmp should be free of quotas. Work out your plan before you begin the actual allocation. After you decide which file systems should have quota restrictions, allocate available space according to your plan.

   After you have decided which file systems need quotas, mark them by creating a file called quotas in the top directory of each.

   Make sure that the new quotas files are owned by root. If necessary, use the chown command to change their ownership.

The edquota(8) command actually sets quotas for specified users. edquota is a quota editor that allows you to set limits for one or more users at a time. The **-p** option allows you to duplicate the quotas of a prototypical user for a list of actual users. This is helpful when many or all users are given the same limits. The **-t** option edits the over-quota time-limit for the specified users. You can impose any combination of hard and soft limits for each user. A limit set to zero is disabled; if all the limits for a user are disabled, the entire quota for that user is also disabled and the system will not keep track of his or her usage. For details see the man page edquota(8).

## Turning on the Quota System

Once quotas are set and ready to operate, they must be turned on to start working. quotaon(8) tells the system to start enforcing the quotas set with edquota. quotaon enables disk quotas for one or more file systems; the specified file systems must be mounted at the time.

The quotaoff(8) command disables quotas — see man page for quotaon(8). However, when a file system is about to be unmounted by umount(8), quotas are disabled just before the specified file system is unmounted. This guarantees that the quota system will not be disabled if the unmount fails because the file system is busy.

Disk quota use and limits are stored in the file quotas on the root of the file system where the quotas are imposed, for example, directly under /home in this file system. The name quotas is not known to the system in any way. However, several user level utilities depend on it, so choosing any other name would cause problems.

The data in quotas is a list, indexed by user ID, with one entry for each user on the system, whether or not the user has a quota on this file system. If the user ID space is sparse, the file may contain holes that would be lost by copying it. Therefore, avoid copying the file.

### Administration of the Quota System

All quota administration commands must be run on mounted file systems. The commands will work whether or not the quota system is disabled.

The Sun disk quota system is based on the 4.2BSD quota system. The SunOS quota system limits the amount of time a user can be over quota. You can adjust that time-limit on a per-file-system basis using `edquota`(8).

You should periodically check the records retained in the quota file for consistency with the actual number of blocks and files allocated to a user. You certainly should do this after each reboot, and when quotas are first enabled for a file system. Do this using `quotacheck`(8).

While logged in as superuser, you can use `quota`(1) to examine the use and quotas for an individual, and `repquota`(8) to check the usages and limits for all users on a file system.

You can increase or decrease a user's quota limits at any time. Thus, a user who exceeds his limits can ask the system administrator to increase them.

When you remove a user from a system, you should use `edquota`(8) to set the removed user's limits to zero. Then, when the user's login is removed there will not be an entry for it in the `quotas` file.

### The User's View of Disk Quotas

The `quota`(1) command gives a user upon whom you've imposed quotas information about disk quotas. It also reports on current usage by a user. You can impose two kinds of limitations on a user. You can set a quota on the amount of disk space a user can fill up, or you can limit the number of files (inodes) the user can own. Typically, you impose both limitations.

The disk quota system has soft limits and hard limits. The soft limit is the number of 1K blocks (or files) that the user is expected to remain below. Each time the user exceeds this limit, a warning is displayed and a time limit set. If the user remains above quota and the time limit expires, the system now acts as if the hard limit has been reached, and no longer allocates resources to the user. The **only** way to reset this condition is for the user to reduce usage below quota. You may set the over quota time limits for each file system restricted by quotas.

The user cannot exceed the hard limit. If usage reaches this number, the system displays an error message in response to any further requests for space, or attempts to create a file. The first time this occurs, a message is displayed on the user's terminal. Only one message is written for each time the hard limit is reached. Space occupied must be reduced below the limit.

### What To Do When Quota Limit Is Reached

To recover from exceeding the hard quota limit, the user must do the following:

1. Abort the process or processes in progress on the file system that has reached its limit.

2. Remove enough files to bring the limit back below quota.

3. Retry the failed program.

The case is different when the user is in an editor and cannot write a file to a disk because it exceeds the quota. Most likely, the write attempt that generated the quota exceeded message also truncated the file in the editor. If the user aborts the editor in these circumstances, he or she will not only lose recent changes, but probably some, or all, of the file proper, as well.

There are several safe exits from this situation. The user may escape from the file to the shell, using the appropriate escape command, examine file space, then remove surplus files. However, it is probably quicker for the user to type `csh` to suspend the editor session while removing files.

For example, if the user types [CTRL-Z] (the CTRL key and the Z key simultaneously) while in an editor, he or she is returned to the `csh` for as many commands as needed to remove files and get below quota limit. To continue the editor session, the user just types '**`fg`**' (for foreground) to the shell prompt, and the suspended editor will restart. A third possibility is to write the file to some other file system (perhaps to a file on `/tmp`) where your quota has not been exceeded. After rectifying the quota problem, the file can be moved back to the file system where it belongs.

## Quotas and NFS Clients

Quotas on remotely mounted file systems work much the same as quotas on locally mounted file systems. The important difference is that no warnings are printed when the user goes over the soft limit. If your configuration is an NFS client, use `quota`(1) to find out the state of your quota allocation. Hard limit errors are treated like other hard errors in the NFS — for example exceeding actual disk capacity. When users exceed the hard limit, they get return code errors, not system error messages. The user may see `EDQUOT` error return codes from the system calls `write`(2), `fsync`(2) and `close`(2). Check these return codes to avoid disaster.

# 8

# Special Maintenance

# 8

# Special Maintenance

This chapter discusses many of the tasks that you, as system administrator, need to do on an as needed basis. Topics covered include

- What to do in case there is a power loss.
- How to troubleshoot system problems, based on error messages you receive from the monitor and boot program.
- What to do if your system crashes.
- How to set up the system log configuration and system performance.
- How to use system accounting.

## 8.1. Power Loss

A program called `/etc/update`, which is executed from the `/etc/rc` file, does a `sync` command every 30 seconds, thus insuring that the file system is up to date in the event the system crashes. If SunOS has not attempted to write to the file system for about 30 seconds before losing power, chances are that nothing abnormal will happen when you try to re-power the system — no guarantees on this, since losing power can always cause problems.

If you lose power before the system has had a chance to complete all outstanding disk writes, or if you halt the system without using `/etc/halt`, `/etc/shutdown`, or `sync`, SunOS may complain when it checks its file system with the `fsck` program.

You should always check file system consistency with `fsck` when rebooting. It will attempt to make corrective changes where it detects problems. You should let it do so, by answering "yes" to questions it asks you. During `fsck`, some unreferenced files may be placed in the `lost+found` directory. See the manual entry for `fsck` for more information about how this works and how to retrieve files placed there. If you do not allow `fsck` to make its changes, your file system will be in an unreliable state when you finally boot up. This can have disastrous consequences!

## 8.2. Messages from the Monitor and the Boot Program

### Message

```
Abort at aaaaaa
```

The monitor has aborted execution of the current program because you entered the "abort sequence" (upper left key held while pressing **A**) from the Sun keyboard, or pressed [BREAK] on a serial console. *aaaaaa* is the address of the next instruction. This address is always the same because the kernel calls the monitor. You can continue the program from there by entering the `c` command.

**Message**

```
Address Error, addr: xxxxxx at aaaaaa
```

The current program has stopped because it made an invalid memory access. *xxxxxx* is the (invalid) address; *aaaaaa* is an address near the instruction which failed (typically two to ten bytes beyond). There is no general way to recover from this error, except to debug the program.

**Message**

```
ar: cartridge is write protected
```

The current program is trying to write on an Archive tape cartridge, but the "Safe" switch at the top left corner of the cartridge is set to prevent writing on the tape. **Message**

```
ar: xxxx error
```

The monitor or boot program is trying to boot from an Archive tape, and encountered an unexpected error. The status bytes *xxxx* can be decoded by looking under "Read Status Command" in the *Archive Product Manual.* This error could be caused by incorrect cables, a bad tape, or other problems.

**Message**

```
ar: drive not responding
```

The monitor is trying to boot from an Archive tape, but can get no response from the tape drive. This can occur if your system contains an Archive controller board but no tape drive, or if the tape drive's cable is loose or disconnected, or if the tape drive's power is not on.

**Message**

```
ar: invalid state xx
```

This message indicates that the standalone I/O system has a bug in its Archive driver.

**Message**

```
ar: no cartridge in drive
```

The monitor or boot program is trying to boot from an Archive tape, but there is no cartridge in the tape drive.

**Message**

```
ar: no drive
```

The monitor or boot program is trying to boot from an Archive tape, but the specified drive does not exist. Typical Archive configurations include only Drive 0.

**Message**

```
ar: RDST gave Exception, retrying
```

The current program is trying to use the Archive tape drive, and encountered an error. The error is probably caused by hardware. Check the cable(s) that connect the tape drive to the system.

**Message**

```
ar: triggered at idle xx
```

This message indicates that the standalone I/O system has a bug in its Archive driver.

**Message**

```
Auto-boot in progress...
```

The monitor has finished its power-on sequence and is looking for a good device to boot SunOS from.

**Message**

```
Bad device
```

The current program (possibly the boot program) has tried to open a file without a device name (for example, `xy()`). This could mean that the boot command you typed had no device name.

**Message**

```
Bad format
```

The boot program is trying to boot from a file which is not in a standard `a.out`(5) format. The boot program can only boot files that are in this format, which is generated by the `ld`(1) command.

**Message**

```
bn void dd
```

A standalone program (such as the boot program) is trying to read a file from disk or net disk, and the block number it is trying to read is invalid.

**Message**

```
Boot:
```

The boot program is waiting for you to specify a device and file name to boot from. The boot program accepts the same commands that the monitor would, without the initial `b`. See the section *Booting From Specific Devices* above.

**Message**

```
Boot: dev(ctlr,unit,part)name options
```

The monitor or boot program is preparing to boot the specified file from the specified device. Either you typed a boot command, or this is an autoboot after power-on. *dev* is the device type; *ctlr, unit,* and *part* are the controller, unit-within-controller, and disk partition number. *name* is the name of the file to boot from, if any; *options* are arguments for the booted program, such as `-s`. If you enter a boot command to the monitor, this message will be printed twice; once by the monitor and once by the boot program.

**Message**

```
boot failed
```

The boot program has tried to boot the device and/or file you specified, but could not. A preceding message should give more details about why.

**Message**

```
Boot syntax: b [!][dev(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.

**Message**

```
Bus Error, addr: xxxxxx at aaaaaa
```

The current program has stopped because it tried to make an invalid memory access. The reason for the error is shown before this message. The memory location being accessed was *xxxxxx*, and the instruction that made the access is near location *aaaaaa*. There is no way to recover from this error, in general, except to debug the program.

**Message**

```
Can't write files yet...Sorry
```

The current program is trying to write to a disk or network disk file through the standalone I/O system. Writing on files (as opposed to writing on devices) is not supported when running standalone (that is, before booting the kernel).

**Message**

```
Corrupt label

Corrupt label on head h
```

The monitor or boot program is trying to boot from a disk. The first sector of the disk appears to be a label (as it ought to be), but the checksum on the label is wrong. Try again a few times; if the problem recurs, you should probably relabel your disk. Before trying to relabel your disk, make sure that you know what ought to be in the label — writing the wrong label on the disk is likely to destroy some or all files on the disk.

**Message**

```
count=ddd ?
```

A standalone program is trying to write to a device and has specified a block size that is not a multiple of 512. The write proceeds anyway, but may cause incorrect results.

**Message**

```
Damage found, damage...
```

As part of the power-on self test procedure, the monitor has found damage in one or more parts of the system. Report this message to your local service representative or Sun Microsystems Field Service. *Damage* is a list of subsystem names, such as "memory" or "timer."

**Message**

```
ec%d: Ethernet jammed.
```

After 16 failed transmissions and backoffs using the exponential backoff algorithm, the packet was dropped.

**Message**

```
ec%d: can't handle af%d.
```

The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

**Message**

```
Exception ee at aaaaaa
```

The current program has stopped because it received an interrupt. The interrupt could have been caused either by hardware or software. *ee* is the hexadecimal address of the interrupt vector used; you can look it up on a Motorola 68010 or 68000 reference card or CPU manual to see what kind of interrupt has occurred. *aaaaaa* is the address of the instruction where the interrupt occurred. If *ee* is `2c`, the partition you are trying to boot from is probably missing its boot track.

**Message**

```
Extra chars in command
```

Your previous `u` command had extra, unrecognized characters on the end.

**Message**

```
FCn space
```

The address space being accessed by the monitor's memory reference commands is defined by Function Code number *n*. See the Motorola 68010 or 68020 CPU manual for more information. This message is printed by the `s` command.

**Message**

```
For phys part p, No label found.
```

The boot program is trying to boot from a non-zero "physical partition" on a disk, and can't find a label. Physical partitions are used for disk drives, part of which are fixed and part of which are removable.

**Message**

```
-> Give the above information to your service-person.
```

The monitor has found a hardware problem while executing its power-on self test procedure. The preceding messages describe the error in more detail. Report the problem to your local service staff, or to Sun Microsystems Field Service.

**Message**

```
Giving up...
```

See `Waiting for disk to spin up....` The monitor has given up on waiting for the disk to become ready.

**Message**

```
ie: cannot initialize
```

The monitor or boot program is trying to boot from a Sun-2 Ethernet controller, and something serious has gone wrong with the board. Call your local Sun Microsystems Field Service person.

**Message**

```
ie%d: Ethernet jammed
```

Network activity has become so intense that sixteen successive transmission attempts failed, causing the 82586 to give up on the current packet. Another possible cause of this message is a noise source somewhere in the network, such as a loose transceiver connection.

**Message**

```
ie%d: no carrier
```

The 82586 has lost input to its carrier detect pin while trying to transmit a packet, causing the packet to be dropped. Possible causes include an open circuit somewhere in the network and noise on the carrier detect line from the transceiver.

**Message**

```
ie%d: lost interrupt: resetting
```

The driver and 82586 chip have lost synchronization with each other. The driver recovers by resetting itself and the chip.

**Message**

```
ie%d: iebark reset
```

The 82586 failed to complete a watchdog timeout command in the allotted time. The driver recovers by resetting itself and the chip.

**Message**

```
ie%d: WARNING: requeueing
```

The driver has run out of resources while getting a packet ready to transmit. The packet is put back on the output queue for retransmission after more resources become available.

**Message**

```
ie%d: panic: scb overwritten
```

The driver has discovered that memory that should remain unchanged after initialization has become corrupted. This error usually is a symptom of a bad 82586 chip.

**Message**

```
ID PROM INVALID
```

The monitor cannot find a valid ID PROM on the CPU board. The ID PROM contains the machine's serial number and other information specific to your system. If you have recently changed CPU boards, it is possible that you installed the ID PROM incorrectly. You should attempt to locate the correct ID PROM and install it in your CPU board.

**Message**

```
Invalid Page Bus Error ...
```

See `Bus Error....`
The attempted access was invalid because the virtual page containing the addressed data has been designated as invalid. It usually means that your program is using the wrong address.

**Message**

```
Invalid selection
```

Your last `u` command was not correct.

**Message**

```
Keyboard error detected
```

The microprocessor on the keyboard has reported an error. This probably means that your keyboard hardware is broken and should be replaced.

**Message**

**Note:** The LANCE Ethernet chip is a boot device option only on Sun-3/50s and 3/60s. Other models do not support this chip

```
le%d: transmitter frozen -- resetting
```

A bug in the LANCE chip has caused the chip's transmitter section to stop. The driver has detected this condition and reinitialized the chip.

**Message**

```
le%d: out of mbufs: output packet dropped
```

The driver has run out of memory to use to buffer packets on output. The packet being transmitted at the time of occurrence is lost. This error is usually symptomatic of trouble elsewhere in the kernel.

**Message**

```
le%d: stray transmitter interrupt
```

The LANCE chip has signaled that it completed transmitting a packet but the driver has sent no such packet.

**Message**

```
le%d: LANCE Rev C/D Extra Byte(s) bug; Packet dropped
```

The LANCE chip's internal silo pointers have become misaligned. This error arises from a chip bug.

**Message**

```
le%d: trailer error
```

An incoming packet claimed to have a trailing header but did not.

**Message**

```
le%d: runt packet
```

An incoming packet's size was below the Ethernet minimum transmission size.

**Message**

```
le%d: Receive buffer error - BUFF bit set in rmd
```

This error should never happen, as it occurs only in conjunction with a LANCE feature that the driver does not use.

**Message**

```
le%d: Received packet with STP bit in rmd cleared
```

This error should never happen, as it occurs only in conjunction with a LANCE feature that the driver does not use.

**Message**

```
le%d: Received packet with ENP bit in rmd cleared
```

This error should never happen, as it occurs only in conjunction with a LANCE feature that the driver does not use.

**Message**

```
le%d: Transmit buffer error - BUFF bit set in tmd
```

Excessive bus contention has prevented the LANCE chip from gathering packet contents quickly enough to sustain the packet's transmission over the Ethernet. The affected packet is lost.

**Message**

```
le%d: Transmit late collision -  Net problem?
```

A packet collision has occurred after the channel's slot time has elapsed. This error usually indicates faulty hardware elsewhere on the net.

**Message**

```
le%d: No carrier - transceiver cable problem?
```

The LANCE chip has lost input to its carrier detect pin while trying to transmit a packet.

**Message**

```
le%d: Transmit retried more than 16 times - net jammed
```

Network activity has become so intense that sixteen successive transmission attempts failed, causing the LANCE chip to give up on the current packet.

**Message**

```
   le%d: missed packet
```

The driver has dropped an incoming packet because it had no buffer space for it.

**Message**

```
le%d: Babble error - sent a packet longer than the maximum length
```

While transmitting a packet, the LANCE chip has noticed that the packet's length exceeds the maximum allowed for Ethernet. This error indicates a kernel bug.

**Message**

```
le%d: Memory Error!  Ethernet chip memory access timed out
```

The LANCE chip timed out while trying to acquire the bus for a DVMA transfer.

**Message**

```
   le%d: Reception stopped
```

Because of some other error, the receive section of the LANCE chip shut down and had to be restarted.

**Message**

```
   le%d: Transmission stopped
```

Because of some other error, the transmit section of the LANCE chip shut down and had to be restarted.

**Message**

```
   Load: dev(ctlr,unit,part)boot
```

The monitor has loaded in the *mini* boot program from a disk drive or network disk. The mini boot is now reading in the real boot program from the disk. The real boot program will then read in the program you requested.

**Message**

```
Lower Byte Parity Bus Error ...
```

See `Bus Error...` and `Parity....` The preceding access was to a word in memory with a parity error in its lower byte.

**Message**

```
Misplaced label on head n
```

The monitor or boot program is trying to boot from a disk. It has found a label that seems to identify itself as belonging to a different read/write head from the one where the label is written. See `Corrupt label...` above.

**Message**

```
mt: controller does not initialize
```

The monitor is trying to boot from nine-track tape, and could not get the tape controller to complete its initialization sequence. This might indicate a possible defect in the controller, or incorrect configuration of the controller board.

**Message**

```
mt: error 0xxx
```

The monitor is trying to boot from nine-track tape, and encountered an unexpected error. The error number *xx* can be decoded by looking in *Appendix C* of the *Tapemaster Product Specification*, which is supplied with your tape drive.

**Message**

```
mt: unit not ready
```

The monitor is trying to boot from nine-track tape, but the tape drive is not ready. Check to see that the drive is online.

**Message**

```
No controller at mbio xxxx
```

The monitor is trying to boot, but it can't find a device controller where you asked it to look. You should try another boot command, or make sure that your controller board is plugged in and has all its jumpers and switches set properly.

**Message**

```
No default boot devices
```

The monitor is trying to boot but it can't find a disk or Ethernet interface to boot from. To boot from a tape, you must specify the device name explicitly, as in '`b ar()`'.

**Message**

```
No label found -- attempting boot anyway.
```

The monitor or boot program is trying to boot from a disk, and can't find a valid label on the disk. This is best fixed by booting a copy of `format` (8S) from a different device (for example, network disk or tape) and using the `verify label` and `label` commands. See the warning under `Corrupt label` above. This error might also be caused by missing or bad disk cables.

**Message**

```
No more file slots
```

The current program is using the standalone I/O library and has opened too many devices or files.

**Message**

```
not a directory
```

The current program (possibly the boot program) has tried to open a disk or network disk file with a pathname, but one of the names in the path is not a directory.

**Message**

```
name not found
```

The boot program has searched for the requested file, but cannot find it. You can retry your boot command, using * instead of *name*, to get a list of the names that exist in that directory.

**Message**

```
null path
```

The current program (possibly the boot program) has tried to open a file whose name is empty.

**Message**

```
PageMap aaaaaa [ss]: xxxxxxxx?
```

The monitor is displaying or modifying a page map entry because you entered a `p` command. *aaaaaa* is the virtual memory address whose map entry is being examined. *ss* is the segment map entry that is being used to map this page map entry and page. *xxxxxxxx* is the page map entry itself. You can enter a space and type RETURN to get back to command mode.

**Message**

```
Parity Bus Error ...
```

See `Bus Error....` The attempted access was probably valid, but was canceled because the preceding access was to memory with bad parity. (Parity errors are reported on the memory cycle **after** the failing cycle.) If neither "Upper Byte" nor "Lower Byte" is reported, the parity on both bytes was invalid. The access address printed in the Bus Error message is probably not relevant to the parity error. There is no general way to recover from this error; a good starting point, though, is to boot `parscan` (8S), which will search all of memory for parity errors.

**Message**

```
Please clear keyboard to begin
```

The monitor is trying to listen for your typing on the keyboard, but cannot tell which shift keys are down until you release all the locking keys ( Caps Lock and Shift Lock). Once it has seen all the keys released, it can then track the movements of the keys and typing will work.

**Message**

```
Please start it, if necessary, -OR- press any key to quit.
```

See `Waiting for disk to spin up....`

**Message**

```
Possible boot devices:
```

You have asked for a list of boot devices with the `b ?` command.

**Message**

```
Protection Bus Error ...
```

See `Bus Error....` The attempted access was invalid because your program is not permitted to access the addressed data in this way; for example, writing to that page is disallowed.

**Message**

```
ROM Rev x, mm memory installed
```

The monitor is identifying its revision level and the system configuration as part of the power-on sequence. *x* is a letter or phrase indicating which particular version of the monitor is installed; *mm* shows how much memory was found during system configuration at power-on. If an even number of megabytes is installed, *mm* is displayed as *nn*MB; otherwise as *nnnn*KB.

**Message**

```
Retensing...
```

The monitor is attempting to boot from an Archive tape. Its first attempt failed, so it is retensing the tape (winding all the tape from one reel to the other), which makes it much more likely to succeed.

**Message**

```
sd%d%c:  cmd how (msg) starting blk %d,  blk %d (abs blk %d)
```

A command such as read or write encountered an error condition (how): either it *failed*, the unit was *restored*, or an operation was *retry*'ed. The *msg* is derived from the error number given by the controller, indicating a condition such as "drive not ready" or "sector not found." The *starting blk* is the first sector of the erroneous command, relative to the beginning of the partition involved. The *blk* is the sector in error, again relative to the beginning of the partition involved. The *abs blk* is the absolute block number of the sector in error.

**Message**

```
Seek not from beginning of file
```

The current program is using the standalone I/O library and has tried to do an unsupported seek operation.

**Message**

```
SegMap aaaaaa: xx?
```

The monitor is examining or changing the segment map in response to your recent `m` command. You can enter a space and type RETURN to get back to command mode.

**Message**

```
Self Test completed successfully
```

The monitor has completed its power-on self test without finding any hardware problems.

**Message**

```
Self Test found a problem in something
```

The monitor has completed its power-on self test and found a problem in some subsystem. *something* describes the general location of the error. Further messages give more details; see `Wrote ...` and `Damage found....`

**Message**

```
Serial #some_number, Ethernet address n:n:n:n:n: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`.

**Message**

```
   Short read
```

The boot program is trying to boot a program from disk or net disk. It has located the program, but encountered an error while reading it into memory.

**Message**

```
   Size: text +data +bss bytes
```

The boot program is loading in the program you requested. *Text*, *data*, and *bss* are the sizes of the three sections of the program; they are printed as each is read into memory. After finishing display of this message, the boot program begins execution of your program; further messages can come from it instead of from the boot program or monitor.

**Message**

```
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` command, and the monitor is identifying its configuration. *Keyb* is either VT100 or Two-tone, depending which keyboard your monitor ROMs support.

**Message**

```
Sun Workstation, Model Sun-2/120 or Sun-2/170, Sun-2 keyboard
```

The Sun-2 Model 120 or 170 workstation has just been powered on, or you entered a `kb` command, and the monitor is identifying its configuration.

**Message**

```
Sun Workstation, Model Sun-3/100 or Sun-3/200, Sun-3 keyboard
```

The Model 3/100 or 3/200 workstation has just been powered on, or you entered a `kb` command, and the monitor is identifying its configuration.

**Message**

```
   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 that does not exist, or that has gotten into a hung state. If this occurs in response to a boot command, the device you are trying to boot from is not installed in your system.

**Message**

```
tm: no response from 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.*

**Message**

```
Unknown device
```

The current program (possibly the boot program) has tried to use a device which is unknown to the standalone I/O system.

**Message**

```
Upper Byte Parity Bus Error ...
```

See `Bus Error...` and `Parity....` The preceding access was to a word in memory with a parity error in its upper byte.

**Message**

```
uii, uoo, uaabaud, ubbbaud, uuaaaaaa, uecho
```

The monitor is describing its console and serial port configuration in response to a `u` command. *i* is the input device (`k` for keyboard, or `a` or `b` for a serial port); *o* is the output device (`s` for screen, or `a` or `b`); *abaud* and *bbaud* are the baud rates on the serial ports; *aaaaaa* is the address of the Zilog 8530 chip that implements the serial ports, and *echo* is `e` if input echoing is enabled (full duplex) or `ne` if disabled (half duplex).

**Message**

```
Using RS232 A input.
```

The monitor did not find the Sun keyboard, so it is taking input from one of the serial ports on the back of the workstation, marked RS232 A. If this is unexpected, make sure that the keyboard is plugged into the correct socket on the workstation. The keyboard must be plugged in before system power is turned on. If you connect a Sun keyboard after this message appears, you can let the monitor know about the keyboard by entering the Abort sequence. (Hold down the upper left key on the Sun keyboard, and press **A**.) The monitor will switch to using the Sun keyboard, since that's where the Abort was typed. Then type **c** to continue whatever program was running when you aborted.

If you don't want to use a Sun keyboard, connect a normal ASCII terminal to the RS232 A connector on the back panel. Configure the terminal for 9600 baud, no parity, one stop bit. Things that you type on the terminal will be displayed on the Sun video screen, if you have one, or on the terminal's screen.

**Message**

```
Waiting for disk to spin up...
```

The monitor is trying to boot from a disk. The disk is not ready, so the monitor is waiting in the hope that the disk is just starting to spin and will become ready soon. If you get this message when the power has been on for a while, your disk cables are probably loose or misconnected.

**Message**

```
Watchdog reset!
```

The current program has stopped executing with a " bus fault." This is explained in detail in the Motorola 68010 and 68020 manuals. It also occurs on machines with the SPARC architecture. The two most common causes are that low memory (interrupt vectors) has been overwritten, or the system stack pointer is pointing to an invalid address, or the kernel stack overflowed. There is a serious problem, possibly in the kernel you are running, more likely it is in the hardware.

**Message**

```
What?
```

You typed a command that the monitor does not recognize. Try again.

**Message**

```
Wrote wdata at address addr, but read rdata
```

The monitor has completed its power-on self test and found a problem in some subsystem. The preceding ''Self Test found a problem...'' message describes which part of the system was in error. This message gives more details about the error. *wdata* is the data that was written into part of the system, or which was expected to be there if the system was functioning normally. *addr* is the address where the data was read and/or written. For memory errors, this is a physical memory address; for other errors, the interpretation of this field depends on what subsystem was being tested. *rdata* is the data that was read back from *addr* and was found to be invalid because it was not the same as *wdata*. This information should be written down and reported to your local Field Service organization, or to Sun Microsystems Field Service. See the section, "Non-Critical Errors From Self Test" above.

**Message**

```
xt: 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 Xylogics Product Specification.

**Message**

```
xycn: self test error
```

Self test error in the controller; see the Maintenance and Reference Manual.

**Message**

```
xycn: WARNING: n bit addresses
```

The controller is strapped incorrectly. Sun systems use 20-bit addresses for Multibus based systems and 24-bit addresses for VMEbus based systems. See the subsection on the Xylogics controller in the appropriate Sun *Hardware Installation Manual* for your machine(s) for instructions on how to set the jumpers on the 450.

**Message**

```
xyn: unable to read bad sector info
```

The bad sector forwarding information for the disk could not be read.

**Message**

```
xyn and xyn are of same type (n) with different geometries
```

The Xylogics 450/451 does not support mixing the drive types found on these units on a single controller.

**Message**

```
xyn: initialization failed
```

The drive could not be successfully initialized.

**Message**

```
xyn: unable to read label
```

The drive geometry/partition table information could not be read.

**Message**

```
xyn: Corrupt label
```

The geometry/partition label checksum was incorrect.

**Message**

```
xyn: offline
```

A drive ready status is no longer detected, so the unit has been logically removed from the system. If the drive ready status is restored, the unit will automatically come back online the next time it is accessed.

**Message**

```
xync: cmd how (msg) blk #n abs blk #n
```

A command such as read or write encountered an error condition (how): either it *failed*, the controller was *reset*, the unit was *restored*, or an operation was *retry*'ed. The *msg* is derived from the error number given by the controller, indicating a condition such as "drive not ready," "sector not found", or "disk write protected." The *blk #* is the sector in error relative to the beginning of the partition involved. The *abs blk #* is the absolute block number of the sector in error. Some fields of the error message may be missing since the information is not always available.

**Message**

```
xy: init error xx
```

The monitor is trying to boot from the Xylogics disk and has encountered an error. The command being executed at the time is defined by the hexadecimal value *xx* (if present); the block number is *bbbbb* (if present), and the particular error is encoded as *nn*. The error and command can be decoded by looking in the Xylogics manual.

**Message**

```
xy: no bad block info
```

The boot program is trying to read from the Xylogics disk, but can't find the information about bad blocks on the disk. It continues, but if the program attempts to read any bad blocks (which have been remapped to elsewhere on the disk), the attempt will fail.

**Message**

```
zero length directory
```

A standalone program (possibly the boot program) is trying to read a file from disk, but one of the directories in the pathname has no files in it. The file system should be checked and fixed by using `fsck` (8).

## 8.3. When the System Crashes

With every computer system there is a possibility that it may crash or hang so that it no longer responds to commands. This section discusses some of the ways you can deal with system hangs and crashes.

When a Sun machine crashes, it attempts to run `sync`, which forces changed blocks to be automatically written on the disks. Then the system prints out a short message telling why it crashed, attempts to preserve a core image of memory, and invokes an automatic reboot procedure. If the reboot finds no unexpected inconsistency in the file systems due to software or hardware failure, it resumes multiuser operations. Note that crashing programs create a *core dumps*, which show the contents of registers when the program crashed. When the system panics or crashes, it produces a file known as a *crash dump*, which contains a core image of memory at the time of the crash. If you enable the crash dump option in the `rc.local` file, the system will place crash dumps in the `/var/crash` directory.

Error messages generated by a crash are discussed below. Also discussed are how the system crash dump is saved, how to get rid of unwanted crash dumps that can accumulate on your system, how to force a crash dump, what steps can be taken to analyze a crash dump, what to do if the automatic reboot fails, and when to call for help.

Typically, a bad program should never crash the system. If it does, there is probably a bug in the kernel. Sometimes, however, a user program may crash and "hang" something in the system — a user process, a user's window, or even the whole system — even though it continues to run. Some aspects of a hung system and how to overcome them are discussed below. As the section "User Program Crashes" explains below, you are sometimes forced to reboot after a crashed program, even though UNIX has not crashed.

### Crash Error Messages

When the system crashes, it prints out a message such as:

```
panic:  error message
```

where *error message* is one of the panic error messages described in the `crash`(8s) man page. Less frequently you might see the message

```
Watchdog reset!
```

in place of `panic`. If the system is crashing repeatedly, you should keep exact notes of each message, including punctuation and upper or lowercase lettering. This will help you in repairing the problem.

The file `/var/adm/messages` automatically stores system messages after a crash. If you cannot get a copy of a crash message from the console, look in this file for the information. It may also be helpful in determining patterns of behavior over a period of time.

## System Crash Dumps

As currently distributed, the system will not save a crash dump. You can enable crash dumps, however, by editing the /etc/rc.local file on your machine. The lines that tell the system to do the crash dump are commented out with pound signs for distribution. To enable crash dumps, locate the following lines in /etc/rc.local and remove the pound signs in front of them.

```
#mkdir /var/crash/`hostname`
#savecore /var/crash/'hostname' >/dev/console 2 > E1
```

Then change /var/crash/ `hostname ` to the directory where you want your crash dumps placed.

When the system crashes it writes, or attempts to write, an image of memory into the primary paging partition on disk. After the system is rebooted, the program savecore (8) runs and preserves a copy of this core image in the directory /var/crash/*hostname* (or /var/crash). In most cases the system reboots automatically after a crash, cleaning up any problems and allowing you to continue working as before. In cases like this, you probably will not want to save the crash dump.

If you allow crash dumps to accumulate in /var/crash/*hostname* (or /var/crash), they will eventually fill up the file system. There is a convenient way to prevent this problem. The savecore program reads from a file called minfree, in /var/crash/*hostname* (or /var/crash) that contains a single number (in ASCII). If the file system contains fewer free kilobytes than the number in minfree, then the crash dump will not be saved. If you do use minfree to remove the crash dumps, remember to lower its value, or remove the file entirely during those times when you want to save the core image for debugging purposes. minfree prevents crash dumps from being saved and never removes them.

Sometimes your machine will hang without crashing. You can usually force a crash dump in this situation. First, abort to the PROM monitor by typing the appropriate abort sequence for your keyboard. (Refer to *Installing the SunOS* for this information.) Then type:

```
    > g 0
```

The dump will be saved in the /var/crash/*hostname* (or /var/crash) directory according to the savecore procedure explained above. However, you cannot force a crash dump in this manner unless you have edited your /etc/rc.local file as explained above.

Sometimes the system can get so badly hung that a crash dump cannot be forced. This is especially true on diskless machines because the network has to be operating smoothly to be able to perform the dump.

## Analyzing System Crash Dumps

If you are a novice user, it is not recommended that you try to debug the system from the crash dump. Experienced users can use the `dbx -k(1)` command or the `adb(1)` command (with the `-k` option) to debug the crash dump. See *Debugging UNIX Kernels with* `adb` in *Debugging Tools.*

## User Program Crashes

User program crashes, as distinguished from operating system crashes, are almost always due to programmer error, that is, a bug in the program. The most common messages from a program crash are: `Segmentation Fault, Bus Error, Arithmetic Exception`. These often indicate an illegal pointer in the program, perhaps the result of using a value where a pointer to a value was expected. System error messages are documented in *Debugging Tools.* A crashing program will usually crash dump in its current directory if it has write permission. If it does, you will see the message

```
Core Dumped
```

Sometimes the system appears hung or dead; that is, it does not respond to anything you type. Before assuming your program has crashed, check the items below to make sure there is not a simple reason why the system is not responding.

1. Type CTRL-Q in case you accidentally pressed (CTRL-S. CTRL-S freezes the screen.)

2. The `tty` mode may be fouled up. Try typing the line feed character, CTRL-J, instead of the RETURN key, to force a line feed. If the system responds, type CTRL-J **`/usr/ucb/reset`** CTRL-J to reset the `tty` modes.

3. If you are running *SunView*, make sure the mouse cursor resides in the window where you are trying to type commands.

4. Type CTRL-\ (CTRL backslash). This should force a "quit" in the running program, and probably the writing of a core file.

5. Type CTRL-C to interrupt the program that may be running.

6. If possible, try logging in to the same CPU from another terminal, or `rlogin` from another system on the network. Type **`ps -ax`** and look for the hung process. If you can identify it, try to kill it. You will have to be superuser or be logged in as the same user running the process. Type **`kill`** *<pid number>*. If that does not work try **`kill -9`** *<pid>*. (A quick way to see if a **`kill`** has worked is to repeat it. If the response is `no such process,` it was killed.)

7. If all of the above fail, abort and reboot.

8. If even that fails, call Customer Service for help.

## Crashes Associated with Shared Libraries

Shared libraries are a new feature of SunOS for Release 4.0. The *Programming Utilities and Libraries* manual describes them fully. This section discusses problems that programmers might have with the runtime linker `ld.so` and shared libraries. If `ld.so` or a widely used shared library, such as `libc.so`, is deleted, most system utilities will fail to execute correctly and the system will become unusable.

As system administrator, you probably will have to install shared libraries and clear any problems associated with them. You must be root to install and delete `ld.so`, as well as any libraries in `/usr/lib`.

### Error Messages Associated with Shared Libraries

The following error messages appear if you have problems related to the runtime linker. Recovery procedures follow in the next subsection.

```
    ** crt0: no /usr/lib/ld.so
    ** ld.so: lib%s.%d.%d  not found
```

These are fatal error messages. They indicate that `ld.so` or a widely-used shared library is missing.

```
    ** ld.so: open error errno for %s
    ** ld.so: can't read struct exec for %s
    ** ld.so: %s is not for this machine type
```

These are fatal error messages, indicating that the shared library executable has been corrupted, was installed with the wrong access rights, or was built to execute on another architecture. The problems indicated by these messages are very rare.

```
** ld.so: warning: %s has older version than expected %d
warning for older minor version used
```

This is a warning message. It occurs when an executable that was previously linked with a newer version of a shared library gets linked against an older version of the library.

### Recovering from Problems with Shared Libraries

The following is a partial list of utilities that have been rebuilt without shared libraries so that a system can be restored:

```
    ln mount mv rcp restore tar umount
```

To recover from the problems listed above, you need to install a copy of `ld.so` into `/usr/lib`. Note that you must be root to install and delete `ld.so`, as well as any libraries in `/usr/lib`. If you can obtain a copy of `ld.so` but do not have write privileges to `/usr/lib` as root, you can install `ld.so` in another location, such as `/usr/tmp`, then use `mv` to move the linker to `/usr/lib`. Then reboot in single user mode, and use one of the above utilities to restore `ld.so` directly into `/usr/lib`. Then continue booting in multi-user mode.

### Installing a New Shared Library

If you have to install a new shared library, it must be assigned a major and minor number. Refer to the *Programming Utilities and Libraries* manual for more information. After a new library is installed it is highly recommended that you run `ldconfig` so that the cache of shared library information is updated.

### When To Call for Help

Before calling for help, make sure you have accurately copied down crash messages from the console, or taken them from the `/var/adm` files mentioned above.

If the automatic reboot fails with a message such as:

```
  reboot failed: help
```

try to run `fsck` in single user mode. If reboot efforts still fail, call for Tech Support help.

If you are having frequent crashes, gather all the information you can about them, and have it ready when you call for help.

## 8.4. System Log Configuration

The `syslogd` (error logging daemon) is now used by many system facilities. Typically these messages are written to `/var/adm/messages`. You can send other messages to specific facilities elsewhere. Messages generated by various programs are redirected to a file. You can have them written to the console, a log file, or sent to selected users, or to everyone who is logged in. You use the `/etc/syslog.conf` file to configure system logging.

On a weekly basis, `/var/adm/messages` will be aged by a command in the / file system's `crontab` file. `/var/adm/messages` will be renamed `/var/adm/messages.0`; `/var/adm/messages.1` will be renamed `/var/adm/messages.2;` and `/var/adm/messages.2` will be renamed `/var/adm/messages.3` . The current `/var/adm/messages.3` file, if any, will be deleted. Very old messages are not kept around indefinitely. A new `/var/adm/messages` file will be created, and subsequent error messages will be logged there.

### Pre-4.0 Program Logging to 4.0 `syslog` Daemon

Pre-4.0 programs will log messages with no facility code but with priorities in the range 1 - 9. Since the 4.0 `syslog` accepts priorities in the range 0 - 7, priorities 8 (LOG_INFO) and 9 (LOG_DEBUG) will look like priorities 0 (LOG_EMERG) and 1 (LOG_ALERT) from facility 1 (LOG_USER). Unfortunately, this will have the effect of making these low priority messages seem to be much higher priority than they really are.

Also, almost all of the values for logging levels have changed. This will cause, for instance, messages logged at the old LOG_CRIT level to be logged at the new LOG_NOTICE level. In general, old log messages will appear to be less important than intended.

The 4.0 `syslog` daemon will force messages that claim to be from the LOG_KERNEL facility to look like they came from the LOG_USER facility, unless they come from the local kernel. The syslog.conf file on the ''loghost'' machine will be set up to log all LOG_USER messages in the log file used to log LOG_MAIL messages.

### 4.0 Program Logging to Pre-4.0 syslog Daemon

All 4.0 programs using syslog will send their log messages to the local syslog daemon. The default 4.3BSD syslog configuration file causes all syslog messages to be logged in local files, although it does provide a facility to forward the syslog message onto a syslog daemon on another machine. This forwarding will be enabled for sendmail log messages and perhaps for "authorization system" log messages, forward- ing them to "loghost." These forwarded messages will include a facility code in their log message. Except for LOG_USER|LOG_EMERG (== 8) and LOG_USER|LOG_ALERT (== 9) (which, by default, will not be forwarded to the syslog daemon at "loghost"), these will all cause the priority field in the message to be two digits. The old syslog daemon does not understand multi-digit priority fields, and so will log the message with a default priority of LOG_ERR (== 4). Using the default configuration file, this will cause the message to be logged with all the sendmail log messages in /var/spool/log/syslog. This is, of course, not nearly so useful as logging to a 4.0 syslog daemon that would put the message into an appropriate file, but is consistent with the pre-4.0 syslog.

Various system daemons and programs record information in the system log to help you to analyze problems. They send this information to the syslog daemon on the local machine. The daemon receives these messages and records the information, notifies users of problems, or forwards the information to another machine, typically "loghost." See syslog(8) for more details on this process.

The default configuration runs a syslog daemon on each machine, and also keeps all messages on the local machine. If you are running standalone, the default is for the syslog daemon to keep all messages on your machine. However, in a network environment it's much easier to track problems if all machines log their information in a single place. Therefore, during first time SunOS installation, the suninstall program reconfigures things so that only the designated server is the loghost: suninstall strips the loghost alias from the /etc/hosts entries for the clients, and adds the alias for the server machine. This means that, for example, if the machine named "unity" is your network server, the beginning of your machines' /etc/hosts files might look like:

```
192.9.1.1    unity loghost
192.9.1.2    dynamic
192.9.1.3    string
192.9.1.4    relative
192.9.1.5    dimension
```

Now all messages sent to loghost (from dynamic, string, etc.) are sent to unity. There might also be other aliases on the same line of the /etc/hosts entry, like lprhost or mailhost.

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

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

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

## 8.5. Monitoring System Performance

The vmstat program provided with the system is designed to be an aid to monitoring systemwide activity — see vmstat(8). Together with the ps(1) command (as in ps av), it can be used to investigate system-wide virtual memory activity. By running vmstat(8) when the system is active, you can measure system activity in several areas, such as 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, 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 amount of active jobs may be great, and several of these jobs may often be blocked (b). If virtual memory is active, then the paging daemon will be running (sr will be non-zero). It is normal for the paging daemon to free pages when the virtual memory becomes 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 long pauses when interactive jobs such as editors or window system programs swap out to memory. If you expect to be in a memory-poor environment for an extended period, you might consider limiting system load.

The nfsstat(1) command displays statistical information about the Network File System (NFS), Remote Procedure Call (RPC), interfaces to the kernel. It can also be used to reinitialize this information.

If nfsstat with the default -csnr option shows a packet drop rate of more than 3 percent, you should suspect problems in the network interface hardware. Check the transceiver, the cable, and the Ethernet board. See nfsstat(1) for details about the command's options.

## 8.6. Accounting

To run accounting, the system kernel must include the `options SYSACCT` and `pseudo-device sysacct` lines. Make sure your kernel has them if you are trying to make accounting work.

SunOS records two kinds of accounting information: connect-time accounting and process-resource accounting. Connect-time accounting information is stored in the `/var/adm/wtmp` file, which is summarized by the program `/usr/etc/ac` (see `ac`(8)). Process-time accounting information is stored in `/var/adm/acct`, and analyzed and summarized by the program `/usr/etc/sa` (see `sa`(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 `/var/etc/cron` to be executed every day at a specified time. This is done by adding lines to root's `crontab` file; see `cron`(8) for details.

Each line in a `crontab` file consists of six fields, separated by spaces or tabs, as follows:

1. Minutes field, which can have values in the range 0 through 59.
2. Hours field, which can have values in the range 0 through 23.
3. Day of the month, in the range 1 through 31.
4. Month of the year, in the range 1 through 12.
5. Day of the week, in the range 0 through 6. Sunday is day 0 in this scheme of things. For backward compatibility with older systems, Sunday may also be specified as day 7.
6. The remainder of the line is the command to be run. A percent character in this field (unless escaped by \) is translated to a new-line character. Only the first line (up to a % or end of line) of the command field is executed by the Shell. The other lines are made available to the command as standard input.

Any of fields 1 through 5 can be a list of values separated by commas. A value can either be a number, or a pair of numbers separated by a hyphen, indicating that the job is to be done for all the times in the specified range. If a field is an asterisk character (*) it means that the job is done for all possible values of the field.

Note that the specification of days may be made by two fields (day of the month and day of the week). If both are specified as a list of elements, both are adhered to. For example, `0 0 1,15 * 1` would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to * (for example, `0 0 * * 1` would run a command only on Mondays).

## 8.7. Making Local Modifications

Locally written commands are typically kept in /usr/src/local and their binaries in /usr/local. This allows /usr/bin, /usr/ucb, and /bin to correspond to the distribution tape (and to the system manuals). People wishing to use /usr/local commands should be made aware that they aren't in the base manual.

A /usr/junk directory to throw garbage into, as well as binary directories /usr/old and /usr/new are useful. The man command supports manual directories such as #/usr/man/manj for junk /usr/man/manl for local manual page entries and /usr/man/mann for new manual page entries to make this or something similar practical.

# 9

# Reconfiguring the System Kernel

# 9

# Reconfiguring the System Kernel

You should reconfigure your system's kernel after installing Release 4.0. The kernel provided for SunOS Release 4.0 supports many more devices than previous releases. Therefore, it is significantly larger than the kernels of earlier releases, and will occupy a considerable amount of memory. Tailoring the kernel for your own system significantly improves system performance.

Reconfiguring the kernel is not a difficult task. The `GENERIC` kernel configuration files for each architecture contain instructions that help you decide which kernel entries your particular system needs. If you carefully follow the instructions provided, particularly in the section, "Procedures for Reconfiguring the Kernel," and in the `README` file, also in the `/usr/share/sys/sun`*[2,3,4]*`/conf` directory, you should have a streamlined, workable kernel without experiencing any problems. Advanced system administrators and others who want to configure their own device drivers, maintain multiple kernel images, and further customize their systems' kernels should also read Chapter 16, "Advanced Kernel Reconfiguration."

Chapter 9 discusses the following subjects:

- Reasons for reconfiguring the system's kernel
- Major sections of the `GENERIC` kernel configuration file, from a broad perspective.
- Contents of `GENERIC` for each model of Sun computer.
- Procedures for putting the reconfigured kernel into operation.
- Procedures for changing swap space.

## 9.1. Why Reconfigure the Kernel?

There are two major reasons why you should reconfigure the kernel:

- To free up memory that would otherwise be consumed by unused kernel modules, thus improving your system's performance.
- To tell the kernel about hardware that you have added after installation or about software packages that require kernel modification to support.

The SunOS Release 4.0 tapes contain the kernel configuration file for your Sun workstation. When you build the kernel from the kernel file supplied on the release tape, the utilities involved create code that supports all hardware and software devices available for your Sun workstation architecture. The resultant kernel is unnecessarily large; your particular system more than likely does not have nor need all these items. Furthermore, on machines with a small amount of memory, using the kernel configuration file on the release tape will waste a significant amount of main memory, seriously degrading system performance.

By modifying a copy of the `GENERIC` configuration file, you can restrict the modified kernel to only those items that apply to your configuration. The smaller, customized kernel takes up less space in memory, giving larger effective memory size to programs. This improves system performance substantially, particularly if you intend to run SunView and other large applications or programs.

You also must reconfigure the kernel when you make major hardware or software additions to your system. For example, you edit the kernel configuration file when you add new hardware, such as a second disk or graphics controller, as explained in Chapter 11, "Adding Hardware to Your System." In addition, you need to edit the kernel configuration file when you add major software packages that may not have been selected when you initially ran `suninstall`. For example, if you decide to attach your standalone configuration to the network file system (NFS), you have to enable this option from the configuration file.

## 9.2. Parts of the Kernel Configuration File

This section examines the kernel configuration file from both a broad and narrow perspective. First, it explains how the configuration file is used during the kernel building process. Then an overview is given, which describes the major sections of the configuration file. Finally, the annotated configuration section explains each line in the `GENERIC` configuration file for a Sun-2, Sun-3, and Sun-4.

### The Reconfiguration Process

Building a new system is a semi-automatic process. Most of it is handled by a configuration-build utility called `/usr/etc/config`, which generates the files needed to compile and link your kernel. Your major activity in the configuration process is to create *SYSTEM_NAME*, the kernel configuration file. *SYSTEM_NAME* will actually be the name of your machine or other identifying name. This file contains a description of the kernel you want `/usr/etc/config` to produce. `/usr/etc/config` then uses this information to create the directory `/usr/share/sys/sun`*[2,3,4]/SYSTEM_NAME* and builds the kernel there.

Rather than creating the configuration file from scratch, you can copy and edit the `GENERIC` configuration file provided with your release in `/usr/share/sys/sun[`*2,3,4*`]/conf`. `GENERIC` reflects the configuration file supplied by Sun, in that it contains all possible entries for your model of Sun workstation.

### Major Sections of the `GENERIC` Configuration File

The `GENERIC` configuration file is divided into three sections:

- A system identification section that identifies the type of machine you have, including the name that you want to give the kernel. This section is similar for the `GENERIC` configuration file for each model type.
- A connections section that tells the kernel which CPU board and bus connections are available for attaching controllers.

- A devices section that contains lines specific to each type of controller, disk, tape, or other type device board that you want to configure.

To understand the format of the configuration file, it is best to begin by examining GENERIC. Note that the pound sign character (#) starts a comments that continues to the end of the line. Here is an example of the first few lines of the Sun-3 GENERIC file.

```
# @(#)GENERIC 1.82 88/02/08 SMI
#
# This config file describes a generic Sun-3 kernel, including all
# possible standard devices and software options.
#
# The following lines include support for all Sun-3 cpu types.
# There is little to be gained by removing support for particular
# cpu's, so you may as well leave them all in.
#
machine         "sun3"
cpu             "SUN3_160"      # Sun-3/75, Sun-3/140, Sun-3/160, or Sun-3/180
cpu             "SUN3_50"       # Sun-3/50
cpu             "SUN3_260"      # Sun-3/260 or Sun-3/280
cpu             "SUN3_110"      # Sun-3/110
cpu             "SUN3_60"       # Sun-3/60
cpu             "SUN3_E"        # Sun-3E (Eurocard VMEbus cpu)
#
# Name this kernel "GENERIC".
#
ident           GENERIC
#
# This kernel supports about eight users.  Count one
# user for each timesharing user, one for each window
# that you typically use, and one for each diskless
# client you serve.  This is only an approximation
# used to control the size of various kernel data
# structures, not a hard limit.
#
maxusers        8

#
# Include all possible software options.
#
# The INET option is not really optional, every kernel must include it.
#
options         INET            # basic networking support - mandatory
#
# The following options are all filesystem related.  You only need
# QUOTA if you have UFS.  You only need UFS if you have a disk.
# Diskless machines can remove QUOTA, UFS, and NFSSERVER.  LOFS is
# only needed if you're using the Sun Network Software Environment.
#
options         QUOTA           # disk quotas for local disks
options         UFS             # filesystem code for local disks
options         NFSCLIENT       # NFS client side code

                .
                .
                .
#
# Build one kernel based on this basic configuration.
# It will use the generic swap code so that you can have
# your root filesystem and swap space on any supported device.
# Put the kernel configured this way in a file named "vmunix".
#
config          vmunix          swap generic

#
# Include support for all possible pseudo-devices.
#
# The first few are mostly concerned with networking.
# You should probably always leave these in.
              .
              .
              .
```

You can see by these line groupings that the configuration file has three different types of entries:

- Lines that give a general description of the system (parameters global to the kernel image that this configuration generates)
- A line that describes items specific to each kernel image generated
- Lines that describe the devices on the system and connections to which these devices are attached.

### The System Identification Section

The system identification section, which is shown in the illustration above, contains general system description lines and system-specific lines. They are described below.

### General System Description Lines

The first six general description lines in the configuration file are mandatory for every Sun workstation. They are:

#### machine *type*

This field tells the kernel that the system is to run on the machine type specified. The legal *types* for a Sun workstation are "sun2", sun3", and "sun4". Note that the double quotes are considered part of the description.

#### cpu *type*

This field indicates that the system is to run on the CPU type specified. More than one CPU type can appear in the configuration file.

Legal types for a Sun-2 machine are:

```
"SUN2_120"  # Sun-1/100U, Sun-1/150U, Sun-2/120, Sun-2/170
"SUN2_50"   # Sun-2/50, Sun-2/160
```

Legal types for a Sun-3 machine are:

```
"SUN3_160"  # Sun-3/75, Sun-3/140, Sun-3/160, or Sun-3/180
"SUN3_50"   # Sun-3/50
"SUN3_260"  # Sun-3/260 or Sun-3/280
"SUN3_110"  # Sun-3/110
"SUN3_60"   # Sun-3/60
"SUN3_E"    # Sun-3E (Eurocard VMEbus cpu)
```

Legal types for a Sun-4 machine are

```
"SUN4_260"  # Sun-4/260, Sun-4/280
"SUN4_110"  # Sun-4/110
```

ident *name*

This field tells the kernel the name you wish for the system identifier—the name for the machine or machines that run this kernel. You must include *name* in double quotes if it contains any numbers (for example, "SDST120"), or you will get a syntax error when you run `/usr/etc/config`.

Note that when editing the `GENERIC` file, you do not have to name your kernel `GENERIC`. If you specify `options generic`, then you must use the `swap generic` clause on the `config` line. When you specify these two, the kernel prompts you during the boot process for the location of `/root` and `/swap`, for example, on the local disk or from an NFS server.

maxusers *number*

This field tells the kernel that the maximum expected number of simultaneously active users on this system is *number*. The number you specify is used to size several system data structures. The recommended value for `maxusers` on a server, regardless of model type is 8. The `GENERIC` configuration files for each model type give further instructions to help you determine a specification for `maxusers` that is appropriate to your system's needs.

options *type*

The fields beginning with the word `options` tell the kernel to compile the selected software options into the system. *type* has a different value for each options line. For example, here are several options lines for a Sun-4 GENERIC kernel:

```
#
options        QUOTA          # disk quotas for local disks
options        UFS            # filesystem code for local disks
options        NFSCLIENT      # NFS client side code
options        NFSSERVER      # NFS server side code
options        LOFS           # loopback filesystem - needed by NSE
#
```

Refer to the `GENERIC` configuration file for your system to determine which options are appropriate for it.

## System-Specific Description Lines

The next type of line in the kernel configuration file is a single line specifying the name of the file the kernel build procedure will create.

The line has the following syntax:

```
config kernelname config_clauses
```

Here *kernelname* indicates the name of the loaded kernel image. Its value is usually vmunix.

*config_clauses* are one or more specifications indicating where the root file system is located and where the primary paging (or swap) device is located. A *config_clause* may be one or more of the following:

```
root [on] root device
```

This clause specifies the location of the root file system.

```
swap [on] swap_device
```

This clause specifies the location of the primary swapping and paging area. If you specify `swap generic` this will enable you to place your root file system and swap space on any supported device. If you have specified `options generic`, you must specify `swap generic`, as well.

```
dumps [on] dump_device
```

This clause specifies where the `/export/dump` kernel should place core images after a crash.

The "on" in the syntax of each clause is optional. Separate multiple *config_clauses* by white space. For example, the *config* line for a system with root on its first SMD disk (Partition a) and swap on Partition b of the same disk might be:

```
config vmunix root on xy0a swap on xy0b
```

Note also that the device names supplied in the clauses may be fully specified—as a device, unit, and file system partition—or underspecified. If underspecified, the `config` program uses built-in rules to select default unit numbers and file system partitions. (Chapter 16 explains rules that are followed for underspecified location of devices.) For example, the swap partition need not be specified at all if the root device is specified. This is because the default is to place the `/swap` partition in Partition b of the same disk where the root file system is located. Thus you could use the following *config_clause* to represent the same information as the previous clause:

```
config vmunix root xy0
```

**For diskless clients:**

Use the following `config_clause`

```
config vmunix root on type nfs
```

### The Pseudo-Devices Section

This section lists all possible *pseudo devices* for your model. A pseudo-device is a collection of programs or a device driver that has no associated hardware. For example, here are three pseudo-devices needed to run SunView 1

```
pseudo-device   win128  # window devices, allow 128 windows
pseudo-device   dtop4   # desktops (screens), allow 4
pseudo-device   ms3     # mouse support, allow 3 mice
```

The `GENERIC` configuration file gives suggestions as to which pseudo-devices you may need.

### The Connections Section

The next section in the configuration file lists the possible on board and bus connections, grouped together by the machine model. These connections, in conjunction with controllers, devices, and disks form a structure that enables your system to recognize various hardware attached to it. For each device or controller on a bus, you need to select the bus type it is connected to, as listed under connections for your machine type.

For a Sun-2, connections are divided into two groups, machine 1, which includes all workstations with a Multibus, and machine 2, which includes all workstations with a VMEbus. Sun-3s and Sun-4s have separate connection lists for each model, or group of models, as you will see if you examine their GENERIC configuration files. Here is a segment from the Sun-3 GENERIC kernel connections section.

```
# connections for machine type 1 (SUN3_160)
controller      virtual 1 at nexus ?    # virtually addressed devices
controller      obmem 1 at nexus ?      # memory-like devices on the cpu board
controller      obio 1 at nexus ?       # I/O devices on the cpu board
controller      vme16d16 1 at nexus ?   # VME 16 bit address 16 bit data devices
controller      vme24d16 1 at nexus ?   # VME 24 bit address 16 bit data devices
controller      vme32d16 1 at nexus ?   # VME 32 bit address 16 bit data devices
controller      vme16d32 1 at nexus ?   # VME 16 bit address 32 bit data devices
controller      vme24d32 1 at nexus ?   # VME 24 bit address 32 bit data devices
controller      vme32d32 1 at nexus ?   # VME 32 bit address 32 bit data devices
```

```
# connections for machine type 2 (SUN3_50)
controller      virtual 2 at nexus ?
controller      obmem 2 at nexus ?
controller      obio 2 at nexus ?
```

Note that machine type 1 is a Sun-3/160 and machine type 2 is a Sun-3/50. The first three connections, `virtual`, `obmem`, and `obio` are used by Sun for specific devices. For example, the `fpa` floating point accelerator uses the `virtual` connection, and various graphics controllers use `obmem` and `obio`. For machine type 1, connections prefaced with "`vme`" are on the VMEbus. For example, the phrase vme32d16 indicates a 32 bit VMEbus with 16 bit data. Note however that the Sun-3/50 does not have a vme connection listed.

The easiest way to modify the connections section is to leave as is all connections lines listed for your machine type. Then, comment out each connection line for all other machine types. That way, as you add controllers and devices, the connections are already enabled and will be recognized by your system.

### The Devices Section

The final section of the configuration file lists all devices that can be supported by a Sun-2, Sun-3, or Sun-4. Devices are grouped into controllers and, if applicable, the disks and tapes that may be connected to them. Each device is listed on a separate device description line. The basic format of these lines is described as follows

### Device Description Lines

When reconfiguring the kernel configuration file, you need to specify each device on your machine so that the generated kernel will recognize these devices during the boot process. Devices may be hardware-related entities, that is, controller boards and devices attached to the controllers, or software pseudo-devices .

The device description lines tell the system what devices to look for and use, and how these devices are inter-connected. Each line has the following syntax:

*dev_type dev_name* `at` *connect_dev more info*

Below is a definition of each parameter on the device description line.

### *dev_type*

This item specifies the device type. *dev_type* may be one of the following:

- **controller.** Usually a disk or tape controller.
- **disk or tape.** Devices connected to a controller.
- **device.** Hardware entity attached to the main system bus, for example, an Ethernet controller board.
- **pseudo-device.** Software subsystems or drivers with no associated hardware, such as the pseudo-tty driver and various network subsystems, such as the NFS and Internet subsystems.

### *dev_name*

Standard device name and unit number of the device you are specifying (if the device is not a pseudo-device). For example, *dev_name* for the first Xylogics disk controller on a system is `xyc0`.

*connect_dev*

Connection to which this device is attached. Here are the possible connections for all Sun workstation configurations:

| | |
|---|---|
| `virtual` | Virtual preset |
| `obmem` | On-board memory |
| `obio` | On-board I/O |
| `mbio` | Multibus I/0 (Sun-2 only) |
| `mbmem` | Multibus Memory (Sun-2 only) |
| `vme16d16 (vme16)` | VMEbus: 16 bit address/16 bit data |
| `vme24d16 (vme24)` | VMEbus: 24 bit address/16 bit data |
| `vme32d16` | VMEbus: 32 bit address/16 bit data (Sun-3 and Sun-4) |
| `vme16d32` | VMEbus: 16 bit address/32 bit data (Sun-3 and Sun-4) |
| `vme24d32` | VMEbus: 24 bit address/32 bit data (Sun-3 and Sun-4) |
| `vme32d32` | VMEbus: 32 bit address/32 bit data (Sun-3 and Sun-4) |

When modifying the configuration file, you must specify the connections that apply to your system, but do not need to specify connections that don't apply. If you are unsure of the type of system bus or data bus you have, refer to the hardware manuals that came with the system.

*more_info*

This is a sequence of the following:

```
csr addr drive number flags number priority level vector intr number
```

The arguments above are completely described in Chapter 16, because you need to supply values for them only if you are going to configure your own device drivers.

Briefly `csr` *addr* specifies the address of the csr (command and status registers) for a device. `drive` *number* specifies which drive the line applies to. `flags` *number*, `priority` *level*, and `vector` *intr number* are all values defined in the device driver programs.

Here is a sample set of lines describing Xylogics 450/451 disk controllers and disks.

```
controller   xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller   xyc1 at vme16d16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk         xy0 at xyc0 drive 0
disk         xy1 at xyc0 drive 1
disk         xy2 at xyc1 drive 0
disk         xy3 at xyc1 drive 1
```

Bus types and devices must hang off the appropriate controller, which in turn hangs off another controller until a configuration is formed that gets you to a bus type that hangs off a "nexus." Notice how each line in the connections section concludes with the words "`at nexus.`" On Sun systems, all bus types are considered to hang off a nexus. For example, the following SMD disk :

```
disk     xy0 at xyc0 drive 0
```

is attached to the Xylogics controller:

```
controller   xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
```

which is attached to the bus type vme16d16, as listed in the following entry from the connections section:

```
controller      vme16d16 1 at nexus ?
```

In order to determine and note which standard devices are present on your machine, boot the GENERIC kernel after you have executed *suninstall*. If you want, you can delete those lines that pertain to devices not on your machine. Or you can configure your file with the devices that are on other machines that you will possibly want to boot from the same kernel.

## 9.3. Modifying the Kernel Configuration Files

This subsection shows the annotated `GENERIC` kernel configuration files for each model of Sun computer. Note how the GENERIC file contains comments that suggest which entries to pick for your particular system.

**Note:** Some parameters relating to the System V Inter-Process Communication (IPC) extensions may also be tuned in the configuration file. These parameters do not appear in the GENERIC file but are documented in Chapter 16.

If the comments indicate that the line is **mandatory**, you *must* include it in every system configuration file, either exactly as it stands, or, if commentary indicates variables, with the variables adjusted to fit your system. Some options indicated as mandatory are only required if you have other related options selected for your system.

### Modification Procedures

Here are suggested procedures for modifying the GENERIC kernel configuration file.

1. Go through the next subsections and find the copy of `GENERIC` that pertains to your model of Sun computer.
2. Read the annotated `GENERIC` file and determine how you want to modify each line. You can modify a line by doing one of the following:

□ Changing its parameters so that it applies to your configuration. Usually you do this for lines indicated as **mandatory**. For example, the `maxusers` line is mandatory, but you can change its value from the default.

- "Commenting out" a line if it does not apply to your current configuration, but may apply to it in the future. To do this, type a pound sign (#) at the beginning of the line. The utilities that build the kernel ignore lines beginning with the pound sign.

  For example, suppose you have a SCSI-2 controller with one disk and one tape drive. You might want to comment out the lines that apply to a second SCSI-2 controller, second disk, and second tape drive. At a later date, you may add some or all of this equipment. All you need to do to have this equipment recognized is to remove the pound sign, then make the new kernel.

- Deleting any lines that will never apply to your configuration. For example, if you have a diskless client, you might want to delete lines applying to Xylogics disk and tape controllers. These controllers are often used with servers. If you do add a disk or tape to your machine, it will probably be enclosed in a "shoebox" containing SCSI controller, disk, and possibly, tape drive.

3. If you wish, mark up each line in the text, indicating the changes you want to make to the actual file.

4. Type the following:

```
# cd /usr/sys/sun[2,3,4]/conf
```

   to go to the directory containing the GENERIC kernel configuration file.

5. Copy the file `GENERIC`. Call the new file *SYS_NAME*, where *SYS_NAME* represents the name you want to give to your system. Use the following command:

```
# cp GENERIC SYS_NAME
```

   If your customized kernel is already in use and you now want to modify it, you should copy the customized kernel configuration file and edit the copy.

6. Change the permissions for *SYS_NAME* as follows:

```
# chmod +w SYS_NAME
```

7. Edit *SYS_NAME* using your preferred text editor. Use the notes you made as a guide while you make these changes. Make sure to include the proper device description lines for your machine.

Now you are ready to begin the reconfiguration process. Go on to the next major section, "Procedures for Reconfiguring the Kernel."

## The Sun-2 GENERIC Kernel

The following is the GENERIC configuration file for a Sun-2 system.

```
# For VME machines
#
# @(#)GENERIC 2.66 88/02/08 SMI
#
# This config file describes a generic Sun-2 kernel, including all
# possible standard devices and software options.
#
# The following lines include support for all Sun-2 cpu types.
# There is little to be gained by removing support for particular
# cpu's, so you may as well leave them all in.
#
machine     "sun2"
cpu     "SUN2_120"  # Sun-1/100U, Sun-1/150U, Sun-2/120, Sun-2/170
cpu     "SUN2_50"   # Sun-2/50, Sun-2/160
#
# Name this kernel "GENERIC".
#
ident       GENERIC
#
# This kernel supports about eight users.  Count one
# user for each timesharing user, one for each window
# that you typically use, and one for each diskless
# client you serve.  This is only an approximation
# used to control the size of various kernel data
# structures, not a hard limit.
#
maxusers    8

#
# Include all possible software options.
#
# The INET option is not really optional, every kernel must include it.
#
options     INET        # basic networking support - mandatory
#
# The following options are all filesystem related.  You only need
# QUOTA if you have UFS.  You only need UFS if you have a disk.
# Diskless machines can remove QUOTA, UFS, and NFSSERVER.  LOFS is
# only needed if you're using the Sun Network Software Environment.
#
options     QUOTA       # disk quotas for local disks
options     UFS     # filesystem code for local disks
options     NFSCLIENT   # NFS client side code
options     NFSSERVER   # NFS server side code
options     LOFS        # loopback filesystem - needed by NSE
#
# The following options are for accounting and auditing.  SYSAUDIT
# should be removed unless you are using the C2 security features.
#
options     SYSACCT     # process accounting, see acct(2) & sa(8)
options     SYSAUDIT    # C2 auditing for security
```

```
#
# The following options are for various System V IPC facilities.
# No standard software needs them, although some third party
# software relies on at least IPCSHMEM.
#
options     IPCMESSAGE  # System V IPC message facility
options     IPCSEMAPHORE    # System V IPC semaphore facility
options     IPCSHMEM    # System V IPC shared-memory facility
#
# The following option is only needed if you want to use the trpt
# command to debug TCP problems.
#
options     TCPDEBUG    # TCP debugging, see trpt(8)
#
# The following option includes the software DES support, needed if
# you're using secure NFS or secure RPC and you don't have a DES chip.
#
options     CRYPT       # software encryption (if no DES chip)

#
# Build one kernel based on this basic configuration.
# It will use the generic swap code so that you can have
# your root filesystem and swap space on any supported device.
# Put the kernel configured this way in a file named "vmunix".
#
config      vmunix      swap generic

#
# Include support for all possible pseudo-devices.
#
# The first few are mostly concerned with networking.
# You should probably always leave these in.
#
pseudo-device   pty     # pseudo-tty's, also needed for SunView
pseudo-device   ether       # basic Ethernet support
pseudo-device   loop        # loopback network - mandatory
#
# The next few are for SunWindows support, needed to run SunView 1.
#
pseudo-device   win128      # window devices, allow 128 windows
pseudo-device   dtop4       # desktops (screens), allow 4
pseudo-device   ms3     # mouse support, allow 3 mice
#
# The following is needed to support the Sun keyboard, with or
# without the window system.
#
pseudo-device   kb3     # keyboard support, allow 3 keyboards
#
# The following is for asynchronous tty support for the ALM-2 (aka MCP).
# If you have an ALM-2 (MCP) and it is being used to connect timesharing
# terminals, you will need this.
#
#pseudo-device  mcpa64
```

```
#
# The following is for the streams pipe device.  Currently nothing
# depends on this device so it is entirely optional.
#
pseudo-device   sp
#
# The following are for streams NIT support.  NIT is used by
# etherfind, traffic, rarpd, and ndbootd.  As a rule of thumb,
# NIT is almost always needed on a server and almost never
# needed on a diskless client.
#
pseudo-device   snit        # streams NIT
pseudo-device   pf      # packet filter
pseudo-device   nbuf        # NIT buffering module
#
# The following is for the "clone" device, used with streams devices.
# This is required if you include streams NIT support.
#
pseudo-device   clone

#
# The following sections describe what kinds of busses each
# cpu type supports.  You should never need to change this.
# (The word "nexus" is historical...)
#


# connections for machine type 1 (SUN2_120)
controller  virtual 1 at nexus ?    # virtually addressed devices
controller  obmem 1 at nexus ?  # memory-like devices on the cpu board
controller  obio 1 at nexus ?   # I/O devices on the cpu board
controller  mbmem 1 at nexus ?  # Multibus memory space
controller  mbio 1 at nexus ?   # Multibus I/O space

# connections for machine type 2 (SUN2_50)
controller  virtual 2 at nexus ?    # virtually addressed devices
controller  obmem 2 at nexus ?  # memory-like devices on the cpu board
controller  obio 2 at nexus ?   # I/O devices on the cpu board
controller  vme16 2 at nexus ?  # 16 bit address VMEbus (16 bit data)
controller  vme24 2 at nexus ?  # 24 bit address VMEbus (16 bit data)

#
# The following (large) section describes which standard devices this
# kernel supports.
#

#
# Support for 2 Xylogics 450/451 controllers with 2 drives each.
#
controller  xyc0 at mbio ? csr 0xee40 priority 2
controller  xyc0 at vme16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller  xyc1 at mbio ? csr 0xee48 priority 2
controller  xyc1 at vme16 ? csr 0xee48 priority 2 vector xyintr 0x49
```

```
disk        xy0 at xyc0 drive 0
disk        xy1 at xyc0 drive 1
disk        xy2 at xyc1 drive 0
disk        xy3 at xyc1 drive 1
#
# Support for the SCSI-2 host adapter with 2 disks and 1 1/4" tape
# on the first SCSI controller and 1 disk and 1 1/4" tape on the
# second SCSI controller.
#
controller  sc0 at mbmem ? csr 0x80000 priority 2
controller  sc0 at vme24 ? csr 0x200000 priority 2 vector scintr 0x40
disk        sd0 at sc0 drive 0 flags 0
disk        sd1 at sc0 drive 1 flags 0
disk        sd2 at sc0 drive 8 flags 0
tape        st0 at sc0 drive 32 flags 1
tape        st1 at sc0 drive 40 flags 1
#disk       sf0 at sc0 drive 8 flags 2
#
# Support for the second SCSI-2 host adapter.
# Only supports one SCSI controller.
#
controller  sc1 at mbmem ? csr 0x84000 priority 2
disk        sd2 at sc1 drive 0 flags 0
disk        sd3 at sc1 drive 1 flags 0
tape        st1 at sc1 drive 32 flags 1
#disk       sf1 at sc1 drive 8 flags 2
#
# Support for the Sky floating point processor.
#
device      sky0 at mbio ? csr 0x2000 priority 2
device      sky0 at vme16 ? csr 0x8000 priority 2 vector skyintr 0xb0
#
# Support for the 2 tty lines (ttya, ttyb) on the cpu board.
# Needed when using a terminal for the console device.
# Flags=3 says to supply carrier in software for both lines.
#
device      zs0 at obio 1 csr 0x2000 flags 3 priority 3
device      zs0 at obio 2 csr 0x7f2000 flags 3 priority 3
#
# Support for the keyboard and mouse interface.  Needed when
# using a frame buffer as the console device or with SunView.
# You can remove this line if you don't use the standard Sun
# Workstation keyboard and mouse, but if you leave it in don't
# change it.
#
device      zs1 at obmem 1 csr 0x780000 flags 0x103 priority 3
device      zs1 at obio 2 csr 0x7f1800 flags 0x103 priority 3
#
# Support for tty lines on first Multibus SCSI-2 board.
#
device      zs2 at mbmem ? csr 0x80800 flags 3 priority 3
device      zs3 at mbmem ? csr 0x81000 flags 3 priority 3
#
```

```
# Support for tty lines on second Multibus SCSI-2 board.
#
device      zs4 at mbmem ? csr 0x84800 flags 3 priority 3
device      zs5 at mbmem ? csr 0x85000 flags 3 priority 3
#
# Support for 4 ALM's (Systech MTI-800/1600).  Flags set for
# all lines to be local, i.e., carrier supplied by software
# rather than by the device.
#
device      mti0 at mbio ? csr 0x620 flags 0xffff priority 4
device      mti1 at mbio ? csr 0x640 flags 0xffff priority 4
device      mti2 at mbio ? csr 0x660 flags 0xffff priority 4
device      mti3 at mbio ? csr 0x680 flags 0xffff priority 4
device      mti0 at vme16 ? csr 0x620 flags 0xffff priority 4
    vector mtiintr 0x88
device      mti1 at vme16 ? csr 0x640 flags 0xffff priority 4
    vector mtiintr 0x89
device      mti2 at vme16 ? csr 0x660 flags 0xffff priority 4
    vector mtiintr 0x8a
device      mti3 at vme16 ? csr 0x680 flags 0xffff priority 4
    vector mtiintr 0x8b
#
# Support for the on-board Intel 82586 Ethernet chip on the Sun-2/50.
#
device      ie0 at obio 2 csr 0x7f0800 priority 3
#
# Support for the first Multibus Intel Ethernet board.
#
device      ie0 at mbmem ? csr 0x88000 priority 3
#
# Support for the second Multibus Intel Ethernet board.
#
device      ie1 at mbmem ? csr 0x8c000 flags 2 priority 3
device      ie1 at vme24 ? csr 0xe88000 priority 3 vector ieintr 0x75
#
# Support for the first 3COM Ethernet board.
#
device      ec0 at mbmem ? csr 0xe0000 priority 3
#
# Support for the second 3COM Ethernet board.
#
device      ec1 at mbmem ? csr 0xe2000 priority 3
#
# Support for 2 Ciprico TapeMaster tape controllers with 1 tape drive each.
#
controller  tm0 at mbio ? csr 0xa0 priority 3
controller  tm0 at vme16 ? csr 0xa0 priority 3 vector tmintr 0x60
controller  tm1 at mbio ? csr 0xa2 priority 3
controller  tm1 at vme16 ? csr 0xa2 priority 3 vector tmintr 0x61
tape        mt0 at tm0 drive 0 flags 1
tape        mt1 at tm1 drive 0 flags 1
#
# Support for 2 Xylogics 472 tape controllers with 1 tape drive each.
```

```
#
controller  xtc0 at mbio ? csr 0xee60 priority 3
controller  xtc0 at vme16 ? csr 0xee60 priority 3 vector xtintr 0x64
controller  xtc1 at mbio ? csr 0xee68 priority 3
controller  xtc1 at vme16 ? csr 0xee68 priority 3 vector xtintr 0x65
tape        xt0 at xtc0 drive 0 flags 1
tape        xt1 at xtc1 drive 0 flags 1
#
# Support for 2 Sun Archive 1/4" tape controller boards.
#
device      ar0 at mbio ? csr 0x200 priority 3
device      ar1 at mbio ? csr 0x208 priority 3
#
# Support for the GP/GP+/GP2 graphics processors.
# Requires cgtwo as well.
#
device      gpone0 at vme24 ? csr 0x210000
#
# Support for either the Sun-2 color board, Sun-3 color board,
# or GP2 frame buffer.
#
device      cgtwo0 at vme24 ? csr 0x400000 priority 4 vector cgtwointr 0xa8
#
# Support for the Sun-1 color board.
#
device      cgone0 at mbmem ? csr 0xec000 priority 3
#
# Support for monochrome memory frame buffers on various machines.
#
device      bwtwo0 at obmem 1 csr 0x700000 priority 4   # 2/120
device      bwtwo0 at obio 2 csr 0x0 priority 4     # 2/50
device      bwone0 at mbmem ? csr 0xc0000 priority 3    # 1/100U
#
# Support for the Ikon Versatec printer controller.
#
device      vp0 at mbio ? csr 0x400 priority 2
#
# Support for 2 Systech VPC-2200 line printer controllers.
#
device      vpc0 at mbio ? csr 0x480 priority 2
device      vpc0 at vme16 ? csr 0x480 priority 2 vector vpcintr 0x80
device      vpc1 at mbio ? csr 0x500 priority 2
device      vpc1 at vme16 ? csr 0x500 priority 2 vector vpcintr 0x81
#
# Support for the parallel keyboard/mouse interface on the Sun-2/120
# cpu board.  Required if using the Sun-1 style parallel keyboard or
# mouse.
#
device      pi0 at obio 1 csr 0x1800
#
# Support for the hardware Data Ciphering Processor (aka the DES chip).
# Suggested if you make heavy use of secure RPC or secure NFS.
#
```

```
device      des0 at obio 1 csr 0x1000
device      des0 at obio 2 csr 0x7f1000
#
# Support for the time-of-day clock on the Sun-2/120 CPU board.
#
device      tod0 at obio 1 csr 0x3800
#
# Support for the time-of-day clock on the VME Sun-2 SCSI board.
#
device      tod0 at vme24 ? csr 0x200800
```

## The Sun-3 GENERIC Kernel

The following is the GENERIC configuration file for a Sun-3.

```
#
# @(#)GENERIC 1.82 88/02/08 SMI
#
# This config file describes a generic Sun-3 kernel, including all
# possible standard devices and software options.
#
# The following lines include support for all Sun-3 cpu types.
# There is little to be gained by removing support for particular
# cpu's, so you may as well leave them all in.
#
machine     "sun3"
cpu     "SUN3_160"  # Sun-3/75, Sun-3/140, Sun-3/160, or Sun-3/180
cpu     "SUN3_50"   # Sun-3/50
cpu     "SUN3_260"  # Sun-3/260 or Sun-3/280
cpu     "SUN3_110"  # Sun-3/110
cpu     "SUN3_60"   # Sun-3/60
cpu     "SUN3_E"    # Sun-3E (Eurocard VMEbus cpu)
#
# Name this kernel "GENERIC".
#
ident       GENERIC
#
# This kernel supports about eight users.  Count one
# user for each timesharing user, one for each window
# that you typically use, and one for each diskless
# client you serve.  This is only an approximation
# used to control the size of various kernel data
# structures, not a hard limit.
#
maxusers    8

#
# Include all possible software options.
#
# The INET option is not really optional, every kernel must include it.
#
options     INET        # basic networking support - mandatory
#
# The following options are all filesystem related.  You only need
# QUOTA if you have UFS.  You only need UFS if you have a disk.
# Diskless machines can remove QUOTA, UFS, and NFSSERVER.  LOFS is
# only needed if you're using the Sun Network Software Environment.
#
options     QUOTA       # disk quotas for local disks
options     UFS     # filesystem code for local disks
options     NFSCLIENT   # NFS client side code
options     NFSSERVER   # NFS server side code
options     LOFS        # loopback filesystem - needed by NSE
#
```

```
# The following options are for accounting and auditing.  SYSAUDIT
# should be removed unless you are using the C2 security features.
#
options     SYSACCT     # process accounting, see acct(2) & sa(8)
options     SYSAUDIT    # C2 auditing for security
#
# The following options are for various System V IPC facilities.
# No standard software needs them, although some third party
# software relies on at least IPCSHMEM.
#
options     IPCMESSAGE  # System V IPC message facility
options     IPCSEMAPHORE    # System V IPC semaphore facility
options     IPCSHMEM    # System V IPC shared-memory facility
#
# The following option is only needed if you want to use the trpt
# command to debug TCP problems.
#
options     TCPDEBUG    # TCP debugging, see trpt(8)
#
# The following option includes the software DES support, needed if
# you're using secure NFS or secure RPC and you don't have a DES chip.
#
options     CRYPT       # software encryption (if no DES chip)

#
# Build one kernel based on this basic configuration.
# It will use the generic swap code so that you can
# your root filesystem and swap space on any supported device.
# Put the kernel configured this way in a file named "vmunix".
#
config      vmunix      swap generic

#
# Include support for all possible pseudo-devices.
#
# The first few are mostly concerned with networking.
# You should probably always leave these in.
#
pseudo-device   pty     # pseudo-tty's, also needed for SunView
pseudo-device   ether       # basic Ethernet support
pseudo-device   loop        # loopback network - mandatory
#
# The next few are for SunWindows support, needed to run SunView 1.
#
pseudo-device   win128      # window devices, allow 128 windows
pseudo-device   dtop4       # desktops (screens), allow 4
pseudo-device   ms3     # mouse support, allow 3 mice
#
# The following is needed to support the Sun keyboard, with or
# without the window system.
#
pseudo-device   kb3     # keyboard support, allow 3 keyboards
#
```

```
# The following is for asynchronous tty support for the ALM-2 (aka MCP).
# If you have an ALM-2 (MCP) and it is being used to connect timesharing
# terminals, you will need this.
#
pseudo-device   mcpa64
#
# The following is for the streams pipe device.  Currently nothing
# depends on this device so it is entirely optional.
#
pseudo-device   sp
#
# The following are for streams NIT support.  NIT is used by
# etherfind, traffic, rarpd, and ndbootd.  As a rule of thumb,
# NIT is almost always needed on a server and almost never
# needed on a diskless client.
#
pseudo-device   snit        # streams NIT
pseudo-device   pf      # packet filter
pseudo-device   nbuf        # NIT buffering module
#
# The following is for the "clone" device, used with streams devices.
# This is required if you include streams NIT support.
#
pseudo-device   clone

#
# The following sections describe what kinds of busses each
# cpu type supports.  You should never need to change this.
# (The word "nexus" is historical...)
#

# connections for machine type 1 (SUN3_160)
controller  virtual 1 at nexus ?    # virtually addressed devices
controller  obmem 1 at nexus ?  # memory-like devices on the cpu board
controller  obio 1 at nexus ?   # I/O devices on the cpu board
controller  vme16d16 1 at nexus ?   # VME 16 bit address 16 bit data devices
controller  vme24d16 1 at nexus ?   # VME 24 bit address 16 bit data devices
controller  vme32d16 1 at nexus ?   # VME 32 bit address 16 bit data devices
controller  vme16d32 1 at nexus ?   # VME 16 bit address 32 bit data devices
controller  vme24d32 1 at nexus ?   # VME 24 bit address 32 bit data devices
controller  vme32d32 1 at nexus ?   # VME 32 bit address 32 bit data devices

# connections for machine type 2 (SUN3_50)
controller  virtual 2 at nexus ?
controller  obmem 2 at nexus ?
controller  obio 2 at nexus ?

# connections for machine type 3 (SUN3_260)
controller  virtual 3 at nexus ?
controller  obmem 3 at nexus ?
controller  obio 3 at nexus ?
controller  vme16d16 3 at nexus ?
controller  vme24d16 3 at nexus ?
controller  vme32d16 3 at nexus ?
controller  vme16d32 3 at nexus ?
controller  vme24d32 3 at nexus ?
controller  vme32d32 3 at nexus ?

# connections for machine type 4 (SUN3_110)
controller  virtual 4 at nexus ?
controller  obmem 4 at nexus ?
controller  obio 4 at nexus ?
controller  vme16d16 4 at nexus ?
controller  vme24d16 4 at nexus ?
controller  vme32d16 4 at nexus ?
controller  vme16d32 4 at nexus ?
controller  vme24d32 4 at nexus ?
controller  vme32d32 4 at nexus ?

# connections for machine type 7 (SUN3_60)
controller  virtual 7 at nexus ?
controller  obmem 7 at nexus ?
controller  obio 7 at nexus ?

# connections for machine type 8 (SUN3_E)
controller      virtual 8 at nexus ?
controller      obmem 8 at nexus ?
controller      obio 8 at nexus ?
controller      vme16d16 8 at nexus ?
controller      vme24d16 8 at nexus ?
controller      vme32d16 8 at nexus ?
controller      vme16d32 8 at nexus ?
controller      vme24d32 8 at nexus ?
controller      vme32d32 8 at nexus ?

#
# The following (large) section describes which standard devices this
# kernel supports.
#

#
# Support for 4 Xylogics 7053 controllers with 4 drives each.
#
controller      xdc0 at vme16d32 ? csr 0xee80 priority 2 vector xdintr 0x44
controller      xdc1 at vme16d32 ? csr 0xee90 priority 2 vector xdintr 0x45
controller      xdc2 at vme16d32 ? csr 0xeea0 priority 2 vector xdintr 0x46
controller      xdc3 at vme16d32 ? csr 0xeeb0 priority 2 vector xdintr 0x47
disk        xd0 at xdc0 drive 0
disk        xd1 at xdc0 drive 1
disk        xd2 at xdc0 drive 2
disk        xd3 at xdc0 drive 3
disk        xd4 at xdc1 drive 0
disk        xd5 at xdc1 drive 1
disk        xd6 at xdc1 drive 2
disk        xd7 at xdc1 drive 3
disk        xd8 at xdc2 drive 0
disk         xd9 at xdc2 drive 1
disk         xd10 at xdc2 drive 2
disk         xd11 at xdc2 drive 3
disk         xd12 at xdc3 drive 0
disk         xd13 at xdc3 drive 1
disk         xd14 at xdc3 drive 2
disk         xd15 at xdc3 drive 3
#
# Support for 2 Xylogics 450/451 controllers with 2 drives each.
#
controller   xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller   xyc1 at vme16d16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk         xy0 at xyc0 drive 0
disk         xy1 at xyc0 drive 1
disk         xy2 at xyc1 drive 0
disk         xy3 at xyc1 drive 1
#
# Support for the SCSI-2 host adapter with 2 disks and 1 1/4" tape
# on the first SCSI controller, and 2 disks and 1 1/4" tape on the
# second SCSI controller.
#
controller   sc0 at vme24d16 ? csr 0x200000 priority 2 vector scintr 0x40
disk         sd0 at sc0 drive 0 flags 0
disk         sd1 at sc0 drive 1 flags 0
disk         sd2 at sc0 drive 8 flags 0
disk         sd3 at sc0 drive 9 flags 0
tape         st0 at sc0 drive 32 flags 1
tape         st1 at sc0 drive 40 flags 1
#disk        sf0 at sc0 drive 8 flags 2
#
# Support for the SCSI-3 host adapter and the on-board SCSI controller
# on several machines (e.g. 3/50).  Same device support as above.
#
controller   si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
controller   si0 at obio ? csr 0x140000 priority 2
disk         sd0 at si0 drive 0 flags 0
disk         sd1 at si0 drive 1 flags 0
disk         sd2 at si0 drive 8 flags 0
disk         sd3 at si0 drive 9 flags 0
tape         st0 at si0 drive 32 flags 1
tape         st1 at si0 drive 40 flags 1
#disk        sf0 at si0 drive 8 flags 2
#
# Support for the SCSI-E host adapter used with the Sun-3/E.
# Same device support as above.
#
controller       se0 at vme24d16 ? csr 0x300000 priority 2 vector se_intr 0x40
disk         sd0 at se0 drive 0 flags 0
disk         sd1 at se0 drive 1 flags 0
disk         sd2 at se0 drive 8 flags 0
disk         sd3 at se0 drive 9 flags 0
tape         st0 at se0 drive 32 flags 1
tape         st1 at se0 drive 40 flags 1
```

```
#
# Support for the 2 tty lines (ttya, ttyb) on the cpu board.
# Needed when using a terminal for the console device.
# Flags=3 says to supply carrier in software for both lines.
#
device      zs0 at obio ? csr 0x20000 flags 3 priority 3
#
# Support for the keyboard and mouse interface.  Needed when
# using a frame buffer as the console device or with SunView.
# You can remove this line if you don't use the standard Sun
# Workstation keyboard and mouse, but if you leave it in don't
# change it.
#
device      zs1 at obio ? csr 0x00000 flags 0x103 priority 3
#
# Support for 4 ALM's (Systech MTI-800/1600).  Flags set for
# all lines to be local, i.e., carrier supplied by software
# rather than by the device.
#
device      mti0 at vme16d16 ? csr 0x620 flags 0xffff priority 4
    vector mtiintr 0x88
device      mti1 at vme16d16 ? csr 0x640 flags 0xffff priority 4
    vector mtiintr 0x89
device      mti2 at vme16d16 ? csr 0x660 flags 0xffff priority 4
    vector mtiintr 0x8a
device      mti3 at vme16d16 ? csr 0x680 flags 0xffff priority 4
    vector mtiintr 0x8b
#
# Support for 4 MCP boards.
# Note that the MCP's use the same vectors as the ALM's and thus
# there can be a maximum of 4 (total) MCP's and ALM's.
#
device      mcp0 at vme32d32 ? csr 0x01000000 flags 0x1ffff priority 4
    vector mcpintr 0x8b
device      mcp1 at vme32d32 ? csr 0x01010000 flags 0x1ffff priority 4
    vector mcpintr 0x8a
device      mcp2 at vme32d32 ? csr 0x01020000 flags 0x1ffff priority 4
    vector mcpintr 0x89
device      mcp3 at vme32d32 ? csr 0x01030000 flags 0x1ffff priority 4
    vector mcpintr 0x88
#
# Support for the on-board Intel 82586 Ethernet chip on many machines
# (all but 3/50, 3/60, and 3/E).
#
device      ie0 at obio ? csr 0xc0000 priority 3
#
# Support for the Sun-3/E Intel Ethernet board.
#
device      ie0 at vme24d16 ? csr 0x31ff02 priority 3 vector ieintr 0x74
#
# Support for a second Intel Ethernet board, either a second
# Sun-3/E board or a Multibus Ethernet board used with a
# Multibus-to-VME adapter.  Used for Ethernet to Ethernet
# gateways.
#
device      ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75
#
# Support for the on-board LANCE Ethernet chip on the 3/50 and 3/60.
#
device      le0 at obio ? csr 0x120000 priority 3
#
# Support for 2 Ciprico TapeMaster tape controllers with 1 tape drive each.
#
controller  tm0 at vme16d16 ? csr 0xa0 priority 3 vector tmintr 0x60
controller  tm1 at vme16d16 ? csr 0xa2 priority 3 vector tmintr 0x61
tape        mt0 at tm0 drive 0 flags 1
tape        mt1 at tm1 drive 0 flags 1
#
# Support for 2 Xylogics 472 tape controllers with 1 tape drive each.
#
controller  xtc0 at vme16d16 ? csr 0xee60 priority 3 vector xtintr 0x64
controller  xtc1 at vme16d16 ? csr 0xee68 priority 3 vector xtintr 0x65
tape        xt0 at xtc0 drive 0 flags 1
tape        xt1 at xtc1 drive 0 flags 1
#
# Support for the GP/GP+/GP2 graphics processors.
# Requires cgtwo as well.
#
device      gpone0 at vme24d16 ? csr 0x210000   # GP or GP+
device      gpone0 at vme24d32 ? csr 0x240000   # GP2
#
# Support for either the Sun-2 color board, Sun-3 color board,
# or GP2 frame buffer.
#
device      cgtwo0 at vme24d16 ? csr 0x400000 priority 4
            vector cgtwointr 0xa8
#
# Support for color memory frame buffers on various machine types.
#
# 3/110 on-board frame buffer
device      cgfour0 at obmem 4 csr 0xff000000 priority 4    # 3/110
# 3/60 P4 color frame buffer
device      cgfour0 at obmem 7 csr 0xff300000 priority 4    # 3/60
# 3/60 plug-in color frame buffer
device      cgfour0 at obmem 7 csr 0xff400000 priority 4    # 3/60
#
# Support for monochrome memory frame buffers on various machines.
#
device      bwtwo0 at obmem 1 csr 0xff000000 priority 4 # 3/160
device      bwtwo0 at obmem 2 csr 0x100000 priority 4   # 3/50
device      bwtwo0 at obmem 3 csr 0xff000000 priority 4 # 3/260
# 3/110 on-board frame buffer overlay plane
device      bwtwo0 at obmem 4 csr 0xff000000        # 3/110
device      bwtwo0 at obmem 7 csr 0xff000000 priority 4 # 3/60
device      bwtwo0 at obmem 8 csr 0x1000000         # 3/E
# 3/60 P4 color frame buffer overlay plane, or P4 monochrome frame buffer
```

```
device      bwtwo1 at obmem 7 csr 0xff300000 priority 4 # 3/60
# 3/60 plug-in color frame buffer overlay plane
device      bwtwo1 at obmem 7 csr 0xff400000        # 3/60
#
# Support for the TAAC-1 Application Accelerator.
#
device      taac0 at vme32d32 ? csr 0x28000000
#
# Support for 2 Systech VPC-2200 line printer controllers.
#
device      vpc0 at vme16d16 ? csr 0x480 priority 2 vector vpcintr 0x80
device      vpc1 at vme16d16 ? csr 0x500 priority 2 vector vpcintr 0x81
#
# Support for the hardware Data Ciphering Processor (aka the DES chip).
# Suggested if you make heavy use of secure RPC or secure NFS.
#
device      des0 at obio ? csr 0x1c0000
#
# Support for the Floating Point Accelerator.
#
device      fpa0 at virtual ? csr 0xe0000000
```

## The Sun-4 GENERIC Kernel

The following is a GENERIC configuration file for a Sun-4.

```
#
# @(#)GENERIC 1.28 88/02/08 SMI
#
# This config file describes a generic Sun-4 kernel, including all
# possible standard devices and software options.
#
# The following lines include support for all Sun-4 cpu types.
# There is little to be gained by removing support for particular
# cpu's, so you may as well leave them all in.
#
machine     "sun4"
cpu     "SUN4_260"  # Sun-4/260, Sun-4/280
cpu     "SUN4_110"  # Sun-4/110
#
# Name this kernel "GENERIC".
#
ident       GENERIC
#
# This kernel supports about eight users.  Count one
# user for each timesharing user, one for each window
# that you typically use, and one for each diskless
# client you serve.  This is only an approximation
# used to control the size of various kernel data
# structures, not a hard limit.
#
maxusers    8

#
# Include all possible software options.
#
# The INET option is not really optional, every kernel must include it.
#
options     INET        # basic networking support - mandatory
#
# The following options are all filesystem related.  You only need
# QUOTA if you have UFS.  You only need UFS if you have a disk.
# Diskless machines can remove QUOTA, UFS, and NFSSERVER.  LOFS is
# only needed if you're using the Sun Network Software Environment.
#
options     QUOTA       # disk quotas for local disks
options     UFS     # filesystem code for local disks
options     NFSCLIENT   # NFS client side code
options     NFSSERVER   # NFS server side code
options     LOFS        # loopback filesystem - needed by NSE
#
# The following options are for accounting and auditing.  SYSAUDIT
# should be removed unless you are using the C2 security features.
#
options     SYSACCT     # process accounting, see acct(2) & sa(8)
options     SYSAUDIT    # C2 auditing for security
#
# The following options are for various System V IPC facilities.
# No standard software needs them, although some third party
# software relies on at least IPCSHMEM.
#
options     IPCMESSAGE  # System V IPC message facility
options     IPCSEMAPHORE    # System V IPC semaphore facility
options     IPCSHMEM    # System V IPC shared-memory facility
#
# The following option is only needed if you want to use the trpt
# command to debug TCP problems.
#
options     TCPDEBUG    # TCP debugging, see trpt(8)
#
# The following option includes the software DES support, needed if
# you're using secure NFS or secure RPC and you don't have a DES chip.
#
options     CRYPT       # software encryption (if no DES chip)

#
# Build one kernel based on this basic configuration.
# It will use the generic swap code so that you can have
# your root filesystem and swap space on any supported device.
# Put the kernel configured this way in a file named "vmunix".
#
config      vmunix      swap generic

#
# Include support for all possible pseudo-devices.
#
# The first few are mostly concerned with networking.
# You should probably always leave these in.
#
pseudo-device   pty     # pseudo-tty's, also needed for SunView
pseudo-device   ether       # basic Ethernet support
pseudo-device   loop        # loopback network - mandatory
#
# The next few are for SunWindows support, needed to run SunView 1.
#
pseudo-device   win128      # window devices, allow 128 windows
pseudo-device   dtop4       # desktops (screens), allow 4
pseudo-device   ms3     # mouse support, allow 3 mice
#
# The following is needed to support the Sun keyboard, with or
# without the window system.
#
pseudo-device   kb3     # keyboard support, allow 3 keyboards
#
# The following is for asynchronous tty support for the ALM-2 (aka MCP).
# If you have an ALM-2 (MCP) and it is being used to connect timesharing
# terminals, you will need this.
#
```

```
pseudo-device   mcpa64
#
# The following is for the streams pipe device.  Currently nothing
# depends on this device so it is entirely optional.
#
pseudo-device   sp
#
# The following are for streams NIT support.  NIT is used by
# etherfind, traffic, rarpd, and ndbootd.  As a rule of thumb,
# NIT is almost always needed on a server and almost never
# needed on a diskless client.
#
pseudo-device   snit        # streams NIT
pseudo-device   pf      # packet filter
pseudo-device   nbuf        # NIT buffering module
#
# The following is for the "clone" device, used with streams devices.
# This is required if you include streams NIT support.
#
pseudo-device   clone

#
# The following sections describe what kinds of busses each
# cpu type supports.  You should never need to change this.
# (The word "nexus" is historical...)
#

# connections for machine type 1 (SUN4_260)
controller  obmem 1 at nexus ?  # memory-like devices on the cpu board
controller  obio 1 at nexus ?   # I/O devices on the cpu board
controller  vme16d16 1 at nexus ?   # VME 16 bit address 16 bit data devices
controller  vme24d16 1 at nexus ?   # VME 24 bit address 16 bit data devices
controller  vme32d16 1 at nexus ?   # VME 32 bit address 16 bit data devices
controller  vme16d32 1 at nexus ?   # VME 16 bit address 32 bit data devices
controller  vme24d32 1 at nexus ?   # VME 24 bit address 32 bit data devices
controller  vme32d32 1 at nexus ?   # VME 32 bit address 32 bit data devices

# connections for machine type 2 (SUN4_110)
controller  obmem 2 at nexus ?
controller  obio 2 at nexus ?
controller  vme16d16 2 at nexus ?
controller  vme24d16 2 at nexus ?
controller  vme32d16 2 at nexus ?
controller  vme16d32 2 at nexus ?
controller  vme24d32 2 at nexus ?
controller  vme32d32 2 at nexus ?

#
# The following (large) section describes which standard devices this
# kernel supports.
#

#
```

```
# Support for 4 Xylogics 7053 controllers with 4 drives each.
#
controller  xdc0 at vme16d32 ? csr 0xee80 priority 2 vector xdintr 0x44
controller  xdc1 at vme16d32 ? csr 0xee90 priority 2 vector xdintr 0x45
controller  xdc2 at vme16d32 ? csr 0xeea0 priority 2 vector xdintr 0x46
controller  xdc3 at vme16d32 ? csr 0xeeb0 priority 2 vector xdintr 0x47
disk        xd0 at xdc0 drive 0
disk        xd1 at xdc0 drive 1
disk        xd2 at xdc0 drive 2
disk        xd3 at xdc0 drive 3
disk        xd4 at xdc1 drive 0
disk        xd5 at xdc1 drive 1
disk        xd6 at xdc1 drive 2
disk        xd7 at xdc1 drive 3
disk        xd8 at xdc2 drive 0
disk        xd9 at xdc2 drive 1
disk        xd10 at xdc2 drive 2
disk        xd11 at xdc2 drive 3
disk        xd12 at xdc3 drive 0
disk        xd13 at xdc3 drive 1
disk        xd14 at xdc3 drive 2
disk        xd15 at xdc3 drive 3
#
# Support for 2 Xylogics 450/451 controllers with 2 drives each.
#
controller  xyc0 at vme16d16 ? csr 0xee40 priority 2 vector xyintr 0x48
controller  xyc1 at vme16d16 ? csr 0xee48 priority 2 vector xyintr 0x49
disk        xy0 at xyc0 drive 0
disk        xy1 at xyc0 drive 1
disk        xy2 at xyc1 drive 0
disk        xy3 at xyc1 drive 1
#
# Support for the SCSI-2 host adapter with 2 disks and 1 1/4" tape
# on the first SCSI controller, and 2 disks and 1 1/4" tape on the
# second SCSI controller.
#
controller  sc0 at vme24d16 ? csr 0x200000 priority 2 vector scintr 0x40
disk        sd0 at sc0 drive 0 flags 0
disk        sd1 at sc0 drive 1 flags 0
disk        sd2 at sc0 drive 8 flags 0
disk        sd3 at sc0 drive 9 flags 0
tape        st0 at sc0 drive 32 flags 1
tape        st1 at sc0 drive 40 flags 1
#disk       sf0 at sc0 drive 8 flags 2
#
# Support for the SCSI-3 host adapter.  Same device support as above.
#
controller  si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
disk        sd0 at si0 drive 0 flags 0
disk        sd1 at si0 drive 1 flags 0
disk        sd2 at si0 drive 8 flags 0
disk        sd3 at si0 drive 9 flags 0
tape        st0 at si0 drive 32 flags 1
tape        st1 at si0 drive 40 flags 1
#disk       sf0 at si0 drive 8 flags 2
#
# Support for the "SCSI weird" host adapter used with the Sun-4/110.
# Same device support as above.
#
controller  sw0 at obio 2 csr 0xa000000 priority 2
disk        sd0 at sw0 drive 0 flags 0
disk        sd1 at sw0 drive 1 flags 0
tape        st0 at sw0 drive 32 flags 1
disk        sd2 at sw0 drive 8 flags 0
disk        sd3 at sw0 drive 9 flags 0
tape        st1 at sw0 drive 40 flags 1
#disk       sf0 at sw0 drive 8 flags 2
#
# Support for the 2 tty lines (ttya, ttyb) on the cpu board.
# Needed when using a terminal for the console device.
# Flags=3 says to supply carrier in software for both lines.
#
device      zs0 at obio ? csr 0xf1000000 flags 3 priority 3
#
# Support for the keyboard and mouse interface.  Needed when
# using a frame buffer as the console device or with SunView.
# You can remove this line if you don't use the standard Sun
# Workstation keyboard and mouse, but if you leave it in don't
# change it.
#
device      zs1 at obio ? csr 0xf0000000 flags 0x103 priority 3
#
# Support for 4 ALM's (Systech MTI-800/1600).  Flags set for
# all lines to be local, i.e., carrier supplied by software
# rather than by the device.
#
device      mti0 at vme16d16 ? csr 0x620 flags 0xffff priority 4
    vector mtiintr 0x88
device      mti1 at vme16d16 ? csr 0x640 flags 0xffff priority 4
    vector mtiintr 0x89
device      mti2 at vme16d16 ? csr 0x660 flags 0xffff priority 4
    vector mtiintr 0x8a
device      mti3 at vme16d16 ? csr 0x680 flags 0xffff priority 4
    vector mtiintr 0x8b
#
# Support for 4 MCP boards.
# Note that the MCP's use the same vectors as the ALM's and thus
# there can be a maximum of 4 (total) MCP's and ALM's.
#
device      mcp0 at vme32d32 ? csr 0x01000000 flags 0x1ffff priority 4
    vector mcpintr 0x8b
device      mcp1 at vme32d32 ? csr 0x01010000 flags 0x1ffff priority 4
    vector mcpintr 0x8a
device      mcp2 at vme32d32 ? csr 0x01020000 flags 0x1ffff priority 4
    vector mcpintr 0x89
device      mcp3 at vme32d32 ? csr 0x01030000 flags 0x1ffff priority 4
```

```
    vector mcpintr 0x88
#
# Support for the on-board Intel 82586 Ethernet chip on many machines.
#
device      ie0 at obio ? csr 0xf6000000 priority 3
#
# Support for a second Intel Ethernet board, either a second
# Sun-3/E board or a Multibus Ethernet board used with a
# Multibus-to-VME adapter.  Used for Ethernet to Ethernet
# gateways.
#
device      ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75
#
# Support for 2 Xylogics 472 tape controllers with 1 tape drive each.
#
controller  xtc0 at vme16d16 ? csr 0xee60 priority 3 vector xtintr 0x64
controller  xtc1 at vme16d16 ? csr 0xee68 priority 3 vector xtintr 0x65
tape        xt0 at xtc0 drive 0 flags 1
tape        xt1 at xtc1 drive 0 flags 1
#
# Support for the GP/GP+/GP2 graphics processors.
# Requires cgtwo as well.
#
device      gpone0 at vme24d16 ? csr 0x210000   # GP or GP+
device      gpone0 at vme24d32 ? csr 0x240000   # GP2
#
# Support for either the Sun-2 color board, Sun-3 color board,
# or GP2 frame buffer.
#
device      cgtwo0 at vme24d16 ? csr 0x400000 priority 4
            vector cgtwointr 0xa8
#
# Support for color memory frame buffers on various machine types.
#
# 4/110 on-board frame buffer
device      cgfour0 at obio 2 csr 0xfb300000 priority 4 # 4/110
#
# Support for monochrome memory frame buffers on various machines.
#
device      bwtwo0 at obio 1 csr 0xfd000000 priority 4  # 4/260
# 4/110 on-board frame buffer overlay plane
device      bwtwo0 at obio 2 csr 0xfb300000 priority 4  # 4/110
#
# Support for the TAAC-1 Application Accelerator.
#
device      taac0 at vme32d32 ? csr 0x28000000
#
# Support for 2 Systech VPC-2200 line printer controllers.
#
device      vpc0 at vme16d16 ? csr 0x480 priority 2 vector vpcintr 0x80
device      vpc1 at vme16d16 ? csr 0x500 priority 2 vector vpcintr 0x81
#
# Support for the hardware Data Ciphering Processor (aka the DES chip).
# Suggested if you make heavy use of secure RPC or secure NFS.
#
device      des0 at obio ? csr 0xfe000000
```

## 9.4. Procedures for Reconfiguring the Kernel

This section contains instructions for creating the new kernel once you have modified the kernel configuration file. In order to perform these steps, you should have done the following:

- □ Installed Release 4.0 by using the `suninstall` program.
- □ Logged in as superuser.
- □ Created the file `/usr/share/sys/sun`*[2,3,4]*`conf/`*SYS_NAME* from the GENERIC or other template configuration file.
- □ Modified the `SYS_NAME` file according to the instructions in the previous section, and changed the file's permissions.

Two sets of instructions follow: one for standalones and one for servers and their clients. You should refer to the set that applies to your configuration.

### Kernel Reconfiguration for Standalone Systems

For standalone machines, proceed as follows.

1. Go to the directory `/usr/share/sys/sun`[*2,3,4*]`/conf` if you are not there already:

```
# cd /usr/share/sys/sun[2,3,4]/conf
```

2. Run `/usr/etc/config`. Then change to the new configuration directory, and make the new system as shown below. (Remember to substitute your actual system image name for *SYS_NAME*):

```
# /usr/etc/config SYS_NAME
# cd ../SYS_NAME
# make
[ lots of output ]
```

3. Now save the old kernel and install the new one as follows:

```
# mv /vmunix /vmunix.old
# cp vmunix /vmunix
# /usr/etc/shutdown -h now

                The system goes through the halt sequence, then
                the monitor displays its prompt, at which point you
                can boot the system:

> b vmunix
```

>b xy(0,0,0) VMUNIX

4. If the system appears to work, this completes the upgrade procedure. If the new kernel doesn't seem function properly, boot `/vmunix.old`, copy it back to `/vmunix`, and fix your new kernel as follows:

```
# /usr/etc/shutdown -h now
> b vmunix.old -s
# mv /vmunix /vmunix.oops
# mv /vmunix.old /vmunix
# ^D        [ Brings the system up multi-user ]
```

It is advisable to keep a copy of the distributed GENERIC kernel available in case you have problems with any reconfigured kernel.

## Kernel Reconfiguration for Servers and Their Clients

For server machines, proceed as follows.

1. Go to `/usr/share/sys/sun`[*2,3,4*]`/conf` if you are not there already:

```
# cd /usr/share/sys/sun[2,3,4]/conf
```

2. Run `/usr/etc/config`. Then change to the new configuration directory, and make the new system. (Remember to substitute your actual system image name for *SYS_NAME*):

```
# /usr/etc/config SYS_NAME
# cd ../SYS_NAME
# make
[ lots of output ]
```

3. Create kernel configuration files for your clients. (If you need help, refer to the previous section for the appropriate annotated configuration file for the client's architecture.) When editing the clients' configuration files (called *CLIENT_KERNEL_NAME* in the following steps), remember to include the entire set of devices used by all the machines:

```
# cd /export/exec/sun[2,3,4]/sys/sun [2,3,4]/conf
# cp TEMPLATE_NAME CLIENT_KERNEL_NAME
# chmod +w CLIENT_KERNEL_NAME
[ Edit CLIENT_KERNEL_NAME to reflect all clients' systems.
  Be especially careful with the device description lines. ]
# /usr/etc/config CLIENT_KERNEL_NAME
# cd ../CLIENT_KERNEL_NAME
# make
[ lots of output ]
```

4. Now you can position yourself in the directory that has the server's kernel in it, save your server's old kernel, install your new one, and try everything out:

```
# cd /usr/sys/sun[2,3,4]SYS_NAME
# mv /vmunix /vmunix.old
# cp vmunix /vmunix
```

5. Next, install the appropriate client kernel under the client's root directory. Note that clients do not have to be halted at this time, but they must reboot in order to run the new kernel. To install the clients' kernel, save the original kernel (if there is one), install the new kernel image in `/export/root/client_name`, and then test it out by booting up one of the clients:

```
# cd /export/exec/sun[2,3,4]/sys/sun[2,3,4]CLIENT_KERNEL_NAME
#cp /export/root/client_name/vmunix /export/root/client_name/vmunix.old
[or wherever your client kernel is]
# mv vmunix /export/root/client_name/vmunix
[ On the client machine : ]
>b vmunix
```

6. Since at this point normal system performance is a highly, but not absolutely, certain indicator of a trouble-free kernel, if your system(s) appears to work you may proceed with some confidence. You have successfully completed installation.

   If, on the other hand, either of the new kernels does not seem to be functioning properly, halt all systems and boot from the original kernel. Then move the faulty kernel away and re-install the original in its place. Once you are booted up on the original, you can go about trying to fix the faulty kernel. For example, on the server type:

```
# /usr/etc/halt
> b vmunix.old -s
# cd /
# mv vmunix vmunix.bad
# mv vmunix.old vmunix
# ^D                    [ Brings the system up multi-user ]
```

   For clients, halt all the clients on the server. You will have to correct the problem from the server.

On the server type:

```
# cd /export/root/client_name
# mv vmunix vmunix.bad
# mv vmunix.old vmunix
```

You may now boot up the clients and allow them to run while or until a new client kernel is made and ready to install; or if the clients can remain down, build and install a new client kernel now.

## 9.5. Changing Swap Space

When you run `suninstall`, by default it sets up swap space for an NFS server or standalone system on Partition B. In addition, it sets up the swap space for clients in `/export/swap/`*client_name* to enable the client machine to swap over NFS. At this time, you can specify the size of the swap partition and client swap files, or have `suninstall` use the default.

After a system is in use, you may find the originally specified swap space for a machine is insufficient. Therefore, you will want to change swap size. To do this on a server, you have to back up all the server's file systems, then rerun `suninstall`.

To increase client swap space, you can use a program called `mkfile`. Its syntax is

```
mkfile [ -nv ] size[k|b|m] filename ...
```

The option `-n` tells `mkfile` to create an empty file; `-v` selects verbose mode, in which the system displays messages about what it is doing. The *size* argument specifies the size you want the file to be. The letters `k`, `b`, and `m` represent Kilobytes, Bytes, and Megabytes respectively. (Recommended swap size for a client machine is 10 Mbytes.) The *filename* argument, when configuring a client's swap space, should be the full pathname of the client machine's `swap` directory.

Here are steps you would take to change swap space for client raks.

1. Log in as superuser on your server machine.
2. Type the following:

```
    # mkfile -n 10m /export/swap/raks
```

   This creates an empty file of 10 Mbytes in length in `/export/swap/raks`.

3. Reboot client raks.

10

# Maintaining Disks with `format`

10

# Maintaining Disks with format

format is a SunOS™ utility that enables you to format, label, repair and analyze disks on your Sun system. Unlike previous disk maintenance programs, format runs under SunOS. Because there are limitations to what can be done to the system disk while SunOS is running, format is also supported in the standalone MUNIX environment. However, for most applications, running format under SunOS is sufficient and far more convenient.

format provides a friendly, menu based interface for disk maintenance. For most disk problems, the instructions in this manual can be followed verbatim to achieve the desired results. A detailed understanding of disk hardware is no longer necessary to perform basic maintenance functions.

## 10.1. An Overview of format

You use format to set up a disk for use by SunOS. Running format online provides many advantages, such as the ability to format many drives at once, to format a drive on a remote system, and to save defect lists as normal ASCII files. This last feature is especially useful, as it allows you to recover when the defect list has become destroyed or corrupted. It is also possible (and occasionally necessary) to run format in standalone fashion. This is done by running format under MUNIX, a completely memory resident version of SunOS. For more details on this procedure, see *Installing the SunOS*.

One factor remains constant between format and virtually all other disk programs - there is a very real danger of destroying information inadvertently. When you format a drive, all the information on it is erased. For this reason, you should always back up your data before invoking format, especially when you first start using format.

You should familiarize yourself with a few key concepts used by the format utility before you actually use it. The first concept is that of the *defect list*. This list is comprised of the known defective sectors on a disk, and is used during the formatting procedure to mark all defective sectors in the list as unusable. format requires a defect list before formatting a drive. Defect lists reside in a reserved area on a disk drive, so that they will not be accidentally destroyed. However, you should always save the information in your defect lists after doing a format operation as a precaution. You can do this by saving your defect lists as normal ASCII files with the **dump** command under format's "Defect" menu. You should also photocopy and save the manufacturer's defect list that is usually attached to the drive when it is shipped. When it is necessary to read in a saved defect list, you use the **load** command. Later sections in this chapter provide information and examples of what to do when there is no defect list or it has been corrupted.

`format` makes a necessary distinction between a *current* defect list and a *working* defect list. The *current* defect list is read in automatically when you select a current (default) disk, unless the disk is unformatted. When you need to make additions or deletions to the *current* defect list, `format` uses a temporary copy of this list called the *working defect list.* The working list is only used as a temporary copy. When you are satisfied that your changes are valid, you update the current list with the `commit` command. Prior to invoking `commit`, you can undo all the changes you have made to the working list by using the `restore` command.

Another new concept is `format's` data file. By default, `format` looks in the file `/etc/format.dat` for the default disk types and partition tables. Having a data file external to the program makes it easy for you to add your own disk types or partition tables. If you wish to alter the default entries shipped by Sun, you should create alternate names of your own for them, to avoid any possible confusion.

## `format` Do's and Don'ts

Here is a list of actions you should or should not take when using `format`.

DO the following when using `format`:

- Back up all your files before doing anything else. Multiple copies are recommended.
- Save all your defect lists in files by using `format`'s `dump` command. The file name should include the drive type, model number, and serial number in the filename.
- Save the paper copies of the manufacturer's defect list that are shipped with your drive. These can be invaluable in case of disaster.
- Use the command examples in the manual to help you through the procedures. Be especially careful when executing the `type`, `extract`, `commit`, and `format` operations.

DO NOT do the following when using `format`

- Run `extract` when your disk is brand new or has a defect list, as you can destroy valid defect information.
- Change the Sun-supplied entries in the `format.dat` file. If you need slight variations on these, simply add new ones, being careful to give them unique names.
- Edit a defect list by hand. Doing so will invalidate the `checksum` and make the list useless.
- Do destructive surface analysis (that is, `write` or `compare` commands) on cylinder 0 or the last two cylinders. This would overwrite the label or defect list, respectively.

- Do destructive surface analysis on a disk that has mounted partitions.

While this introduction has given you some of the basic concepts involved in the new format program, it is no substitute for reading the rest of this chapter. In most cases, you can use the examples in the text verbatim to achieve the desired results.

## 10.2. Interacting with format

This section describes the user interface of format. The types of input you need to specify are described, as well as some of the features that help you to interact with format.

### Types of Input

Described below are the several types of input that format can require you to select.

#### Numbers

There are several places in format that require an integer as input. You must either specify the data or select one from a list of choices. In either case, the help facility causes format to print the upper and lower limits of the integer expected. Simply enter the number desired. The number is assumed to be in decimal unless a base is explicitly specified as part of the number (ie - 0x for hexadecimal).

The following are examples of integer input.

```
Enter number of passes [2]: 34
Enter number of passes [34] 0xf
```

#### Block Numbers

Whenever you are required to specify a disk block number, there are a couple of ways to input the information.

You can specify the information as an integer representing the logical block number. As described above, you can specify the integer in any base, but the default is decimal. The maximum operator can also be used here to let format select the appropriate value. Logical block format is used by the SunOS disk drivers in error messages. Some examples of this type of input are described later in this section.

The other way to specify a block number is the cylinder/head/sector format. In this format, you must specify explicitly the three logical components of the block number, the cylinder, head, and sector values. These values are still logical, but they allow you to define regions of the disk related to the layout of the media.

The specifics of the cylinder/head/sector format are as follows.

- Any number preceding the first slash is considered to be the cylinder number.
- Any number after the first slash but before the second slash is considered to be the head number.
- Any number following the second slash is considered to be the sector number.

If any of the numbers are not specified, the appropriate value is assumed to be zero. You can also use the maximum operator in place of any of the numbers and let format select the appropriate value. Below are some examples of this type of input.

```
Enter defective block number: 34/2/3
Enter defective block number: 23/1/
Enter defective block number: 457//
Enter defective block number: 12345
Enter defective block number: 0xabcd
Enter defective block number: 334/$/2
Enter defective block number: 892//$
```

Any time format prints a block number, it is printed in both of the above formats. Also, the help facility shows you the upper and lower bounds of the block number expected, in both formats.

### Command Names

Command names are needed as input whenever format is sitting at a menu prompt. You can abbreviate the command names, as long as what is entered is sufficient to uniquely identify the command desired. You can also use the help facility to get a list of the commands available at any time.

### Other Names

There are certain times in format when you must name an object. In these cases, you are free to specify any string desired for the name. If the name has white space in it, the entire name must be enclosed in double quotes (""). Otherwise, only the first word of the name is used.

## Supporting Features in format

### Help Facility

format provides a help facility you can use any time format is expecting input. By simply entering a question mark **?**, you can request information about what is expected. format then prints a brief description of what type of input is needed. If appropriate, the legal boundaries of the input are also given. If you enter a question mark at a menu prompt, format reprints the list of commands that are available.

### Maximum Operator

The maximum operator allows you to easily specify the largest legal number to format. Whenever you enter a block number, you can replace any of the components with a dollar sign **$**. This causes format to assign that component the largest value possible. This is especially useful when you specify a range of block numbers. For instance, if you want to specify all of cylinder 10 for analysis, you could enter the following:

```
Enter starting block number [6700, 10/0/0]: 10//
Enter ending block number [7369, 10/9/66]: 10/$/$
```

**Defaults**

format supports default values for input whenever possible. If format asks you to specify an object with a current value, the current value is always the default for the input. The default value is always displayed in square brackets ('[]') at the end of the input prompt. If you enter just a carriage return, and there is a default value, it is used by format.

## 10.3. Invoking format

The format utility runs under SunOS like any other utility. Whether running under MUNIX or SunOS, the command line used to run format is the same. The command line consists of the following parts:

```
format [options] [disk list]
```

The options all begin with a minus (-). Some of the options must be followed by a value. Note that several options can be grouped after a single minus, followed by the list of values for the options specified. As soon as format encounters something that is not an option or an option value, it assumes the disk list has begun. The rest of the command line is then assumed to be the disk list.

The following options are recognized by format:

-f *command_file*

Use the specified file for input instead of standard input. The file must contain commands in the same format that they would be entered through standard input, with the exception that no continue? messages are printed when running from a file.

-l *log_file*

Use the specified file to log the entire format session. All I/O to standard input, standard output, and error output are placed in the log file.

-x *data_file*

Use the specified file as the data file.

-d *disk_name*

Specifies which disk should be made current upon entry to the program. The disk is specified by its logical name (ie -xy0). This can also be accomplished by specifying a single disk in the disk list.

-t *disk_type*

Specifies the type of the disk made current upon entry to the program. The type is specified by its name as defined in the data file. This option can only be used if a disk is also being specified as described above.

-p *partition_name*

Specifies the partition table for the disk made current upon entry to the program. The table is specified by its name as defined in the data file. This option can only be used if a disk is being specified, and its type is either specified or available from the disk label.

-s

Run in silent mode. All prints to standard output are suppressed. Error messages are still printed. This is generally used in conjunction with the -f option when the commands have already been determined.

The disk list is used to tell format which disks should be searched for. If no disk list is specified, the list in the data file is used. Entries in the disk list are specified in logical format (ie - xy0). If a single entry is specified, that disk will automatically be made the current disk when format is entered.

## 10.4. format's Data File

The format data file allows you to configure format to support your specific system. There are three things that can be defined in the data file -- search paths, disk types, and partition tables. Usually, the data file shipped with Sun systems is sufficient. It contains all the information necessary to support standard Sun disks. You will want to modify the data file for your system if you have one of the following:

- A disk that has a unit number or controller number that is not found in the GENERIC configuration file.
- A disk that is a different model than those supported by Sun.
- A disk with a partition table that is different from the table Sun shipped it with.

Even if one of the above criteria is met, it is possible to run format without modifying the data file. However, certain information would have to be entered while format is running. By modifying the data file to support your configuration, the information will automatically be known to format.

There are several ways to specify the location of your data file to format. If a pathname is given with the -x command line option, that file is always used as the data file. If the -x option is not specified, then format looks in the current directory for a file named format.dat. If the file exists, it is then used as the data file. If neither of these methods yields a data file, then format uses the file /etc/format.dat as the data file. This file is shipped with your Sun system, so it should always be present.

The data file contains definitions that are read in by format when it starts up. Each definition starts with a keyword, showing which of the three types of definitions it is. The three legal keywords in the data file are search_path, disk_type, and partition.

Use the search_path keyword to tell format which disks it should search for when it starts up. The list in the default data file contains all the disks in the GENERIC configuration file. If your system has disks that are not in the GENERIC configuration file, you should add them to the search_path definition in your data file. Unlike the other definitions, the data file can contain only one search_path definition. However, this single definition lets you specify all the disks you have in your system.

Use the disk_type keyword to define a specific controller and disk model so format can operate on such a disk. Each disk_type definition contains information concerning the physical geometry of the disk. The default data file contains definitions for all the controllers and disks that Sun supports. You only need to add a new disk_type if you have a non-Sun disk. You can add as many disk_type definitions to the data file as you want.

Use the partition keyword to define a partition table for a specific disk type. It contains the partitioning information, plus a name that lets you refer to it in format.

The default data file contains partition definitions for all the partition tables that Sun ships disks with. You should add a partition definition if you repartitioned any of the disks on your system. You can add as many partition definitions to the data file as you require.

The syntax of the data file is an expanded version of the termcap syntax. The following rules apply to the data file:

- The hash sign ('#') is the comment character. Any text on a line after a hash sign is not interpreted by format.
- The data file consists of a series of definitions. Each definition appears on a single logical line. This means that if you break the definition across multiple lines in the data file, all but the last line of the definition must end with a backslash ('\'). A carriage return that is not preceded by a backslash indicates the end of a definition.
- A definition consists of a series of assignments. Each assignment has an identifier on the left side and a value(s) on the right side. The assignment operator is the equal sign ('='). The assignments within a definition must be separated by a colon (':').
- White space is ignored by format. You can use it to make the data file more readable, it will all be stripped off. If you want an assignment value to actually contain white space, you can enclose the entire value in double quotes ('"'). This will cause the white space within the quotes to be preserved as part of the assignment value. Because of this feature, it is a good idea to enclose all string values in double quotes, in case you want them to contain white space.
- Some assignments can have multiple values on the right hand side. When there are multiple values, they must be separated by a comma (',').

The search_path definition consists of only one assignment. The keyword itself is assigned a list of disk names. The disk names are as they appear in the boot messages. Here is an example of the search_path definition:

```
#
# This is the search path for format. It contains all the disks that
# will be searched for if no disk list is given on the command line.
#
        search_path = xy0, xy1, xy2, xy3, xd0, xd1, xd2, xd3, xd4, xd5, xd6, \
                xd7, xd8, xd9, xd10, xd11, xd12, xd13, xd14, xd15, sd0, sd1, sd2, sd3
```

A disk_type definition consists of several general assignments and some assignments that are controller specific. The keyword itself is assigned the name of the disk type. This name appears in the disk's label, and is used to identify the disk type whenever format is run. Remember that you should enclose the name in double quotes so any white space in the name is preserved. The following identifiers must also be assigned values in all disk_type definitions:

- □ ctlr - must be assigned a value that represents the controller type the disk type can be attached to. Currently, the supported values for this assignment are XY450 for Xylogics 450/451 controllers, XD7053 for Xylogics 7053 controllers, MD21 for Emulex MD21 controllers, and ACB4000 for Adaptec ACB4000 controllers.
- □ ncyl - must be assigned the number of data cylinders in the disk type. This determines how many logical cylinders of the disk SunOS will be allowed to access.
- □ acyl - must be assigned the number of alternate cylinders in the disk type. These cylinders are used by format to store information such as the defect list for the drive. You should always leave at least two cylinders for alternates.
- □ pcyl - must be assigned the number of physical cylinders in the disk type. This number is used to calculate the boundaries of the disk media. This number is usually equal to ncyl plus acyl, but there are some circumstances under which it is not. For instance, the Emulex MD21 controller requires four cylinders for internal controller use, so they must be left off the other assignments. Also, to make disks field replaceable with second sources, some disks are artificially limited to be the same size as another type.
- □ nhead - must be assigned the number of heads in the disk type. This number is used to calculate the boundaries of the disk media.
- □ nsect - must be assigned the number of data sectors per track in the disk type. This number is used to calculate the boundaries of the disk media. Note that this is only the data sectors, any spares are not reflected in the assignment.
- □ rpm - must be assigned the rotations per minute of the disk type. This information is put in the label and later used by the file system to calculate the optimal placement of file data.
- □ bpt - must be assigned the physical number of bytes per track for the disk type. This number is used to calculate the boundaries for defects that are in bytes from index format.

Other assignments may be necessary depending on which controller the disk type is attached to. For XY450 controllers, the following assignments are also required:

- □ bps - must be assigned the total number of bytes per sector, including the header and gaps, in the disk type. This number is necessary to locate defects within a track. See the disk manual for information on how to calculate this number.
- □ drive_type - must be assigned the drive type of the disk type. The drive type is a number between 0 and 3 that the 450/451 controller uses to identify the disk's geometry. See the controller manual for more information.

For XD7053 controllers, the following assignments are also required:

- □ bps - must be assigned the total number of bytes per sector, including the header and gaps, in the disk type. This number is necessary to locate defects within a track. See the disk manual for information on how to calculate this number.

For ACB4000 controllers, the following assignments are also required:

- □ skew - must be assigned the buffer skew for the disk type.
- □ precomp - cylinder at which to begin write precompensation.

Below are some examples of disk_type definitions.

```
disk_type = "Fujitsu-M2361 Eagle" \
     : ctlr = XY450 : fmt_time = 4 \
     : ncyl = 840 : acyl = 2 : pcyl = 842 : nhead = 20 : nsect = 67 \
     : rpm = 3600 : bpt = 40960 : bps = 600 : drive_type = 3

disk_type = "CDC EMD 9720" \
     : ctlr = XD7053 \
     : ncyl = 1147 : acyl = 2 : pcyl = 1217 : nhead = 10 : nsect = 48 \
     : rpm = 3600 : bpt = 30240 : bps = 613

disk_type = "Micropolis 1355" \
     : ctlr = MD21 \
     : ncyl = 1018 : acyl = 2 : pcyl = 1024 : nhead = 8 : nsect = 34 \
     : rpm = 3600 : bpt = 20832

disk_type = "Fujitsu M2243AS" \
     : ctlr = ACB4000 \
     : ncyl = 752 : acyl = 2 : pcyl = 754 : nhead = 11 : nsect = 17 \
     : rpm = 3600 : bpt = 10416 : skew = 2 : precomp = 754
```

A partition definition contains some general assignments, and some optional assignments. The keyword itself is assigned the name of the partition table. This name is used inside format to identify the table. Remember to enclose the name in double quotes so any white space in the name is preserved. The following identifiers must also be assigned values in all partition definitions:

- □ disk - must be assigned the name of the disk_type that this partition table is defined for. This name must appear exactly as it does in the disk_type definition.

- □ ctlr - must be assigned a value that represents the controller type disks with this partition table can be attached to. Currently, the supported values for this assignment are XY450 for Xylogics 450/451 controllers, XD7053 for Xylogics 7053 controllers, MD21 for Emulex MD21 controllers, and ACB4000 for Adaptec ACB4000 controllers. The controller type specified here must also be defined for the disk_type chosen above.

The other assignments in a partition definition describe the actual partition information. The identifiers are the letters a through h , with each letter representing that partition. These assignments are optional; any partition not explicitly assigned is set to 0 length. The value of each of these assignments is a pair of numbers separated by a comma. The first number is the starting cylinder for the partition, and the second is the number of sectors in the partition. Below are some examples of partition definitions.

```
    partition = "Fujitsu-M2351 Eagle" \
        : disk = "Fujitsu-M2351 Eagle" : ctlr = XY450 \
        : a = 0, 16560 : b = 18, 34040 : c = 0, 772800 : g = 55, 722200

    partition = "Fujitsu-M2351 Eagle Old Type" \
        : disk = "Fujitsu-M2351 Eagle" : ctlr = XY450 \
        : a = 0, 15884 : b = 18, 33440 : c = 0, 772800 : g = 55, 722200

    partition = "Fujitsu-M2351 Eagle" \
        : disk = "Fujitsu-M2351 Eagle" : ctlr = XD7053 \
        : a = 0, 16560 : b = 18, 34040 : c = 0, 772800 : g = 55, 722200

    partition = "Fujitsu-M2351 Eagle Old Type" \
        : disk = "Fujitsu-M2351 Eagle" : ctlr = XD7053 \
        : a = 0, 15884 : b = 18, 33440 : c = 0, 772800 : g = 55, 722200

    partition = "Fujitsu-M2333" \
        : disk = "Fujitsu-M2333" : ctlr = XD7053 \
        : a = 0, 16080 : b = 24, 33500 : c = 0, 550070 : g = 74, 500490

    partition = "Micropolis 1355" \
        : disk = "Micropolis 1355" : ctlr = MD21 \
        : a = 0, 16048 : b = 59, 33456 : c = 0, 276896 : g = 182, 227392

    partition = "Toshiba MK 156F" \
        : disk = "Toshiba MK 156F" : ctlr = MD21 \
        : a = 0, 15980 : b = 47, 33660 : c = 0, 277100 : g = 146, 227460
```

## 10.5. `format` Command Reference

This section describes the commands available in `format`. The user interface is organized as a tree of menus. Each menu is explained separately in the following sections.

### The *Command Menu*

The *command menu* is entered when the `format` program is invoked. It is identified by the *format>* prompt. The following commands are included in the *command menu:*

```
    disk       - select a disk
    type       - select (define) a disk type
    partition  - select (define) a partition table
    current    - describe the current disk
    format     - format and analyze the disk
    repair     - repair a defective sector
    show       - translate a disk address
    label      - write label to the disk
    analyze    - surface analysis
    defect     - defect list management
    backup     - search for backup labels
    quit
```

disk - select a disk

This command is used to select the *current disk*. The *current disk* is the disk that is currently being operated on by `format`. Most of the commands in `format` cannot be run unless a *current disk* is selected. The *current disk* can be set on the command line, so the `disk` command is not always necessary. When `disk` is run, it lists the boot lines for all the disks that were found during the disk search. The disks are numbered, and the *current disk* is selected by entering the number next to the desired disk. If there is already a *current disk* when `disk` is run, it is automatically the default choice. The *current disk* may be changed at will without worrying about harming the state of any of the disks. This allows you to work on several disks in one invocation of `format`. Below is an example of the `disk` command. At the time it was run, `xy0` was the *current disk.*

```
format> disk


AVAILABLE DISK SELECTIONS:
        0. xy0 at xyc0 slave 0
           xy0: <drive type unknown>
        1. xy1 at xyc0 slave 1
           xy1: <Fujitsu-M2333 cyl 821 alt 2 hd 10 sec 67>
        2. xd0 at xdc0 slave 0
           xd0: <drive type unknown>
        3. sd0 at si0 slave 0
           sd0: <drive type unknown>
Specify disk (enter its number) : 3
```

Note the messages displayed after the choice was made. If the disk type is known, it is printed inside the angle brackets ('<>'). If the disk type is not known,

```
    <drive type unknown>
```

is printed. If the type is known, an attempt is made to see if the disk is formatted. The results of this test are shown in the square brackets ('[ ]'). If you know that the result of this test is incorrect, something is wrong. One possibility is that the disk type assigned to the disk is incorrect. Be sure that the type is correct, and fix it with the type command if it is not. If that isn't the problem, there is probably a hardware problem with your system. Exit format immediately and diagnose the problem. Continuing in format only causes damage to the disk. If the disk appears formatted, an attempt is made to read the defect list stored at the back of the disk. The result of this is also displayed in the square brackets. If a defect list was found, the *current defect list* will be set equal to it. If not, the *current defect list* is set to null. All disks that were originally formatted by format should have a defect list on them. Those that were formatted by diag may or may not have a defect list, depending on which version of diag was used. If the selected disk does not have a defect list, you can create one for it. See the section on "Using format for Basic Maintenance" for details.

### type - select (define) a disk type

This command is used to specify the *current disk type.* The *current disk type* describes the physical characteristics of the *current disk.* Most of the commands in format cannot be run unless the *current disk type* is set. The *current disk type* can be set from the command line, and is set automatically if the *current disk* is labeled. Thus, the type command is not always necessary.

When type is run, it lists the disk types that are supported for the *current disk.* It also lists *other* as a choice in case you need to define your own disk type. The types are numbered, and the *current disk type* is selected by entering the number next to the desired disk type. If there is already a *current disk type* when type is run, it will automatically be the default choice.

Exercise extreme caution when using the type command. If the *current disk* is labeled correctly, the *current disk type* is set automatically when the disk is selected. Thus, you should never have to run the type command unless you have a brand new disk or a disk that was incorrectly labeled. Specifying the wrong type for a disk can cause a variety of strange behaviors. In many cases, a slew of errors occus during one of the other commands, because format has the wrong geometry for the disk. Another sign of the wrong disk type is if format claims the disk is unformatted when you know it's formatted. Because it may be hard to correct the situation by the time you discover it, you should always be certain of the correct disk type before selecting it.

Below is an example of the type command. At the time it was run, sd0 was the *current disk.* It's type was unknown, and it was a brand new disk.

```
format> type

AVAILABLE DRIVE TYPES:
        0. Micropolis 1355
        1. Toshiba MK 156F
        2. Micropolis 1558
        3. other
Specify disk type (enter its number) : 2
selecting sd0: <Micropolis 1558>
[disk unformatted, no defect list found]
```

Note that the messages printed after the selection is made are identical to those printed for the disk command. See the previous section for a full explanation.

`partition` - select (define) a partition table

This command is used to enter the *partition menu.* The *partition menu* contains commands to define a partition table and select the *current partition table.* The *current partition table* is used when labeling the *current disk.* More information on the `partition` command is given in a later section.

`current` - describe the current disk

This command displays a brief summary of the *current disk.* It allows you to see which disk is currently selected. The format of the summary is similar to the boot line for a disk. If *current disk* and *current disk type* are both set, the summary will show which disk is selected and describe its physical attributes. If either of these is not set, the summary will reflect that fact. Below are some examples of the `current` command. They show the various summaries that will be printed depending on whether *current disk* and *current disk type* are set.

```
format> current
No Current Disk.

format> current
Current Disk = xd0: <drive type unknown>

format> current
Current Disk = sd0: <Micropolis 1558 cyl 1218 alt 2 hd 15 sec 35>
```

`format` - format the disk

This command is used to format part or all of the *current disk.* Both *current disk* and *current disk type* must be set to run this command. Also, the *current defect list* must be initialized. If the disk is brand new, you must format the whole disk. Failure to do so will leave the disk in a very bad state. After that, the only time formatting is necessary is when the *current defect list* has been modified using commands in the defect menu. In this case, only the portion of the disk affected by the change needs to be reformatted. When `format` is run, it first asks for the bounds of the `format` operation. The default choices for the starting and ending blocks are always the first and last blocks on the *current disk.* Using the default values will cause the entire disk to be formatted. If only part of the disk needs to be formatted, there are some things to keep in mind. First, the starting and ending blocks you specify are both inclusive. This means that both of the sectors specified as the bounds will be included in the `format` operation. For example, if you want to format all of cylinder 10, you should enter:

```
format> format
Enter starting block number [0, 0/0/0]: 10/0/0
Enter ending block number [551409, 822/9/66]: 10/9/66
Ready to format, continue?
```

Note that the ending head and sector numbers specify the last block on the cylinder. You can tell this because they are equal to the default values, which always indicate the last block on the disk. A more convenient way to specify the same bounds would be to enter:

```
format> format
Enter starting block number [0, 0/0/0]: 10/
Enter ending block number [551409, 822/9/66]: 10/$/$
Ready to format, continue?
```

For a complete explanation on how to enter block numbers, see the section on "Interacting with `format`".

Another concern when formatting only part of the *current disk* are the restrictions imposed by your disk controller. Some controllers can only format the entire disk, and others can format only whole tracks. If the bounds you enter do not satisfy the requirements of the controller, an error message is printed and the command aborted.

After the bounds are specified, if `format` is being run interactively instead of from a command file, you are asked to verify that you want to do the `format` operation. Also, if the portion of the disk specified for formatting overlaps any mounted file systems, you are warned and given another chance to cancel the command. It is strongly recommended that you never go ahead with the format under these circumstances. Doing so could cause the system to crash since the mounted file system will be destroyed. This problem cannot occur when running from a command file, since the `disk` command will generate an error if a disk with a mounted file system is selected to be the *current disk.*

At this point, the format operation starts. While the format is in progress, you cannot interrupt the command with a CTRL-C. This is to avoid leaving the disk partially formatted. The specified portion of the disk is formatted, and all defects in the *current defect list* that fall into the formatted area are repaired. If something went wrong, a

```
format failed
```

message is displayed, and the command is aborted. If everything went well, a

```
format succeeded
```

message is displayed. At this point, CTRL-C interrupts are enabled again, so you can stop the command at any time without harming the disk. If the surface analysis parameters are not set up to verify after formatting, the command is finished. However, the default parameters, which are strongly recommended for peace of mind, have verification enabled.

Two passes of the write test will be run over the portion of the disk that was just formatted. Unless you have changed the analysis parameters, automatic repair is enabled during the analysis. This means that any defects found are repaired if possible.

```
format> format
Ready to format. Formatting cannot be interrupted
and takes 11 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
        pass 0 - pattern = 0xc6dec6de
   1217/11/14

        pass 1 - pattern = 0x6db6db6d
   1217/11/14
Total of 0 defective blocks repaired.
```

### repair - repair a defective sector

This command is used to repair a defective sector on the *current disk.* Both *current disk* and *current disk type* must be set to run this command. Also, the *current defect list* must be initialized, and the *current disk* must be formatted. The repair command is optionally supported by a disk controller, so some may not support repairing defective sectors. If you run repair on a controller of this type, it will print an error message and abort the command. In this case, you must use the add command to add the sector to the *current defect list* then reformat the *current disk.*

When repair is run, it first asks for the sector number to be repaired. This is the logical block number of the defective sector. It is typically the block number reported by a SunOS error message, or by a surface analysis command in format. After the block number is specified, if repair is run interactively instead of from a command file, you are asked to verify that you want to do the repair operation. Also, if the sector being repaired is within a mounted file system, you are warned and again given the chance to cancel the command. You should exercise caution repairing a sector in a mounted file system. If the sector is accessed by SunOS while the repair is occurring, the result is unpredictable. Thus, repair should only be run within a mounted file system if the system is idle.

At this point, the repair starts. While the repair is in progress, you cannot interrupt the command with a CTRL-C. This is necessary to insure the integrity of the *current defect list.* If something goes wrong, a

```
   repair failed
```

message is displayed, and the command is aborted. If everything went well, a

```
   repair succeeded
```

message is printed and the defective sector is automatically added to the *current defect list.* SunOS is notified of the new defect, so the repair takes effect immediately. Below are some examples of the repair command.

```
format> repair
Enter block number of defect: 73/0/23
Ready to repair defect, continue? y
Repairing block 35063  (73/0/23)...done.

format> repair
Enter block number of defect: 48997
Ready to repair defect, continue? y
Repairing block 48997  (73/1/20)...done.
```

### `show` - show a disk address

This command is used to show a disk block in several of the formats understood by `format`. Because the disk geometry is necessary to translate the block number, both *current disk* and *current disk type* must be set to run this command. When `show` is run, you are prompted for a disk block number. The block number can be entered in any form understood by `format`. For a full explanation of specifying block numbers, see the section on "Interacting with `format`".

Once the block number is entered, it will be echoed in decimal, hexadecimal, and cylinder/head/sector format. The `show` command allows you to translate the decimal block numbers given in SunOS error messages into cylinder, head, and sector numbers. This may be useful in determining whether your disk is deteriorating in a pattern, such as a specific head going bad. This command in no way affects the *current disk*. It is intended for informational purposes only.

```
format> show
Enter a disk block: 28/9/34
Disk block = 19397 = 0x4bc5 = (28/9/34)
format> show
Enter a disk block: 12345
Disk block = 12345 = 0x3039 = (18/4/17)
format> show
Enter a disk block: 334466
Disk block = 334466 = 0x51a82 = (499/2/2)
format>
```

### `label` - label the disk

This command is used to label the *current disk*. Both *current disk* and *current disk type* must be set to run this command. Also, the *current disk* must be formatted or `label` will fail. When `label` is run, it first checks to see if the *current partition table* is set. If set, the *current partition table* is used when the disk is labeled. If it's not set, `format` uses the first partition table on the list of known tables for the *current disk type*. When this occurs, you are informed that `format` will use the default table. If there are no known partition tables for the *current disk type*, an error message is printed and the command aborted.

At this point, if `format` is being run interactively, you are asked to verify if you want to label the *current disk*. Also, if there are any mounted file systems on the *current disk*, you are warned and again given the chance to cancel the command. You should exercise caution when labeling a disk with file systems on it. If the file systems are mounted, changing the partition information for the partition containing the file system may cause the system to crash. Even if the file system is not mounted, changing its partition information may destroy the files in it. You should always back up the system before relabeling disks with file systems.

After all checks have been completed, the primary and backup labels are written on the *current disk*. The kernel is also notified of the information in the new label, so any changes will take effect immediately. During labeling, CTRL-C interrupts are disabled, so the process cannot be stopped midway. This is necessary to prevent a disk from having only some copies of the label updated.

`analyze` - surface analysis

This command is used to enter the *analyze menu*. The *analyze menu* contains commands to analyze the surface of a disk for media defects. More information on the `analyze` command is given in a later section.

`defect` - defect list management

This command is used to enter the *defect menu*. The *defect menu* contains commands to define and manipulate the *current defect list*. The *current defect list* is used when formatting and repairing the *current disk*. More information on the `defect` command is given in a later section.

`backup` - search for backup labels

This command is used to restore the primary label from backup copies stored on the *current disk*. If the primary label was corrupted, `format` has no way of knowing the disk type or partition table for the disk. You will have to specify the disk type with the `type` command. However, the partition table for the *current disk* may be recovered by searching for backup labels. Since the location of the backup labels is geometry dependent, both *current disk* and *current disk type* must be set to run this command. Also, the *current disk* must be formatted or the command will fail.

When `backup` is run, it will first check to see if the primary label was present on the *current disk*. If it was, it informs you that the disk was labeled, and allows you to cancel the command. If the disk was not labeled, or you choose to go on anyway, an attempt is made to read each of the five `backup` labels stored on the *current disk*. If no good backup label is found, the command ends. If a good copy is found, it is checked against the geometry information of the *current disk* for correctness. If it passes these checks, the information in the label is used to set the *current partition table*. At this point, the primary label is rewritten, and the *current defect list* is copied to the disk. Below are some examples of the `backup` command.

```
format> backup
Disk had a primary label, still continue? y
Searching for backup labels...found.
Restoring primary label and defect list.
format>

format> backup
Searching for backup labels...not found.
format>
```

The fact that `backup` also rewrites the *current defect list* onto the disk is worth discussion. This feature is very useful if the *current disk* has had its defect list corrupted. It allows you to recreate the defect list, then use the `backup` command to write it back to the *current disk.* This will restore the defect list without requiring you to reformat the disk. However, a word of caution is in order. If the *current defect list* does not reflect the actual state of the media when `backup` is run, the defect list on the disk will become incorrect. Thus, extreme caution should be taken when you run `backup` after manipulating the defect list in any way.

### The *Partition Menu*

The *partition menu* is entered by running the `partition` command. It is identified by the *partition>* prompt. The following commands are included in the *partition menu* :

```
a      - change `a' partition
b      - change `b' partition
c      - change `c' partition
d      - change `d' partition
e      - change `e' partition
f      - change `f' partition
g      - change `g' partition
h      - change `h' partition
select - select a predefined table
name   - name the current table
print  - display the current table
label  - write partition map and label to the disk
quit
```

### a - change 'a' partition

This command is used to modify the *a* partition of the *current partition table.* If the *current partition table* is named when `a` is run, a new, unnamed partition table will be created with the changed `a` partition, and the *current partition table* will be set equal to it. If the *current partition table* is unnamed when `a` is run, the *a* partition in the current table is modified. This is explained more under the `name` command. When the `a` command is run, it first displays the current values for the `a` partition of the *current partition table.* It then asks you for new values for the starting cylinder and the number of blocks in the partition. The current values are always the default. Below is an example of the `a` command.

```
partition> a

    partition a - starting cyl      0, # blocks   ...

    Enter new starting cyl [0]: <cr>
    Enter new # blocks [16080, 24/0/0]: 25/
partition>
```

b - change ‘b’ partition

This command is used to modify the b partition of the *current partition table.* The interface to the b command is identical to that of the a command.

c - change ‘c’ partition

This command is used to modify the c partition of the *current partition table.* The interface to the c command is identical to that of the a command.

d - change ‘d’ partition

This command is used to modify the d partition of the *current partition table.* The interface to the d command is identical to that of the a command.

e - change ‘e’ partition

This command is used to modify the e partition of the *current partition table.* The interface to the e command is identical to that of the a command.

f - change ‘f’ partition

This command is used to modify the f partition of the *current partition table.* The interface to the f command is identical to that of the a command.

g - change ‘g’ partition

This command is used to modify the g partition of the *current partition table.* The interface to the g command is identical to that of the a command.

h - change ‘h’ partition

This command is used to modify the h partition of the *current partition table.* The interface to the h command is identical to that of the a command.

select - select a predefined table

This command is used to set the *current partition table* equal to a predefined partition table. First, we need an explanation of what a predefined table really is.

Whenever a *current disk* that is labelled is selected, the *current partition table* will automatically be set to partition information in the label. If this information does not match a predefined table, then a new predefined table is created for it. The name of this new table is always "original xxN", where xx is the disk name and N is the unit number. For instance, if xy0 was selected as the *current disk,* and it’s partition information did not match a predefined table, the *current partition table* would be set to a new table called "original xy0".

Predefined tables can also be specified in the format data file. The default data file contains predefined partition tables for all Sun supported disks. You can add your own partition tables for other disks, or define new tables for Sun supported disks, simply by editing the data file. See “format’s Data File” for more details.

You can also create your own predefined partition tables by using other commands in the *partition menu* . The `a-h` commands allow you to modify the *current partition table*, then the `name` command allows you to freeze a copy of the *current partition table* and give it a name. Remember though, that defining a partition table this way only lasts for one invocation of `format`, whereas adding it to the data file causes it to be defined every time you run `format`.

Thus, when the `select` command is executed, there may be many different predefined partition tables for the *current disk*. They will be displayed in a list, and you can choose whichever one you want the *current partition table* set to. The default choice is always the current value of the *current partition table*. Below is an example of the `select` command.

```
partition> select
     0. Fujitsu-M2333
     1. original xy0
Specify table (enter its number) [1]: 0
partition>
```

`name` - name the current table

This command is used to name the *current partition table*. When a partition table gets named, it freezes the values of the table at their current settings. This way you can create your own predefined partition tables for use on multiple disks. You only need to name the *current partition table* if you have changed some of the partition information in it. Whenever you modify the partition information and the *current partition table* was named, a new, unnamed partition table is created to hold the new information. Once you are done modifying all the partitions you wish to change, you can name the resulting table so it can be used again and again. When you run the `name` command, it simply asks you for the name you wish to assign to the *current partition table*. If you want any spaces to appear in the name, you must remember to enclose the name in double quotes. Otherwise, only the first word you specify is used as the name. Below is an example of the `name` command. It shows how the *current partition table* is named after the `name` command is run.

```
partition> select
          0. Fujitsu-M2333
          1. original xy0
Specify table (enter its number) [1]: 0

partition> name
Enter table name (remember quotes):  "new table"

partition> select
          0. Fujitsu-M2333
          1. original xy0
          2. new table
Specify table (enter its number) [0]: 0
```

### print - display the current table

This command is used to print out the *current partition table.* It prints the name of the table (or "unnamed") and the values of all the partitions in the table. Below is an example of the. print command.

```
partition> print
Current partition table (unnamed):
        partition a - starting cyl     0, # blocks    16750 (25/0/0)
        partition b - starting cyl    24, # blocks    33500 (50/0/0)
        partition c - starting cyl     0, # blocks   550070 (821/0/0)
        partition d - starting cyl    34, # blocks    20345 (30/3/44)
        partition e - starting cyl     0, # blocks        0 (0/0/0)
        partition f - starting cyl     0, # blocks        0 (0/0/0)
        partition g - starting cyl    74, # blocks   500490 (747/0/0)
        partition h - starting cyl     0, # blocks        0 (0/0/0)
partition>
```

### The *Analyze Menu*

The *analyze menu* is entered by running the analyze command. It is identified by the *analyze>* prompt. The following commands are included in the *analyze menu:*

```
read    - read only test           (doesn't harm SunOS)
refresh - read then write          (doesn't harm data)
test    - pattern testing          (doesn't harm data)
write   - write then read           (corrupts data)
compare - write, read, compare     (corrupts data)
print   - display data buffer
setup   - set analysis parameters
config  - show analysis parameters
quit
```

### read - read only test (doesn't harm SunOS)

This command is used to analyze the surface of the *current disk* for media defects. Both the *current disk* and *current disk type* must be set to run this command. Also, if the analysis parameters are set up to do automatic repairing, the *current defect list* must be initialized. The read command is the one type of surface analysis that can be safely run on mounted file systems. It simply reads the sectors specified, looking for media related errors.

When the read command is run, it uses the analysis parameters to control the operation of the command. See the description of the setup command for a complete explanation of the analysis parameters and what they do. If read is being run interactively, it asks you to verify that you want to do the analysis operation. Also, if the boundaries of the analysis overlap with any mounted file systems, you are warned and again given the chance to cancel the command. In the case of the read command, there is no danger from running on a mounted file system, unless automatic repairing is enabled. You should only allow automatic repairing on a mounted file system if the system is idle, so a repair does not interfere with the operation of the system. You can easily monitor the progress of the command since it is continuously displayed. Below is an example of the read command.

```
analyze> read
Ready to analyze (won't harm SunOS). This takes a long time,
but is interruptible with CTRL-C. Continue? y
                pass 0
                pass 1
6/6/50
Total of 0 defective blocks repaired.
analyze>
```

If automatic repairing is enabled, then all media defects found during the analysis will be repaired. Any time an error is detected in a command, the SunOS driver will notify `format` whether the error indicates a media defect or not. If it does indicate a media defect, then the area around the error is analyzed carefully. If the defective sector can be pinpointed, it is automatically repaired.

You do not need to let the `read` command run to completion. Any time you are satisfied that the disk is sufficiently tested, you can abort the command with a CTRL-C.

`refresh` - read then write (doesn't harm data)

This command is very similar to the `read` command, but performs a more comprehensive check of the media. It will not change the data on the *current disk*, but should not be run on a mounted file system. If the warning concerning mounted file systems is displayed, you should never proceed. The `refresh` command reads each sector of the specified area, then writes the same data back to the disk. While it does this, it is looking for media related errors. In all other respects, it is identical to the `read` command.

`test` - pattern testing (doesn't harm data)

This command is very similar to the `read` command, but performs a very comprehensive check of the media without corrupting the data on the *current disk*. For each sector in the analysis region, it reads the data and stores it away. `Test` then writes and reads a specific data pattern to the disk. Finally, the original data is put back on the disk. Although this results in no change of the disk data, it should not be run on a mounted file system. If the warning concerning mounted file systems is displayed, you should never proceed. Below is an example of the `test` command.

```
analyze> test
Ready to analyze (won't harm data). This takes a long time,
but is interruptable with CTRL-C. Continue? y
        pass 0 - pattern = 0xc6dec6de
        pass 1 - pattern = 0x6db6db6d
        pass 2 - pattern = 0x0
        pass 3 - pattern = 0xffffffff
        pass 4 - pattern = 0xaaaaaaaa
73/9/47
Total of 0 defective blocks repaired.
analyze>
```

`write` - write then read (corrupts data)

This command is the quickest way to do pattern testing on the *current disk.* The interface to the `write` command is identical to that for the other analysis commands. However, the `write` command does not preserve the data on the *current disk.* This is the type of analysis that is run automatically by the `format` command after it has formatted the *current disk.* A series of patterns are written to the disk then read back. While this is happening, `format` is looking for media related errors. Because this command does not preserve the data on the *current disk,* it should never be run on a portion of a disk that contains valuable data or a mounted file system.

`compare` - write, read, compare (corrupts data)

This command is the most comprehensive way to check the *current disk* for media defects. It is similar to the `write` command, but it also compares the data pattern read back from the disk to the one originally written. Any discrepancy in the data pattern is shown as an error message. Like the `write` command, `compare` should never be run on a portion of the *current disk* that contains valuable data or a mounted file system.

`print` - display data buffer

This command is used to display the contents of format's data buffer. This is not normally necessary for disk maintenance. This command is only useful if you need to look at the data in a certain sector of the *current disk.* Using the `read` command to fill the buffer with the data desired, the `print` command can then be used to get a hexadecimal dump of the data. If you don't want to view the entire data buffer, you can use a CTRL-C to abort the command at any time.

`setup` - set analysis parameters

This command is used to set the parameters for the actual surface analysis commands. This allows you to specify the parameters once, then run analysis multiple times without reinitializing the parameters. This is the list of information queried for in the `setup` command, along with the initial value of each item:

```
- boundaries of the analysis              (entire disk)
- number of analysis passes               (2)
- enable/disable automatic repair         (enabled)
- stop/continue after an error            (continue)
- random/built-in data patterns           (built-in)
- size of each analysis transfer          (126 sectors)
- enable/disable post-format analysis     (enabled)
- enable/disable extended messages        (disabled)
```

The exact format of the questions asked varies slightly depending on the current state of the parameters. However, every question has the current value as the default, so entering just a carriage return will leave that parameter unchanged.

The boundaries of the analysis are specified in two ways. You can either specify the entire disk, or you can specify the beginning and ending block numbers. If you use block numbers to specify the bounds, the block numbers are inclusive. Specifying the entire disk will not hurt the label and defect information stored on the disk. No matter which surface analysis command you use, that information is carefully saved and restored when the command is completed.

The number of passes to perform can also be specified in two ways. You can either tell `format` to loop continuously, or you can give a specific number of passes to run. In loop mode, any surface analysis command will run until you stop it with a [CTRL-C.]

Automatic repair can be enabled or disabled, depending on how much control you want over the analysis procedure. When it's enabled, any sector that can be identified as causing a media related error will be repaired immediately. This is fine for most cases. However, you can disable this feature if you want discretionary control over what gets repaired. When automatic repairing is disabled, it simply notifies you of all the defective sectors that were found. Also, some controllers may not support repairing. If this is the case, you will simply be notified of all the defective sectors found, independently of whether automatic repair is enabled or disabled.

You can choose to stop when the first error is encountered, or continue in spite of the errors. Normally, analysis is used to find any and all errors, so continuing is the logical choice. However, there might be circumstances which cause you to want to stop as soon as any error is encountered.

The data patterns used for the pattern testing commands come from one of two sources. You can select random data patterns, which cause a different random number to be generated for each pass, or you can choose the built-in data patterns, which are designed to maximize the chance of finding a pattern sensitive failure.

The size of each analysis transfer is fully programmable. Any size can be specified between one sector and 126 sectors. For normal surface analysis, scanning 126 sectors at a time is most efficient. It reduces the total time taken by the analysis commands, but still catches any errors. However, if you are trying to pinpoint an elusive error, you may want to scan with a smaller granularity.

By default, surface analysis is run over the newly formatted portion of the *current disk* whenever the `format` command is executed. However, you are free to disable this feature if you want to `format` without running analysis.

Below are some examples of the `setup` command.

```
analyze> setup
Analyze entire disk [yes]? n
Enter starting block number [0, 0/0/0]: <cr>
Enter ending block number [551409, 822/9/66]: 20/$/$
Loop continuously [no]? y
Repair defective blocks [yes]? n
Stop after first error [no]? <cr>
Use random bit patterns [no]? <cr>
Enter number of blocks per transfer [126, 0/1/59]: <cr>
Verify media after formatting [yes]? <cr>
Enable extended messages [no]? <cr>
analyze>

analyze> setup
Analyze entire disk [no]? y
Loop continuously [yes]? n
Enter number of passes [2]: 10
Repair defective blocks [no]? y
Stop after first error [no]? <cr>
Use random bit patterns [no]? y
Enter number of blocks per transfer [126, 0/1/59]: 26
Verify media after formatting [yes]? n
Enable extended messages [no]? n
analyze>

analyze> setup
Analyze entire disk [yes]? n
Enter starting block number [0, 0/0/0]: 73/
Enter ending block number [48910, 73/0/0]: 73/$/$
Loop continuously [no]? <cr>
Enter number of passes [10]: 2
Repair defective blocks [yes]? <cr>
Stop after first error [no]? y
Use random bit patterns [yes]? <cr>
Enter number of blocks per transfer [26, 0/0/26]: 126
Verify media after formatting [no]? y
Enable extended messages [no]? y
analyze>
```

config - show analysis parameters

This command is used to display the current settings for the surface analysis parameters. It prints out the same lines as the setup command, but fills in the fields for you with the current value of each parameter. Below is an example of the config command.

```
analyze> config
        Analyze entire disk? no
        Enter starting block number: 0, 0/0/0
        Enter ending block number: 14069, 20/9/66
        Loop continuously? yes
        Repair defective blocks? no
        Stop after first error? no
        Use random bit patterns? no
        Enter number of blocks per transfer: 126, 0/1/59
        Verify media after formatting? yes
        Enable extended messages? no
analyze>
```

### The *Defect Menu*

The *defect menu* is entered by running the `defect` command. It is identified by the *defect>* prompt. The following commands are included in the *defect menu:*

```
restore    - set working list = current list
original   - extract manfacturer's list from disk
extract    - extract working list from disk
add        - add defects to working list
delete     - delete a defect from working list
print      - display working list
dump       - dump working list to file
load       - load working list from file
commit     - set current list = working list
quit
```

### `restore` - set working list = current list

This command is used to set the *working defect list* equal to the *current defect list.* This command is useful if you change the *working defect list* then decide that you want to undo the changes. As long as you have not committed the changes, running `restore` will successfully undo them. If the *current defect list* is null when `restore` is run, it will set the working defect to null also. When the `restore` command is run, it asks you to verify that you want to do the restore. It then reinitializes the *working defect list.* Below is an example of the `restore` command.

```
defect> restore
ready to update working list, continue? y
working list updated, total of 25 defects.
defect>
```

### `original` - extract manufacturer's list from disk

This command is used to create a *working defect list* equal to the manufacturer's defect list for the *current disk.* This is usually used when a disk is brand new (never formatted), and you wish to create the original defect list for the disk. All disks shipped from Sun are formatted at the factory, so this command is not necessary. This command is also useful if you repair several defects on the disk then discover it was some problem other than the disk media. By going back to the original defect list and reformatting the drive, you can remove all the erroneous repairs.

The `original` command is optionally supported by a disk controller, so some may not allow you to extract the manufacturer's defect list this way. Some controllers can extract the manufacturer's list if the disk is unformatted, but cannot do it once the disk has been formatted. If you can't use the `original` command, you will have to enter the defects from your hard copy of the defect list using the `add` command. In either case, you must use the `commit` command then reformat the *current disk* for the new defects to be used.

The `original` command is not interruptable by a [CTRL-C]. This is necessary so that the *working defect list* does not get partially updated. Extracting the manufacturer's defect list can take a long time on some disks, so be ready to wait for a while. When `original` is run, it notifies you that the command is not interruptable, and asks for verification that you really want to do the command. It then updates the *working defect list* with the manufacturer's list. Below is an example of the original command.

```
defect> original
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting manufacturer's defect list.
Extraction complete.
Working list updated, total of 25 defects.
defect>
```

`extract` - extract working list from disk

This command is used to extract the current defect list from the *current disk* media. This is not `format's` *current defect list*, it is the list of defects that reflects the actual state of the *current disk*. This list is put into `format's` *working defect list*. The `commit` command can then be used to set `format's` *current defect list*, so the defects can be used by other commands.

This command is only necessary if the copy of the defect list stored on the *current disk* by `format` becomes corrupted. The other reason to use this command is to create a defect list for a disk that is older, and therefore never had a defect list on it. This should be done to all your older disks whenever convenient, so that all your disks have defect lists on them.

The `extract` command is optionally supported by a disk controller, so some may not allow you to extract the disk's current defect list this way. Some controllers can extract the current defect list once the disk has been formatted, but not when the disk is brand new. This only affects you if you did not get your disks from Sun, since all Sun disks are formatted at the factory. If the `extract` command cannot be used, you can use a copy of the defect saved with the `dump` command to initialize the *working defect list* instead. This is why you should always use the `dump` command to save the most up to date defect list you have for each disk. If you don't have a saved copy of the defect list, you will have to enter the defects by hand using the `add` command. In either case, you must use the `commit` command then reformat the *current disk* for the defects to be used.

The `extract` command is not interruptable by a CTRL-C. This is necessary so that the *working defect list* does not get partially updated. Extracting the current defect list can take a long time on some disks, so be ready to wait for a while. When `extract` is run, it notifies you that the command is not interruptable, and asks for verification that you really want to do the extraction. It then updates the *working defect list* with the defects it extracts from the disk media. Below is an example of the `extract` command.

```
defect> extract
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting current defect list.
Extraction complete.
Working list updated, total of 25 defects.
defect>
```

### `add` - add defects to working list

This command is used to add defects to the *working defect list*. This may be necessary for a variety of reasons. If the defect list stored on the *current disk* was corrupted or never existed, and you cannot extract the defect list, you can create the defect list by hand. Also, if you wish to repair a sector but the controller doesn't support repairing, you can add the defective sector to the *working defect list*, commit the defects to the *current defect list*, then reformat the *current disk*. This will have the effect of repairing the defective sector.

When the `add` command is run, it first asks you which mode the defects to be entered are in. The two modes are bytes from index and logical block. If the defects are from the manufacturer's defect list, they will be in bytes from index format. If you are adding a defective sector to the list, it will be in logical block format. If you need to add defects of each type to the working defect list, you will have to run the command twice.

If you specify that the defects are in bytes from index format, you are then prompted for the defect's cylinder, head, bytes from index and length. The default value for the length is always -1, which means that the length isn't known. Some manufacturer's defect lists include the length of the defect, so you can specify the correct value if you have it. If you specify that the defects are in logical block format, you are instead prompted for the logical block number of the defective sector. This is the same value that you would specify to `repair` to fix the defective sector.

After the information about the defect has been entered, a summary of the defect is printed, along with which defect number it will become when it is added to the *working defect list*. You are then asked to confirm that the defect should be added. If an affirmative is given, the defect is added to the *working defect list* and the next defect is prompted for. When you are finished adding defects to the *working defect list*, simply use CTRL-C to exit the loop. Below are some examples of the `add` command.

```
defect> add
        0. bytes-from-index
        1. logical block
Select input format (enter its number) [0]: 0

Enter Control-C to terminate.
Enter defect's cylinder number: 127
Enter defect's head number: 0
Enter defect's bytes-from-index: 2345
Enter defect's length (in bits) [-1]: 23
 num     cyl     hd     bfi     len     sec
   1     127      0    2345      23
defect number 1 added.

Enter defect's cylinder number: 400
Enter defect's head number: 5
Enter defect's bytes-from-index: 4567
Enter defect's length (in bits) [-1]:
 num     cyl     hd     bfi     len     sec
   2     400      5    4567
defect number 2 added.

Enter defect's cylinder number: ^C
defect>


defect> add
        0. bytes-from-index
        1. logical block
Select input format (enter its number) [0]: 1

Enter Control-C to terminate.
Enter defective block number: 34/5/20
 num     cyl     hd     bfi     len     sec
   3      34      5                      20
defect number 3 added.

Enter defective block number: 500/0/0
 num     cyl     hd     bfi     len     sec
   4     500      0                       0
defect number 4 added.

Enter defective block number: ^C
defect>
```

delete - delete a defect from working list

This command is used to delete defects from the *working defect list*. This may be necessary if you have added defects or repaired sectors then discovered that the media was not the problem. By deleting the defects from the *working defect list*, then committing to the *current defect list* , then reformatting the *current disk*, you can effectively erase the existence of the defects.

When the `delete` command is run, it asks you for the number of the defect that should be deleted. The defects are numbered in the `print` command, so you can get the number you need by running `print` and finding the defect you want to remove. When you enter the number, `format` then prints a summary of the defect and asks you to verify that you want to delete it. If an affirmative is given, the defect is deleted from the *working defect list.* A word of caution is warranted here. If you are planning on deleting more than one defect, you should do a print command after each `delete` and find the next defect you wish to remove. Once you delete a defect, all remaining defects in the list are renumbered, so other defects you may want to delete can change number on you. Thus, you should always look up each defect right before deleting it, and examine the summary printed carefully before confirming the delete. Below are some examples of the `delete` command.

```
defect> print
 num      cyl      hd      bfi      len      sec
   1      127       0     2345       23
   2      400       5     4567
   3       34       5                         20
   4      500       0                          0
total of 4 defects.

defect> delete
Specify defect to be deleted (enter its number): 2
 num      cyl      hd      bfi      len      sec
   2      400       5     4567
defect number 2 deleted.

defect> delete
Specify defect to be deleted (enter its number): 1
 num      cyl      hd      bfi      len      sec
   1      127       0     2345       23
defect number 1 deleted.

defect> print
 num      cyl      hd      bfi      len      sec
   1       34       5                         20
   2      500       0                          0
total of 2 defects.
defect>
```

`print` - display working list

This command is used to print out the *working defect list.* Remember that this is not necessarily the same as the *current defect list,* which is the list used when you format the *current disk.* See the descriptions of the `restore` and `commit` commands for more details. The *working defect list* is displayed as a numbered list of defects. The number next to each defect can be used to identify the defect for other commands. The information shown for each defect is cylinder, head, bytes from index, length, and logical sector. Some of the information may not be known, so some of the fields may be blank for certain defects. Typically, if the defect was from the manufacturer's defect list, the sector number will not be known. If the defect was added as the result of surface analysis, the bytes from index will generally not be known. This is ok, since only one of these two pieces of information is necessary to locate the defect on the *current disk.*

The defects are shown in the order they are stored in the *working defect list.* This ordering allows `format` to consistently locate the defects on the *current disk.* All defects that are specified in bytes from index format are sorted so they appear in increasing location on the disk. All the defects specified in logical sector format are added to the end of the defect list in the order they are specified.

When the `print` command is run, it will display the *working defect list* one page at a time. You can use a carriage return <cr> to scroll to the next page. Also, you can abort the command at any time with a CTRL-C. Below is an example of the `print` command.

```
defect> print
 num      cyl     hd     bfi      len      sec
   1       86      7   32641        2
   2      171      9    2352        2
   3      263      9   22757        3
   4      264      8    5853        6
   5      329      9   29195        3
   6      334      7     962        2
   7      350      1   39377        2
   8      352      8   27660        3
   9      377      9   21473        2
  10      383      2   13527        3
  11      403      1   31726        2
  12      486      1    7965        2
  13      513      7     260        2
  14      530      2   16558        3
  15      540      1    4325        3
  16      544      5   12517        3
  17      565      3   38136        6
  18      606      7   21997        3
  19      650      2   11262        3
  20      689      3   31911        2
  21      703      1   20294        3
  22      731      4   38475        3
- hit return for more -
  23      738      0   36468        3
  24       73      0                         23
  25       73      1                         20
total of 25 defects.
defect>
```

### dump - dump working list to file

This command is used to dump the *working defect list* to a SunOS file for later retrieval with the load command. This command allows you to save an online copy of the *current disk*'s defect list. By doing this, you insure that the defect list will not be lost, even if the data on the disk is corrupted.

If you are running format under MUNIX, you must take the extra step of saving the defect list to non-volatile storage, such as tape. the MUNIX file system provides the tar command for this purpose. Saving the defect list to tape is not absolutely required—you can also bring up format after you have booted SunOS. The extra step of saving the list to tape is recommended for peace of mind, however. Thus, if you format a new disk under MUNIX, you should run format again after you have booted SunOS and run the dump command to save the defect list. When you run the dump command, it asks you for the name of the file it should dump to. When you specify the defect file, it is a good idea to include the serial number of the disk in the name, so you can later match the file with the correct disk. The file is in ascii, so you can look at it. However, it contains a checksum, so you cannot edit the file by hand and have it still work in the format program. Also, since dump writes the *working defect list*, you should make sure that the *working defect list* and *current defect list* are the same before running the command. Below is an example of the dump command.

```
defect> dump
Enter name of defect file: /usr/defects/2333_deflist.14257
defect file updated, total of 25 defects.
defect>
```

Note that a separate directory is used for defects for easy access and maintainability. In addition, the defect list name contains the drive's mode and serial number (233 and 14257, respectively). This is strongly advised.

### load - load working list from file

This command is used to load a defect file created by dump into the *working defect list*. The file must have been created by the dump command, or load will not accept it. When the load command is run, it asks you for the name of the defect file. It then initializes the *working defect list* from the file, and reports how many defects were in the list. Remember that you still have to run the commit command to set the *current defect list*.

When running under MUNIX, you will have to get the defect list from tape you saved using tar. You can then use the load command as shown. Below is an example of the load command.

```
defect> load
Enter name of defect file: /usr/defects
ready to update working list, continue? y
working list updated, total of 25 defects.
defect>
```

### commit - set current list = working list

This command is used to set the *current defect list* equal to the *working defect list*. When a *current disk* is selected, an attempt is made to read the defect list off the disk. If a list is found, both the *current defect list* and *working defect list* are set equal to it. If no list is found on the disk, then both are set to null. Several commands require that the *current defect list* be initialized before they will execute. If the *current disk* didn't have a list on it, then the commit command is the only way to initialize the *current defect list*. When you run the commit command, the *current defect list* always becomes initialized. If the *working defect list* was null, both lists are set to zero length instead. This is different than a null list, it means that the disk is assumed defect free. This may be useful if you have an older disk with no defect list and don't want to bother creating one.

If the *working defect list* was modified using any of the other commands in the defect menu, commit must be run so the *current defect list* reflects these changes. Also, the disk must be reformatted, so that the media itself uses the new defect list. If you want to, you can only reformat the portion of the *current disk* that the defect list changes affect. However, you should be careful to include all necessary portions of the *current disk*, or some of your changes won't take effect.

If the repair command is used to add a sector to the defect list, the *current defect list* is updated automatically and no further action is necessary. Thus, the commands in this menu should only be necessary if the controller doesn't support repairing or the defect list on the *current disk* was corrupted somehow.

When the commit command is run, it asks you to verify that you want to do the commit. It then tells you how many defects the *current defect list* now contains. If you have modified the *working defect list* and wish to use those modifications, you must run the commit command before reformatting the *current disk*. If you have modified the *working defect list* and you exit the *defect menu* without running the commit command, a warning message is printed telling you that there are uncommitted changes. You can then go back into the *defect menu* and run commit if desired. Below is an example of the commit command.

```
defect> commit
ready to update Current Defect List, continue? y
Current Defect List updated, total of 25 defects.
Disk must be reformatted for changes to take effect.
defect>
```

## 10.6. Using format for Basic Maintenance

This chapter provides step-by-step instructions for using format. All the common uses for format are described in detail.

### Formatting a New Disk from Sun

Whenever you are adding a disk to your system that you purchased from Sun (including any disks that came with the system), you can follow these easy steps to ready the disk for use by SunOS. All disks shipped by Sun are formatted and labelled at the factory. This makes installation very simple. When you first attach a new disk to your system, Sun recommends that you reformat the entire drive. This insures that any head movement that occurred during shipment will not affect the performance of your new disk. Also, if you wish to partition your new disk differently than the Sun default partitions, you will have to create a partition table and relabel the disk. You can create the partition table from within format, or you can add it to the format.dat file. If you are running under

MUNIX, you must create the partition from within format. The following is an example of how you would install a new disk. In this example, you are adding xd0 to the system. You are also running under SunOS, not MUNIX, so xd0 is not your system disk.

First, enter format and select the disk you wish to work on:

```
csh% format
Searching for disks...No permission (or no disks found)!

csh% su
csh# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
        0. xd0 at xdc0 slave 0
           xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
        1. sd0 at si0 slave 0
           sd0: <drive type unknown>
Specify disk (enter its number): 1



AVAILABLE DRIVE TYPES:
        0. Micropolis 1355
        1. Toshiba MK 156F
        2. Micropolis 1558
        3. other
Specify disk type (enter its number): 2
selecting sd0: <Micropolis 1558>
[disk unformatted, defect list found]


FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        show       - translate a disk address
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        quit
format>
```

At this point, note that the disk is already formatted and a defect list is present on it. This should ALWAYS be the case with disks shipped from Sun. If you do not see this message, something is wrong and you should immediately exit format and use diagnostics to locate the problem. Now that the *current disk* is selected, you need to save a copy of the disk's defect list. If you are running under MUNIX, see *Installing the SunOS* for instructions for doing this. If you are bringing up the system disk, and working under MUNIX, you should skip this part for now. Once you have SunOS installed on the disk and running, you should then rerun format and execute just this part of the installation. This way you insure that you will always have a copy of the defect list handy. Note that before saving the defect list, you should compare it to the hard copy of the manufacturer's defect list. It should contain all the defects on the hard copy list. It may also contain some defects that were found at the Sun factory. If any of the defects are missing, you should add them to the defect list before proceeding.

```
format> defect
     restore   - set working list = current list
     original  - extract manufacturer's list from disk
     extract   - extract working list from disk
     add       - add defects to working list
     delete    - delete a defect from working list
     print     - display working list
     dump      - dump working list to file
     load      - load working list from file
     commit    - set current list = working list
     quit
defect> print
     num      cyl      hd      bfi      len      sec
     1        11       6    31858       4
     2        21       4    14820       4
     3       106       7    27403       4

              .
              .
              .

     59      800       0     6156        4
     60      811       0    27738        6
     61      820       4     4502        3
total of 61 defects.
defect> dump
Enter name of defect file: 2333_defs.033537
defect file updated, total of 61 defects.
defect> q
format>
```

It is always a good idea to put the serial number and model of the disk in the name of the defect file. This lets you match up the defect file with the correct disk easily. If the serial number of the disk is not readily available, use some other unique identifier.

The next step is to reformat the *current disk*. The default values for the bounds of the format command will cause the entire drive to be reformatted. Also, by leaving the surface analysis parameters in their default state, 2 passes of analysis will be run over the disk when format completes. It is recommended that you do this analysis to verify the integrity of the media.

```
format> format
Ready to format. Formatting cannot be interrupted
and takes 11 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
        pass 0 - pattern = 0xc6dec6de
   1217/11/14

        pass 1 - pattern = 0x6db6db6d
   1217/11/14
Total of 0 defective blocks repaired.
```

If any defects are found during surface analysis, they will be automatically repaired if possible. If the automatic repair did not succeed, you need to repair them manually. See the section on "Repairing a Defective Sector" for step–by–step instructions.

If you are happy with the default partitioning of the disk, you are now done with `format`. Your new disk is fully functional, and ready to be used by SunOS. If you wish to change the partitioning of the disk, use the commands in the *partition menu* to create a table, and the `label` command to label the *current disk.*

### Formatting a New Disk (not from Sun)

When you are adding a disk to your system that has never been formatted before, follow these easy steps to ready the disk for use by SunOS. All disks shipped by Sun are formatted at the factory, so this section applies only to disks that were purchased elsewhere. The following is an example of how you would install a new disk. In this example, you are adding `xd0` (a Fujitsu-M2333) to the system. You are running under SunOS, not `MUNIX`, so `xd0` is not your system disk.

First, enter `format` and select the disk you wish to work on:

```
csh# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
        0. xd0 at xdc0 slave 0
           xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
Specify disk (enter its number): 1



AVAILABLE DRIVE TYPES:
        0. Fujitsu-M2351 Eagle
        1. Fujitsu-M2333
        2. Fujitsu-M2361 Eagle
        3. CDC EMD 9720
        4. other
Specify disk type (enter its number): 1
selecting sd0: <Fujitsu-M2333>
[disk unformatted]
```

If the disk type is not one known to format , you can add support for the type in two ways. The easiest way is to add a definition to the format.dat file. This way the disk type will be known for all subsequent executions of format. Unfortunately, you can't do this if you are running under MUNIX. Thus, the second approach is designed to allow you to install your disk. By selecting other in the above type command, you can enter all the geometry information for the disk type by hand. The disk type will then be known for this instance of format . If you have to run format again on this disk, you may have to enter more information. Thus, it is always best to add the disk types to the format.dat file.

Now that the *current disk* and *current disk type* are set, the *manufacturer's defect list* can be created. This is done in one of several ways. If the disk controller supports the original command, that should be used to create the list. If that command is not supported, the add command must be used to enter the defect list by hand. A hard copy of the defect list should be attached to the disk. This can be used to enter the list by hand. There is one other exception to the above rules. The Emulex MD21 controller supports the original command, but only after the *current disk* has been formatted. Thus, if you are adding a disk to this controller, you should skip this step for now and come back to it after you have formatted the disk. In this example, you are using the Xylogics 7053 controller, which supports the original command for brand new disks:

```
format> defect
    restore   - set working list = current list
    original  - extract manufacturer's list from disk
    extract   - extract working list from disk
    add       - add defects to working list
    delete    - delete a defect from working list
    print     - display working list
    dump      - dump working list to file
    load      - load working list from file
    commit    - set current list = working list
    quit
defect> original
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting manufacturer's defect list.
Extraction complete.
Working list updated, total of 61 defects.
defect>
```

After initializing the *working defect list* with either `original` or `add`, you should compare the resulting list with the hard copy supplied with the disk. If it doesn't match, you can use the `add` and `delete` commands to make them match. Once the *working defect list* is correct, you must run the `commit` command to tell `format` you have a complete defect list. The *working defect list* is then copied to the *current defect list*, which is used when the *current disk* is formatted.

```
defect> print
     num      cyl     hd     bfi      len      sec
       1       11      6   31858        4
       2       21      4   14820        4
       3      106      7   27403        4

               .
               .
               .

      59      800      0    6156        4
      60      811      0   27738        6
      61      820      4    4502        3
    total of 61 defects.
    defect> commit
    ready to update Current Defect List, continue? y
    Current Defect List updated, total of 61 defects.
    Disk must be reformatted for changes to take effect.
    defect>
```

Before exiting the ,*defect*menu you should save the defect list into If you are running under `MUNIX`, see *Installing the SunOS* for instructions for doing this. When the system is installed and SunOS is running, rerun just this part of format to save the defect list. It is always a good idea to put the serial number and model of the disk in the name of the defect file. This lets you match up the defect file with the correct disk easily. If the serial number of the disk is not readily available, use some other unique identifier. Once you have saved the defect list, you can quit the *defect menu.*

```
defect> dump
Enter name of defect file: 2333_defs.033537
defect file updated, total of 61 defects.
defect> q
format>
```

The next step is to format the *current disk.* The default values for the bounds of the format command will cause the entire drive to be formatted. Also, by leaving the surface analysis parameters in their default state, two passes of analysis will be run over the disk when the format completes. It is recommended that you do this analysis to verify the integrity of the media.

```
format> format
Enter starting block number [0, 0/0/0]: <cr>
Enter ending block number [551409, 822/9/66]: <cr>

Ready to format. Formatting cannot be interrupted
and takes 11 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
        pass 0 - pattern = 0xc6dec6de
   1217/11/14

        pass 1 - pattern = 0x6db6db6d
   1217/11/14
Total of 0 defective blocks repaired.
```

If any defects are found during the surface analysis, they will be automatically repaired if possible. If the automatic repair did not succeed, you need to repair them manually. See the section on "Repairing a Defective Sector" for step–by–step instructions.

If the disk has a partition table defined in the format.dat file, and you are happy with using this default table, you are ready to label the disk. Note that if there are multiple partition tables in the format.dat file for a given disk type, the first one in the file is always used as the default. If there are no predefined tables, or you want to alter the table for this specific disk, you need to use the *partition menu* commands to create your own partition table. Once you are ready to label the disk, you simply run the label command. In this example, you are using the default partition table, so you do not have to specify one.

```
format> label
Current Partition Table is not set, using default.
Ready to label disk, continue? y
format> q
csh#
```

Once the disk is labeled, you are done with `format`. Your new disk is fully functional, and ready to be used by SunOS.

## Repairing a Defective Sector

If a disk on your system has a defective sector, you can repair the sector using the step by step instructions given in this section. You may become aware of defective sectors through several different means. If you run surface analysis on a disk, and repairing is not supported by the controller or is disabled, any defects found will simply be reported to you. In these cases, the exact sector in error will be known, since surface analysis very carefully pinpoints the source of the error before notifying you. In the course of running SunOS, you may get a number of error messages from the disk driver concerning a particular portion of the disk. If these errors seem media related, you may wish to look for a defective sector in that area. Note that you should not take the sector number reported by SunOS as gospel. Since SunOS does disk operations many sectors at a time, it is often hard to pinpoint exactly which sector caused a given error. You should always use the analysis tools provided in `format` to find the defective sector. For this example, assume that you got several ECC errors for sector number 12345 on `xd0`. First, enter `format` and select the disk.

```
csh# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
        0. xd0 at xdc0 slave 0
           xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
Specify disk (enter its number): 0
selecting xd0: <CDC EMD 9720 cyl 1147 alt 2 hd 10 sec 48>
[disk formatted, defect list found]
Warning: Current Disk has mounted partitions.


FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        show       - translate a disk address
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        quit
format>
```

Next, you need to pinpoint exactly where the defective sector is. Because the xd0 disk is being used by SunOS, run the read test. This will not interfere with the operation of the system. Run the read test over a couple tracks worth of sectors, using the sector number given by the SunOS error message as the midpoint of your search. Also set up the analysis parameters to loop continuously, since it is important to find the problem sector. This will cause format to run analysis until you stop it with a [CTRL-C]. Also, run the read test with a transfer size of 1 sector, so it will be very clear which sector caused any errors encountered. If the analysis fails to show any defects, you may want to repair the sector given in the SunOS error message anyway, in hopes that it is the defective sector. Or, you may want to wait a while and see if the error was transient. However, for this example, assume that the error showed itself during analysis.

```
format> analyze


ANALYZE MENU:
        read     - read only test   (doesn't harm SunOS)
        refresh  - read then write  (doesn't harm data)
        test     - pattern testing  (doesn't harm data)
        write    - write then read      (corrupts data)
        compare  - write, read, compare (corrupts data)
        print    - display data buffer
        setup    - set analysis parameters
        config   - show analysis parameters
        quit
analyze> setup
Analyze entire disk [no]? n
Enter starting block number [12330, 25/6/42]: 12330
Enter ending block number [584159, 1216/9/47]: 12360
Loop continuously [no]? y
Repair defective blocks [yes]? n
Stop after first error [no]? <cr>
Use random bit patterns [no]? <cr>
Enter number of blocks per transfer [31, 0/0/31]: 1
Verify media after formatting [yes]? <cr>
Enable extended messages [no]? <cr>

analyze> read
Ready to analyze (won't harm SunOS). This takes a long time,
but is interruptable with CTRL-C. Continue? y


        pass 0
   25/7/24

        pass 1
Block 10974  (18/4/18), Corrected media error (hard data ecc)
   25/7/24

        pass 2
Block 10974  (18/4/18), Corrected media error (hard data ecc)
   25/7/24

        pass 3
   25/7/24

        pass 4
Block 10974  (18/4/18), Corrected media error (hard data ecc)
^C   25/7/24

Total of 0 defective blocks repaired.
analyze> quit
```

Because some controllers do not support repairing, the read test above was run with automatic repair turned off. If we wanted to, we could have had `format` repair the defective sector as soon as it was found. However, we will instead show the two methods for repairing a defective sector manually. If the controller does support repairing, it is a simple case of running the `repair` command on the sector that surface analysis showed to be bad. This will preserve the data in the bad sector if possible, so there is no need to back up your disk.

```
format> repair
Enter defective block number: 18/4/18
Ready to repair defect, continue? y
Repairing 18/4/18.
Repair succeeded.
format> quit
csh#
```

However, if repairing is not supported, it is a little more involved. First, the defective sector must be added to the *working defect list.* Also, the *working defect list* must then be committed to the *current defect list* so the new defect will show up on the disk's copy of the defect list.

```
format> defect
    restore  - set working list = current list
    original - extract manufacturer's list from disk
    extract  - extract working list from disk
    add      - add defects to working list
    delete   - delete a defect from working list
    print    - display working list
    dump     - dump working list to file
    load     - load working list from file
    commit   - set current list = working list
    quit
defect> add
    0. bytes-from-index
    1. logical block
Select input format (enter its number) [0]: 1
Enter defective block number: 18/4/18
 num     cyl     hd     bfi     len     sec
  62      18      4                      18
ready to add defect, continue? y
defect number 62 added.
Enter defective block number: ^C
defect> commit
ready to update Current Defect List, continue? y
Current Defect List updated, total of 62 defects.
Disk must be reformatted for changes to take effect.
defect> quit
format>
```

Finally, the *current disk* must be reformatted, so the media itself reflects the new defect. If possible, you can reformat only the portion of the *current disk* that the defect lies in. However, most controllers that do not support repairing also require that you `format` the entire disk at once, so you will probably need to format the whole disk. Remember to back up your disk before doing any format operation. In this example, only the portion of the *current disk* necessary is reformatted.

```
format> format
Enter starting block number [0, 0/0/0]: 18/4/0
Enter ending block number [551409, 822/9/66]: 18/4/$

Ready to format. Formatting cannot be interrupted
and takes 1 minutes (estimated). Continue? y
Beginning format. The current time is Wed Feb 10 18:31:57 1988

Verifying media...
        pass 0 - pattern = 0xc6dec6de
   18/4/47

        pass 1 - pattern = 0x6db6db6d
   18/4/47
Total of 0 defective blocks repaired.
format> quit
```

NOTE *Because you previously set the surface analysis parameters to loop continuously, you had to use* CTRL-C *to get out of the analysis that runs after you format. The sector is now repaired, and the disk is fully functional once again.*

### Relabeling a Corrupted Disk

If a disk on your system is corrupted, and the primary label is destroyed, you may be able to restore the label from backup copies that are stored on the disk. If this doesn't work, you will have to recreate the partition table yourself and relabel the disk. If the disk used a predefined partition table, this is not difficult. However, if the partition table was not defined in `format.dat`, you will have to remember the partition information that was on the disk. This is why you should always add the partition tables for all your disks to the `format.dat` file. The following is an example of how you would attempt to restore the label. These commands can all be run under `MUNIX`, so this procedure will even work if your system disk has been corrupted. In this example, you are trying to relabel `xd0`.

First, enter `format` and select the disk you need to work on:

```
csh# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
        0. xd0 at xdc0 slave 0
           xd0: <drive type unknown>
Specify disk (enter its number): 0


AVAILABLE DRIVE TYPES:
        0. Fujitsu-M2351 Eagle
        1. Fujitsu-M2333
        2. Fujitsu-M2361 Eagle
        3. CDC EMD 9720
        4. other
Specify disk type (enter its number): 1
selecting xd0: <Fujitsu-M2333>
[disk formatted, defect list found]


FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        show       - translate a disk address
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        quit
format>
```

Once the disk type is specified, format knows the geometry and is now able to locate the defect list. If the defect list is not found at this point (and you know the disk had a defect list), it was probably also corrupted. You can recreate the defect list after you are done fixing the label. See the section on "Creating a Defect List". At this point, you can search for the backup labels. Simply execute the backup command. If a backup label is found, format will use it to rewrite the primary label. The disk is then functional again.

```
    format> backup
    Searching for backup labels...found.
    Restoring primary label and defect list.
    format>
```

If no backup labels were found, you will have to reconstruct the label by hand. If the partition table that was on the disk is in the `format.dat` file, this is fairly easy. If you did not use a predefined partition table, it is more complicated. This is why you should add the partition tables for all the disks on your system to the `format.dat` file.

For disks that use a predefined partition table, creating the label by hand is simple. The geometry information is already known, since the `type` command was run, so it is only the partition information that must still be entered. This is done via the `load` command. First, the *partition menu* must be entered. Next, the `load` command is run. The desired partition table can then be chosen from the list of possibilities.

```
format> partition

PARTITION MENU:
        a      - change 'a' partition
        b      - change 'b' partition
        c      - change 'c' partition
        d      - change 'd' partition
        e      - change 'e' partition
        f      - change 'f' partition
        g      - change 'g' partition
        h      - change 'h' partition
        select - select a predefined table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        quit
partition> select
        0. Fujitsu-M2333
        1. my table
Specify table (enter its number) [0]: 1

partition> label
Ready to label disk, continue? y
```

Once you have selected the correct partition table, all that is left is to label the current disk. You can use the `label` command to label the disk.

### Creating a Defect List

There are certain situations under which you will need create a defect list for a disk on your system. All Sun disks shipped before release 3.2 did not have defect lists on them. If you wish to operate on a disk of this type, you will need to create a defect list first. Also, the defect list on a disk may be corrupted by a media problem, or some other catastrophe. The instructions in this section explain how to recreate the defect list for a disk on your system.

No matter what the reason for needing the defect list, there are only a couple of approaches to solving the problem. First, you need to enter `format` and select the disk that needs work.

```
csh# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
        0. xd0 at xdc0 slave 0
           xd0: <Fujitsu-M2333 cyl 821 alt 2 hd 10 sec 67>
Specify disk (enter its number): 0
selecting xd0: <Fujitsu-M2333>
[disk formatted, no defect list found]


FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        show       - translate a disk address
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        quit
format>
```

Note that you can tell the defect list was not present on the disk by the message printed in the square brackets. Here is where the two approaches differ. If your disk once had a defect list on it, and you had saved a copy of that list in a SunOS file with the dump command, recreating the defect list is very easy. You simply enter the *defect menu*, and use the load command to initialize the *working defect list*. If you can initialize the *working defect list* this way, you can skip the next section and go on to where you commit the defect list.

```
format> defect
    restore   - set working list = current list
    originial- extract manufacturer's list from disk
    extract   - extract working list from disk
    add       - add defects to working list
    delete    - delete a defect from working list
    print     - display working list
    dump      - dump working list to file
    load      - load working list from file
    commit    - set current list = working list
    quit
defect> load
Enter name of defect file: 2333_defs.033537
ready to update working list, continue? y
working list updated, total of 60 defects.
defect>
```

If you do not have a copy of the defect list in a SunOS file, you can try to extract the defect list from the disk media. The `extract` command actually looks at the data on the *current disk* and calculates which sectors have been repaired. This lets it create a defect list that represents the defects on the disk. Simply enter the *defect menu* and run the `extract` command. This command can take quite a while, so be prepared to wait. If you initialize the *working defect list* this way, you should always immediately run the `dump` command to save a copy of the defect list in a SunOS file. This will prevent you from ever having to extract again later.

```
format> defect
    restore   - set working list = current list
    original  - extract manufacturer's list from disk
    extract   - extract working list from disk
    add       - add defects to working list
    delete    - delete a defect from working list
    print     - display working list
    dump      - dump working list to file
    load      - load working list from file
    commit    - set current list = working list
    quit
defect> extract
Ready to update working list. This cannot be interrupted
and may take a long while. Continue? y
Extracting current defect list.
Extraction complete.
Working list updated, total of 60 defects.
defect> dump
Enter name of defect file: 2333_defs.033537
defect file updated, total of 60 defects.
defect>
```

Some controllers do not support extracting the defect list this way. If this is the case, the only way to create a defect list for the disk is to enter the defects manually with the `add` command.

Now, assume that the *working defect list* has been initialized using one of the above methods. Next, you must commit the *working defect list* to the *current defect list*. This causes it to be used when the disk is operated on.

```
defect> commit
ready to update Current Defect List, continue? y
Current Defect List updated, total of 60 defects.
Disk must be reformatted for changes to take effect.
defect> quit
format>
```

After committing the defect list, all that is necessary is to get a copy of the defect list onto the *current disk*. If you were creating the defect list because you intend to repair or reformat the *current disk*, then those commands will write the defect list onto the *current disk*. However, if you were creating the defect list just to have one (which is a good idea), you must manually write it to the *current disk.* The easiest way to do this is with the backup command. By running the backup command and forcing it to relabel the disk, you also cause it to put the defect list back on the disk. This way, the *current disk* will have a defect list that reflects the state of the disk media. You should be careful when you run the backup command, since format cannot enforce that the defect list you are writing to the disk really reflects the state of the disk media. Be sure you initialized the *current defect list* correctly before proceeding. When you run the backup command, it will notify you that the *current disk* was already labeled. That's ok, force it to relabel the disk anyway. This is the only way to cause the defect list to be written out.

```
format> backup
Disk had a primary label, still continue? y
Searching for backup labels...found.
Restoring primary label and defect list.
format> quit
csh%
```

The *current disk* now has the correct defect list on it, and can be used by format without incident.

## 10.7. format Error Messages

This section lists the error messages that you may encounter while running format. Each message is described in detail, and suggestions for how to get around the error are given.

### Backup Label Claims Different Type.

This error occurs when you are executing the backup command. It means that a backup label was found, but the disk type claimed by the backup label is not the disk type currently specified for the disk. This generally means that you selected the wrong disk type with the type command. Try running type again, this time specifying a different disk type. If you are sure that you are specifying the correct disk type, then the backup label must be incorrect. You should ignore it and build your own label, using label to write it on the disk.

### Bad Block Map Table Overflow.

This error occurs when you are repairing a defective sector on an SMD disk. This can happen from repair, format or any of the surface analysis commands. It means that the bad block table is full so another mapping cannot be added. This only happens when the *current disk* already has 126 mapped sectors. This will generally happen only if the *current disk* is not configured for slip sectoring. You should configure all of your SMD disks for slip sectoring, since it has many benefits. Another common cause of this problem is running surface analysis with a bad cable. This will cause defects to be found that aren't really there, and result in many unnecessary mappings. You can remedy this situation by using the delete command to remove the unreal defects from the *current defect list* then reformatting the disk with the format command. If a disk with slip sectoring has 126 genuinely mapped sectors, it is not within the Sun specification, and should be returned.

Bad keyword '%s' -- line %d of data file.

This error occurs during the startup processing of `format,` when it is reading in the data file. It means that `format` was expecting a keyword next in the file, but found something else. The line number of the error is printed so the problem can be easily located. The syntax of the data file consists of keywords followed by operators and values. The keywords recognized at any given time depend on what declaration is currently active. For a complete explanation of the data file syntax and a list of the keywords, see the "Data File Format" section.

Can't open selected disk '%s'.

This error can occur whenever the *current disk* or *current disk* type is changed. It means that `format` was unable to open the file representing the raw C partition of the *current disk*. This error shouldn't occur, since the disk must have been successfully opened during the startup processing or it would not have appeared as a choice. Exit the program and check the file carefully to see if something has happened to it.

Can't repair sector that is already mapped.

This error occurs when you are repairing a defective sector on an SMD disk. This can happen from `repair,` `format,` or any of the surface analysis commands. It means that the sector being repaired is already mapped to an alternate sector. A common cause of this problem is running surface analysis with a bad cable. This will cause defects to be found that aren't really there, and results in many unnecessary mappings. You can remedy this situation by using the `delete` command to remove the unreal defects from the *current defect list* then reformatting the disk with the `format` command. If the sector being repaired is really defective, there is not much you can do. If the *current disk* is configured for slip sectoring, you can try running surface analysis on the alternate cylinders. If the defect in the alternate sector shows up during this analysis, it will be repaired by slipping the sector. This will cause the problem to go away. If this doesn't work, the only thing you can try is reformatting the alternate cylinders in hopes that the defect won't show up again.

Controller does not support extracting current defect list.

The `extract` command is optionally supported by various controller types. If the controller for the *current disk* doesn't support the command, an attempt to run `extract` will generate this message. If the *current defect list* cannot be initialized with `extract,` you can enter the defects manually with the `add` command.

Controller does not support extracting manufacturer's defect list.

The `original` command is optionally supported by various controller types. If the controller for the *current disk* doesn't support the command, an attempt to run `original` will generate this message. If you want to initialize the *current defect list,* you can enter the defects manually with the `add` command.

Controller does not support repairing.

The ability to repair defective sectors is optionally supported by various controller types. If the controller for the *current disk* doesn't support repairing, any attempt to repair a sector will generate this message. This can happen from `format,` `repair,` or any of the surface analysis commands. You can repair the defective sector by using `add` to add it to the *current defect list* then reformatting that portion of the *current disk.*

Controller firmware is outdated and doesn't support extraction.

This error occurs when you are running the `extract` command to extract the manufacturers defect list and the controller firmware has not been upgraded to handle this. The only controller that causes this message to be generated is the Xylogics 450, which formerly did not support extraction. To solve the immediate problem, you can create the defect list manually with the `add` command. However, you should get a firmware update from the manufacturer anyway, since running with outdated firmware can have unfortunate consequences.

Controller requires formatting of entire tracks.

This message can occur when executing the `format` command. It means that the controller does not support formatting of a portion of a track. You will have to respecify the bounds of the format operation in terms of whole tracks.

Controller requires formatting the entire drive.

This message can occur when executing the `format` command. It means that the controller does not support the option of formatting a portion of the disk. You must run `format` over the entire disk. Remember to back up any files on the disk first.

Current Defect List must be initialized to do automatic repair.

This error can occur when executing any of the surface analysis commands. All of these commands require a *current defect list* to be initialized if automatic repairing is enabled. The *current defect list* can be initialized with the `commit` command.

Current Defect List must be initialized.

This warning can occur when you execute the `format` or `repair` commands. It means that you need to initialize a *current defect list* in order to proceed, since both of the above commands require it. You can initialize the *current defect list* with the `commit` command.

Current Disk Type is not set.

This error can occur when executing many of the commands in the *command menu* without first having set the *current disk* type. This parameter must be set for many of the commands to run. If this error occurs after you have already selected a *current disk* with the `disk` command, you need to specify the disk type with the `type` command. Otherwise, the type is automatically set when you select a *current disk.*

Warning: Current Disk has mounted partitions.

This warning occurs when you run the `disk` command to select a disk that has mounted partitions. You should exercise caution when working on a disk with mounted partitions. If possible, you should unmount the partitions before preceding.

Current Disk is not set.

This error can occur on any of the commands which require the *current disk* parameter to be set (e.g. the `format` command). You can select the *current disk* by running the `disk` command. After you do so, your selection remains the *current disk* until you change it with another `disk` command.

Current Disk is unformatted.

This message is displayed when you try to execute a command that only makes sense if the *current disk* is formatted. Examples of such commands are the `repair`, `label`, and `backup` commands, which all assume a formatted disk. If you know the current disk is formatted, then something is wrong with your system. Exit `format` immediately, then use diagnostics to locate the problem.

Current Partition Table is not set, using default.

This message can be printed when you run the `label` command. It means that the *current partition table* is not initialized so `format` is attempting to use the default table specified in the data file. If there is no default, `label` will print out a message to that effect and give up. If you do not want to use the default, initialize the *current partition table* with the *partition menu* commands.

Data miscompare error (expecting 0x%x, got 0x%x) at %d/%d/%d, offset = 0x%x.

This error can occur when running the `compare` command. It means that `format` detected a mismatch between an expected data pattern and the actual data pattern on a sector of the *current disk.* `format` then prints out the defective block number and attempts to repair it if automatic repairing is enabled.

Defect file is corrupted, working list set to NULL.

This error occurs when executing the `load` command. It means the specified defect file was found to have inconsistencies such as an improper number of parameters in a defect entry or a bad checksum on the defect file as a whole. `format` therefore assumes that the *working defect list* contains garbage and reinitializes it to NULL. You will have to rebuild the working defect list manually with the `add` command.

Defect file is corrupted.

This error occurs when executing the `load` command. It means that the header of the specified defect file was found to be logically inconsistent. The working defect list is not updated when this error occurs. Check to be sure you specified the defect file correctly. If so, you will have to rebuild the working defect list manually with the `add` command.

Defect file '%s' is not accessible.

This error occurs when executing the `load` command. It means either that you do not have the proper permissions to access the specified defect file or that the defect file doesn't exist. Check the defect file and it's access permissions and retry the command.

Disk is not fully encoded with defect info.

This error occurs you execute the `original` command on an SMD disk. It means that multiple failures have occurred in extracting the manufacturer's defect list from the disk. Although you have the option of continuing, it is recommended that you exit and create the defect list by hand with the `add` command using the printed manufacturer's defect list.

Disk is not fully formatted.

This error occurs when executing the `extract` command. It means that the disk is in a partially formatted state. If you know the current disk is fully formatted, there is something wrong with your system. You should exit `format` immediately and use diagnostics to locate the problem. It is possible to continue the extraction after this error occurs, but it is not recommended.

Error: found disk attached to unsupported controller type '%d'.

This error occurs when `format` cannot identify the controller attached to one of the disks. This should never happen on a Sun supported controller.

Error: unable to malloc more space.

This error can occur any time there is not enough memory to satisfy a request by `format` for temporary storage. If you are running `format` online, try to re-run it when there is less activity on the system.

Error: unexpected ioctl error from xd driver.

This error indicates an internal inconsistency in `format.` Try rebooting and running `format` again. If the error persists, contact your service representative.

Error: unexpected ioctl error from xy driver.

This error indicates an internal inconsistency in `format.` Try rebooting and running `format` again. If the error persists, contact your service representative.

Error: unexpected null partition list.

This error indicates an internal inconsistency in `format.` Try rebooting and running `format` again. If the error persists, contact your service representative.

Error: unknown address space type encountered.

This error can occur when you execute the `disk` command. It means that the definition for the type of address space the controller uses (e.g. VME24D16) is not currently defined within the Sun architecture. This should never happen on a Sun supported controller.

Error: unknown input type.

This error indicates an internal inconsistency in `format.` Try rebooting and running `format` again. If the error persists, contact your service representative.

Error: unknown severity value from xd driver.

This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Error: unknown severity value from xy driver.

This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Expecting ',', found '%s' -- line %d of data file.

This error can occur when `format` scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the "Data File Format" section.

Expecting ':', found '%s' -- line %d of data file.

This error can occur when `format` scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the "Data File Format" section.

Expecting '=', found '%s' -- line %d of data file.

This error can occur when `format` scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the "Data File Format" section.

Expecting keyword, found '%s' -- line %d of data file.

This error can occur when `format` scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax and a list of the keywords, see the "Data File Format" section.

Expecting value, found '%s' -- line %d of data file.

This error can occur when `format` scans the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the "Data File Format" section.

Extraction failed.

This error can occur when you are executing the `original` or `extract` commands. It means that `format` was unable to extract a complete defect list from the *current disk.* The error messages which precede this one may shed light on the problem. You can create a new defect list by using the `add` command.

Format failed.

This error can occur when you execute the `format` command. It means that `format` could not format the *current disk.* The error messages which precede this one should shed light on the problem.

Hard sector count less than nsect.

This error can occur when executing either the `format` or `repair` commands on an SMD drive. It means the value specified in the disk type for the number of sectors per track is greater than the value claimed by the controller. Check your numbers to make sure they agree with what the controller expects. Also check to be sure the *current disk type* is set correctly.

Incomplete specification -- line %d of data file.

This error can occur during `format's` processing of the data file. It means that `format` has found one or more missing parameters from the specified line in the data file. For a complete explanation of the data file syntax and a list of required parameters, see the "Data File Format" section.

Must specify disk and type as well as partition.

This error can occur when you specify a *partition table* in the command line. It means that you must specify the disk and type as well as the *partition table,* so `format` will know the name and type of the target disk. Reenter the command line and continue.

Must specify disk as well as type.

This error occurs when you specify the disk type on the command line. It means that you must specify the disk as well as the type. Reenter the command line and continue.

No current disk.

This message is displayed when you execute the `current` command without having first set the *current disk.* You can set the current disk with the `disk` command.

No default available, cannot label.

This error can occur when you run the `label` command without having specified a *current partition table.* It means that `format` tried to use the default partition table from the data file but could not find one. Define the *current partition table* and rerun the `label` command.

No default for this entry.

This error can occur when you use `format` interactively. It means that you entered a carriage return when there was no default entry for the current input. Specify the value explicitly.

No defects to delete.

This error can occur when you run the `delete` command. It means that the *working defect list* has no entries. Either there are actually no defects in the list, or the *working defect list* is not initialized.

No defined partition tables.

This error can occur when you execute the load command. It means that there are no pre-defined partition tables for the *current disk type*. You can create a partition table for the disk type by using other commands in the *partition menu.*

No disks found!

This error can occur during format's startup processing. It means that format can't find any of the disks in its search path. Make sure you are logged in as the super-user. Also check to be sure the /dev entries for the disks exist, and use MAKEDEV to create them if necessary.

No working list defined.

This error can occur when you execute the print or dump commands with no *working defect list* defined. You can define a *working defect list* with the other commands in the *defect menu.*

Operation on mounted disks must be interactive.

This error occurs when you execute the disk command from a command file and the disk you select has mounted partitions. format will not let you proceed under these circumstances. Note: although this is permitted in interactive mode, it is strongly recommended that you never run format on a disk that has mounted partitions unless you *have* to.

Repair failed.

This error can occur when executing commands from the *analyze menu* , as well as from the format and repair commands. It means that the attempted repair of a sector has failed. This message should always follow a more specific error message. Refer to the documentation for the other message(s) to diagnose the problem.

Search path redefined -- line %d of data file.

This error can occur when format scans the data file. It means that the search path at line '%d' is not the only definition in the data file. Remove the extra definition(s) and restart format .

Specified table '%s' is not a known table.

This error can occur when you specify a partition table in the command line that format cannot find. Check to make sure that the table you specified really exists.

Specified type '%s' is not a known type.

This error can occur when you specify a disk type in the command line that format does not support. Check to make sure that you typed the disk type correctly.

Unable to find specified disk '%s'.

This error can occur when format cannot find the *current disk* you have specified. Check to make sure that you have typed the disk name correctly. You can use the disk command to get a list of the disks format has found.

Unable to find suitable alternate sector.

This error can occur when you are repairing a defective sector on an SMD disk. This can happen from repair, format or any of the surface analysis commands. It means that format was not able to find an alternate sector to replace the defective one. This is usually because mapping is not allowed on certain areas of the disk. There is nothing you can do about this.

Unable to get tty parameters.

This error can occur when `format` fails in an attempt to obtain the current parameters from the terminal you are running from. This generally means that something is seriously wrong. Reset your terminal and try rerunning `format`.

Unable to locate defect #%d.

This error can occur when executing the `format` command on an SMD disk. It means that either `format` could not read the headers for the track containing the defect, or that the header information returned was garbage. In general, this means that a hardware error has occurred on the controller and/or the disk. Run hardware diagnostics to find the problem.

Unable to locate defect.

This error can occur when you are repairing a defective sector on an SMD disk. It means that either `format` could not read the headers for the track containing the defect, or that the header information returned was garbage. In general, this means that a hardware error has occurred on the controller and/or the disk. Run hardware diagnostics to find the problem.

Unable to open '/dev/tty'.

This error can occur when `format` tries to open the terminal special device file. It indicates a problem in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Unable to open command file '%s'.

This error can occur when `format` attempts to open the command file you have specified. Check to make sure that the file exists, that you have permission to read it, and that the filename has been specified correctly.

Unable to open data file '%s'.

This error can occur when `format` attempts to open the data file you have specified. Check to make sure that the file exists, that you have permission to read it, and that the filename has been specified correctly.

Unable to open defect file '%s'.

This error can occur when executing the `dump` or `load` commands. It means that `format` failed to open the defect file you specified. Check to make sure that the filename has been specified correctly.

Unable to open log file '%s'.

This error can occur when `format` attempts to open the log file you have specified. Check to make sure that the filename has been specified correctly.

Unable to open mount table.

This error can occur when `format` checks to see if the *current disk* has any mounted partitions on it. This error shows an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Unable to read drive configuration.

This error can occur when executing the `format` or `repair` commands on an SMD disk. It means that the controller cannot read the configuration information from the disk. In general, this means that a hardware error has occurred on the controller and/or the disk. Use diagnostics to find the problem.

Unable to read track headers.

This error can occur when executing the `format` or `repair` commands on an SMD disk. It means that the controller cannot read the track headers from the disk. In general, this indicates that a hardware error has occurred on the controller and/or the disk. Use diagnostics to find the problem.

Unable to repair defect #%d.

This error can occur when executing the `repair` command. In general, this indicates that a hardware error has occurred on the controller and/or the disk. Check the cabling and, if necessary, run hardware diagnostics to find the problem.

Unable to repair track headers.

This error can occur when executing the `repair` command on an SMD disk. It means that `format` could not write back the repaired header for the track containing the defect. In general, this indicates that a hardware error has occurred on the controller and/or the disk. Use diagnostics to find the problem.

Unable to set tty parameters.

This error occurs when `format` fails in an attempt to reset your terminal parameters. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Unable to write track headers.

This error can occur when executing the `format` or `repair` commands on an SMD disk. It means that `format` could not write a track header to the *current disk*. In general, this indicates that a hardware error has occurred on the controller and/or the disk. Use diagnostics to find the problem.

Value '%s' is not a known ctlr name -- line %d of data file.

This error occurs when the data file contains a reference to an unknown controller. Check to make sure that the spelling is correct and that the controller is supported by the `format` program. For a complete explanation of the data file syntax and a list of supported controllers, see the "Data File Format" section.

Value '%s' is not a known disk name -- line %d of data file.

This error occurs when the data file contains a reference to an unknown disk type. Check to make sure that the spelling is correct and that the disk type is supported by the `format` program. For a complete explanation of the data file system, see the "Data File Format" section.

Value '%s' is not an integer -- line %d of data file.

This error occurs when `format` is parsing the data file. It means that there is a syntax error in the data file at the specified line number. For a complete explanation of the data file syntax, see the "Data File Format" section.

Warning: disk not set for slip-sectoring.

This warning can occur when you execute the `repair, extract,` or `format` commands on an SMD disk. It means that the *current disk* is not set up for slip-sectoring. It is strongly recommended that you set up all the disks for slip-sectoring. See the disk and controller manuals for more information.

Warning: error saving bad block map table.

This warning can occur when executing the `format` and `repair` commands on an SMD disk. It means that `format` attempted to write the bad block map table to the *current disk* but failed. In general, this indicates a hardware error. Use diagnostics to find the problem.

Warning: error saving defect list.

This warning can occur when executing the `format` and `repair` commands. It means that `format` attempted to write out the defect list to the *current disk* but failed. In general; this indicates a hardware error. Use diagnostics to find the problem.

Warning: error telling SunOS bad block map table.

This error occurs when `format` tries to update the bad sector mapping information for an SMD disk. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error telling SunOS drive geometry.

This error occurs when `format` tries to update the drive geometry information for the *current disk.* This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error telling the SunOS drive type.

This error occurs when `format` tries to update the drive type information for an SMD disk. This error only occurs on the Xylogics 450/451 controller. This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error telling SunOS partition table.

This error occurs when `format` tries to update the partition table information for the *current disk.* This error indicates an internal inconsistency in the SunOS kernel. Try rebooting and running `format` again. If the error persists, contact your service representative.

Warning: error writing backup label.

This warning can occur when executing the `format` or `label` commands, as well as several of the surface analysis commands. It means that an attempt to write the backup disk labels for the *current disk* failed. In general, this indicates a hardware error. Use diagnostics to find the problem.

Warning: error writing primary label.

This warning can occur when executing the `format` or `label` commands, as well as several of the surface analysis commands. It means that an attempt to write the primary disk label for the *current disk* failed.

Warning: unable to pinpoint defective block.

This warning is issued from several commands in the *analyze menu,* as well as from the `format` command. It means that surface analysis encountered a media related error, but was unable to pinpoint exactly which sector is defective because the error did not reoccur upon closer inspection. You can repair the sector in the error message if you want to be safe, or you can wait until the defect becomes more evident.

Warning: unable to restore original data.

This warning is issued from the `test` command. It means that the original data on the disk was overwritten during a scan of the disk and that the attempt to rewrite the original data failed. This message should be accompanied by other messages which will illuminate the cause of the failure.

Warning: unable to save defective data.

This warning can occur when executing `format`, `repair`, or any of the surface analysis commands. It means that `format` repaired a bad sector and tried to save the data in it, but failed. There is not much that can be done under these circumstances except to restore the damaged file from backup media.

Warning: unable to save track data.

This warning can occur when executing the `format`, `repair`, or any of the surface analysis commands. It means that `format` repaired a bad sector and found that it was unable to save the data for the rest of the track past the bad sector. This may mean that the rest of the track is bad as well. At this point, you should backup the disk and do surface analysis to test the disk media.

Warning: working list modified but not committed.

This warning can occur when you exit the *defect menu.* It means that you have modified the *working defect list* but have not committed the list. To set the *current defect list* equal to the *working defect list,* run the `commit` command.

Working list was not modified.

This warning can occur when you execute the `restore` command to set the *working defect list* equal to the *current defect list.* It means that no changes have been made to the *working defect list* since the last `restore` command, so the *working defect list* is already up to date.

'%d' is out of range.

This error can occur whenever `format` is expecting input. It means that a value given to the program was out of the bounds specified for the given parameter. To see the bounds of any input, enter a question mark. Then simply enter the correct value.

'%d/%d/%d' is out of range.

This error can occur whenever `format` is expecting input. It means that a value given to the program was out of the bounds specified for the given parameter. To see the bounds of any input, enter a question mark. Then simply enter the correct value.

'%s' is ambiguous.

This error can occur any time format is expecting input. It means that the input specified is not sufficient to identify the choice made. To see the choices for any input, simply enter a question mark.

'%s' is not an integer.

This error can occur any time `format` is expecting input. It means an integer is required, but something else was specified. Correctly reenter the input.

'%s' is not expected.

This error can occur any time `format` is expecting input. It means the input specified does not match any of the choices. To see the choices available, simply enter a question mark.

# 11

# Adding Hardware to Your System

# 11

# Adding Hardware to Your System

This chapter explains many aspects of adding hardware to your system. It covers how to add a new board to the system, how to add peripheral devices like terminals and modems to a serial port, and how to add a printer to the system.

In most cases adding new hardware to your system is a three-tiered process.

1. You connect the actual hardware piece as appropriate.
2. If necessary, you reconfigure the kernel.
3. You edit files on your system to adapt it to the new device. For each kind of new hardware device, the process is explained in the appropriate section below.

The first kind of hardware device covered is a circuit board. You will probably have to reconfigure your kernel when a board is added.

Next, a section explains general procedures for adding devices to asynchronous serial ports, followed by specific instructions for terminals and modems.

Finally, the chapter explains how to add a printer, and gives some explanation of the line printer spooling system as implemented under SunOS. You can hook up printers on a serial port or connect them directly to a controller board that drives a given model; each case is covered.

## 11.1. Adding a Board to Your System

**Note** When using two Ethernet boards in a machine, the second Ethernet board uses the same address as the first board.

When you add a new board to any Sun system, you need to do at least three things:

1. You must insert the board itself in the computer's card cage.
2. You must add a device driver (a group of procedures or routines) to the kernel.
3. You must modify the file system to accept the new device.

This section explains the last two procedures: adding a device driver and modifying the file system. For each board Sun supports, there is an explanation of card cage installation in the *Hardware Installation Manual* for your Sun system.

## Kernel Modification

Reconfiguring the kernel was fully described in the previous chapter. This next section explains how to modify an existing kernel in order to add new equipment.

### Kernel Configuration for Servers

The kernel contains *device drivers*, which are programs that help the kernel pass information between the system hardware and the system software. When you add a new board to your computer, you also need to add or enable the line in the kernel configuration file that describes the particular device driver that provides the interface between the kernel and the IC board. Then, when you reconfigure the new kernel, the device driver program is enabled in the kernel. Remember that the more device drivers you include in the kernel, the more space in main memory the kernel will take up. Therefore, make sure the kernel configuration file only enables device drivers needed by the equipment on your system.

Below is an example of an entry in the kernel configuration file for a Sun-3 color board. To add a color board device driver to the kernel, include this line in the kernel configuration file.

```
device      cgtwo0 at vme24d16 ? csr 0x400000
```

Chapter 9 contains a copy of GENERIC kernel configuration files for a Sun-2, Sun-3, and Sun-4. For further explanation, each device listed in `/usr/share/sys/sun`*[2,3,4,]*`/conf/GENERIC` is documented in *Section 4, Special Files*, in the *SunOS Reference Manual.* For example, if you looked up `cgone` in that manual, you would find out, among other things, what values are mandatory for specified device driver fields, and how much memory the board consumes.

For boards that Sun does not support, you will need to write your own device driver and place an entry for it in the configuration file. This procedure is described in *Writing Device Drivers* It requires expert understanding of the kernel. You can also obtain device drivers from the board manufacturer or distributor.

When you have determined the device driver information for the new board to add to the kernel configuration file, you will need to do the steps shown below to complete the kernel reconfiguration process. These steps explain the process on a system named "grendel:"

**Note:** The following procedure should be used only if the device you are adding already has a driver installed in the kernel. If you are adding new hardware that goes with a new driver, you must also edit the `conf.c` file and the `/usr/share/sys/sun`*[2,3,4]*`/conf` file, as explained in the *Writing Device Drivers* manual.

1. Change to `/usr/share/sys/sun[`*2,3,4*`]/conf` directory and make a copy of the current kernel configuration file to keep until the new one is installed and running.

```
# cd /usr/share/sys/sun[2,3,4]/conf
#cp GRENDEL old.GRENDEL
```

If you have not built your own kernel, you will have to base your kernel configuration on the GENERIC kernel configuration file and make preparations for building your new kernel:

```
# cd /usr/share/sys/sun[2,3,4]/conf
# cp GENERIC GRENDEL
# chmod +w GRENDEL
# mkdir ../GRENDEL
```

2. Edit the kernel configuration file, adding a device driver entry for the board you have installed into the card cage. Look in *Section 4* of the *SunOS Reference Manual* for specifications for the boards that Sun supports.

3. Run `/usr/etc/config` on the new kernel configuration file.

```
# /usr/etc/config GRENDEL
```

4. Build the new kernel.

```
# cd ../GRENDEL
# make
```

5. Now install the new kernel and try it out.

   First move the original working kernel to another (safe) place, then copy the new kernel to the place of the original, and finally boot up the system with this new kernel.

```
# mv /vmunix /vmunix.old
# mv vmunix /vmunix
# /etc/halt
```

   The system goes through the halt sequence, then the monitor displays its prompt, at which point you can boot the system by going through the abort sequence, then typing **b** at the monitor prompt.

   The system boots up multiuser, and then you can try things out.

6. If the new kernel does not seem to be functioning properly, halt the system and boot from the original kernel. Then reinstall the original kernel. Once you are booted up on the original, you can go about trying to fix the faulty kernel.

### Adding Devices

You must be superuser to make the following system modifications.

The kernel communicates with devices through a special file in the directory `/dev`. When a new board is added to the system, a new entry for it must be made in the `/dev` directory. This is relatively easy to do with a shell script called `MAKEDEV`(8), located in `/dev`. `MAKEDEV` takes an argument that is the device-name of a device supported by the system, and automatically installs the files for that device in the `/dev` directory.

For example, if you are adding a Sun-2 color board, you will need to run `MAKEDEV` like this:

```
# cd /dev
# MAKEDEV cgtwo0
```

The following table gives a list of the arguments MAKEDEV accepts for the boards supported by Sun. It also includes the name of the device driver that is listed into the kernel configuration file — the MAKEDEV argument and the device driver name are often identical, but in several cases are different so be attentive. Where an asterisk appears, it means that a logical number — '0' being the first, '1' being the second, and so forth — should replace it.

Table 11-1 *Sun Supported Tape Controller Boards*

| MAKEDEV Argument | GENERIC File Device Driver | Description of Device |
|---|---|---|
| ar* | ar* | Sun's Archive 1/4" Controller |
| st* | st* | Sysgen SC-4000 1/4" SCSI to QIC 02 Controller |
| st* | st* | Emulex MT02 1/4" SCSI to QIC 36 Controller |
| tm* | tm* | 1/2" 1600bpi, CPC Tapemaster |
| mt* | mt* | 40MB 9 track reel, CDC 92181 (Keystone) |
| tm* | tm* | 1/2" 1600bpi, CPC Tapemaster 40MB 9 track reel, CDC 92181 (Keystone) |
| xtc* | xtc* | 1/2" Xylogics 472 |
| xt* | xt* | 150MB 9 track reel, 6250bpi Fujitsu M244X |
| xtc* | xtc* | 1/2" Xylogics 472 40MB 9 track reel, 1600 bpi CDC 92181 150MB 9 track reel, 6250bpi Fujitsu M244X |

Table 11-2 *Sun Supported Disk Controller Boards*

| MAKEDEV Argument | GENERIC File Device Driver | Description of Device |
|---|---|---|
| xyc* | xyc* | SMD Controller, Xylogics 450 |
| xyc* | xyc* | SMD Controller, Xylogics 451 |
| xy* | xy* | Xylogics 450/451 SMD |
| sd* | sd* | Adaptec SCSI Disk |

Table 11-3 *Sun Supported Terminal Multiplexor Boards*

| MAKEDEV Argument | GENERIC File Device Driver | Description of Device |
|---|---|---|
| ttys | zs2-3 | First Multibus SCSI board UARTS ttys0-ttys3 |
| ttyt | zs4-5 | Second Multibus SCSI board UARTS ttyt0-ttyt3 |
| mti* | mti* | Systech MTI-800/1600 or Sun ALM |

Table 11-4 *Sun Supported Ethernet Controller Boards*

| MAKEDEV Argument | GENERIC File Device Driver | Description of Device |
|---|---|---|
| ec* | ec* | 3COM Ethernet Board |
| ie* | ie* | Intel Ethernet Board |
| le* | le* | LANCE Ethernet board |

Table 11-5 *Sun Supported Printer Boards*

| MAKEDEV Argument | GENERIC File Device Driver | Description of Device |
|---|---|---|
| vpc* | vpc | Versatec & Centronics (Systech VPC-2200) |
| vp* | vp | Versatec (Ikon interface) |

Table 11-6 *Sun Supported Graphics/Windows Boards*

| MAKEDEV Argument | GENERIC File Device Driver | Description of Device |
|---|---|---|
| cgone* | cgone* | Sun-1 color graphics board |
| cgtwo* | cgtwo* | Sun-3/Sun-2 color graphics board |
| cgfour* | cgfour* | Sun-3 color graphics board |
| bwone* | bwone* | Sun-1 black & white graphics board |
| bwtwo* | bwtwo* | Sun-2/Sun-3 black & white graphics device |
| gpone* | gpone* | Sun-3/Sun-2 graphics processor board |

Table 11-7 *Miscellaneous Sun Supported Boards*

| MAKEDEV Argument | GENERIC File Device Driver | Description of Device |
|---|---|---|
| sky | sky | Sky FPP board |
| fpa | fpa | Sun FPA board |

### Trouble Shooting

Here is a general list of hints for debugging new boards.

1. Check the hardware. Re-seat the board to assure electrical contact. Check hardware manuals to make sure all switch and jumper settings are proper.
2. If you boot the system with a copy of the GENERIC kernel after installing a device that Sun supports, and you still have the problem, it is probably a hardware problem. If the problem disappears using GENERIC, you should begin looking in the kernel or in `/dev` for the problem.
3. Make sure the new device is included correctly in the kernel configuration file. One way to ascertain this is to look in a file called `/var/adm/messages`, which collects all the messages generated during boot-up. By looking at the last boot-up listing in `/var/adm/messages`, you can determine positively whether the device was included in the kernel and was one of one of the devices the boot-up procedure found.
4. Do an `ls -l` in the `/dev` directory to make sure the new device is installed. The `ls -l` will also show the permissions and major and minor numbers for each device present. Check the permissions for the device against those found in the section of the `MAKEDEV` script that installs the device in question; they should match. Also make sure the device major and minor numbers match those in `MAKEDEV`.
5. Try the simplest commands to see if ANYTHING is working. The closer you come to isolating the spot where the system breaks down, the fewer places you need to look to make fixes. For example, see if a new printer board works by simply "`cat`–ing" a file to it:

```
% cat myfile > /dev/lp
```

   rather than trying to pipe a job through text processors to `lpr`, where the job could fail for a reason that has nothing to do with the new board.

## 11.2. Connecting Devices to Asynchronous Serial Ports

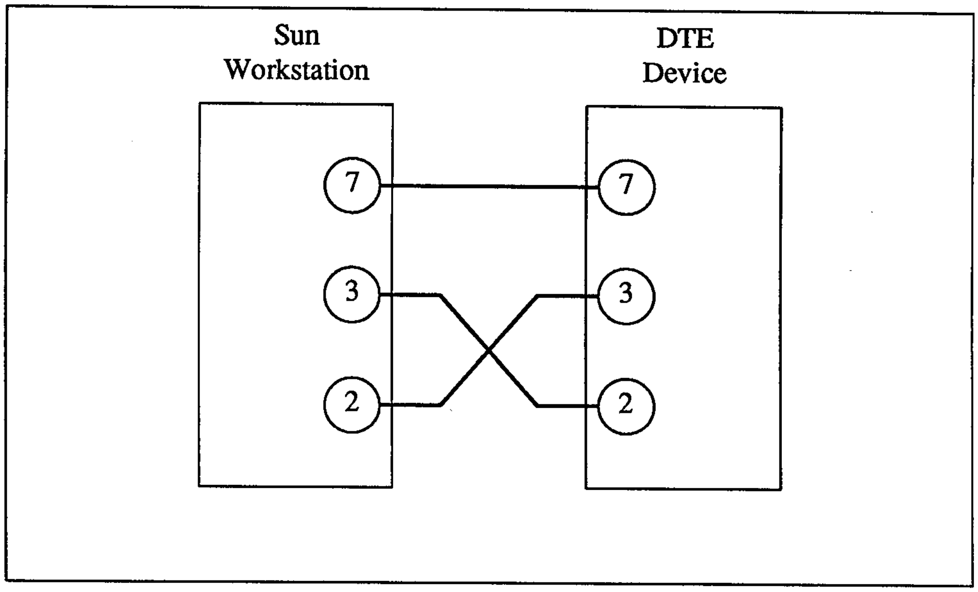
Please read the section called *Asynchronous Serial Ports* in the *Hardware Installation Manual* for your machine before proceeding further. There you will find a discussion of serial port signals for your machine. Become familiar with the hardware specifications for your Sun model.

### General Theory

This section introduces the general topic of connecting equipment to serial ports and discusses some of the terminology. For specific treatment of how to add terminals, modems or printers, see those sections below.

Put simply, a serial port sends a byte of information bit by bit over a single line. All of the Sun serial ports have RS-232-C cabling conventions, but use RS-423 signaling. You can attach modems, terminals, printers, plotters, or other peripheral serial devices that accept the RS-423 signaling to the serial port connectors on your machine. All ports on Sun machines are wired as DTE — data terminal equipment. This means that the data transmit signal from the port is on pin 2 and the receive data from the peripheral is on pin 3. To connect a Sun workstation directly to a terminal, printer, computer, or other DTE device, use a null modem cable that crosses lines 2 and 3, thereby enabling the proper signal connection at the other end. See Figure 11-1 below.

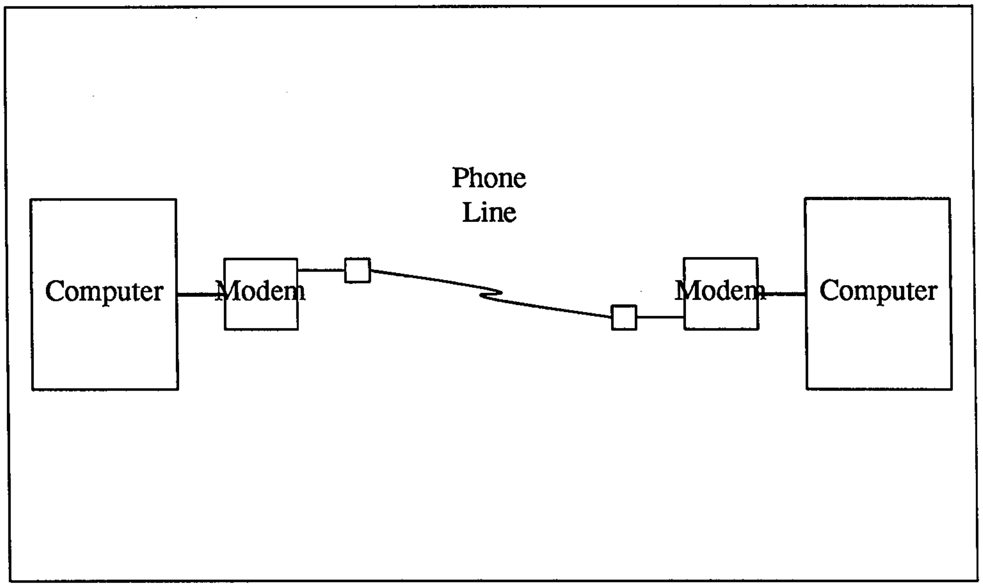
Figure 11-1 *Basic Null Modem*



Some serial interfaces require special additional wiring, for example, the MTI board requires pin 5 to be asserted. This can be satisfied if the peripheral supplies an appropriate signal; otherwise, pin 4 and 5 can be connected at the MTI connector end of the cable.

The Sun workstation is connected to phone lines by adding a modem. You can connect the modem directly to the serial port. Modems are wired as DCE (data communication equipment) devices. A DCE receives serial data from the DTE on pin 2 and sends it down the telephone line; a DCE receives telephone data and sends it out on pin 3 to the DTE. See Figure 11-2 below for a depiction of how two computers would transmit signals through modems along cables and phone lines.

Figure 11-2 *Communications Through Modems*



When you connect a peripheral device to a serial port on the workstation, make sure the cable connection between the serial port and the device is properly installed. Next make sure the appropriate serial device is configured in the kernel. Note: the GENERIC kernel contains all the Sun supported device drivers. If you have reconfigured or customized the kernel, you may need to add back the appropriate serial driver. See Chapter 9 and Chapter 16 for more information about kernel configuration. If specific kernel changes are necessary, they are discussed in the following sections.

You always need a special file in `/dev` for the kernel to run any device. There is an easy way to make this special file: you use the `/dev/MAKEDEV` command, which installs a special file for new devices. The command is explained in `MAKEDEV(8)`.

## General Debugging Hints

Here we discuss two general areas where problems occur when you have added new peripheral devices to the system.

**Hardware Changes** — If you have added new hardware and there are problems, check the cabling and connections first. Is everything tight? Once again verify that the new device accepts the RS-423 communication protocol, and consult the product manufacturer's manual to make sure it has been installed properly. Are the workstation and the new device set up for compatible communication? The Sun defaults to the following: 7 data bits, even parity, 9600 baud rate, flow control enabled (XON/XOFF). Check which control signals the other equipment expects.

Next, check the condition of the cable. Here you may want to refer once again to your Sun Workstation *Hardware Manual* to determine what signals are sent from and received at specific ports on your board. To check for cable problems it is helpful to have a *breakout box*. A breakout box plugs into the RS-232 cable, providing a patch panel that allows you to connect any pin to any other pin(s); it will often contain LEDs, which display the presence or absence of a signal on each pin. Sometimes a cable has become corrupt and signals are being flipped randomly. In this case a line may receive when it should transmit, or the reverse. These suggestions should give you a start on troubleshooting. Problems specific to terminals are covered in the *Adding Terminals* section below.

**New Software** — If you have problems after installing a new system kernel along with the new peripheral device, there could be something wrong with the new kernel. This is likely to be a total non-response symptom rather than the intermittent weirdness of a bad line. Back out the new kernel and install the GENERIC kernel in its place, then boot the system with GENERIC. If the problem disappears with GENERIC, you have a problem with the new kernel — possibly you did not add the device driver, or did not add it properly. If the problem persists with GENERIC, it is probably in the cabling or the hardware.

## 11.3. Adding a Terminal to Your System

Please read the section above called *Connecting Devices To Asynchronous Serial Ports* before proceeding further. The general discussion there will prepare you to install a new terminal on your system.

This section explains the steps necessary for adding a terminal, and gives some hints on what to do if you run into problems.

### Serial Port and Cable Connection

Make sure you have an available serial port on your workstation. The number of serial ports varies from one Sun model to another. Each Sun workstation provides at least two asynchronous serial ports controlled by the Serial Communications Controller on the Sun CPU board. If all the existing serial ports are in use, and you are not planning to disconnect any current peripheral devices to make a port available, you will need to add a new board to the card cage to increase the number of serial ports. The instructions for adding a board are given above in the section *Adding A Board To Your System.*

When you do have a port available, connect the terminal to the workstation with a null modem cable. A basic null modem cable swaps lines 2 and 3 so that the proper transmit and receive signals are communicated between two Data Terminal Equipment (DTE) devices. Line 7 goes straight through.† Make sure all the connections are secure and check control signals.

### Kernel Modification

You must be superuser to make kernel modifications. If you are adding a first terminal, make sure that the system kernel contains a `zs0` device driver to run the CPU ports (this is standard); if you are adding terminals to a Multibus SCSI board make sure the proper `zs*` driver entry is in the kernel configuration file; if you are adding terminals to a Sun ALM or a Systech multi-terminal interface board (MTI), make sure the proper `mti` driver entry is in the kernel configuration file.

† The Systech MTI board requires pin 5 asserted. The simplest way to get this is to connect pins 4 and 5 at the connector going to the MTI port.

If you had to add a new board to make ports available for your terminals, you have probably made necessary kernel changes already, as outlined above in the section *Adding A Board To Your System.* That section explains how to edit and reconfigure the system kernel.

Use dmesg (8) to look at system diagnostic messages to make sure the system is configured the way you want it.

Look in your kernel configuration file in the /usr/share/sys/sun[2,3,4]/conf directory to find the device drivers for which your system is configured. An explanation of the device driver entries in this file is given in Chapters 9 and 16. For terminal connection, make sure the flags bit is set for a hardwired line on the port — set to on.

## Adding Devices

The SunOS operating system needs a special file in /dev to run any device. For serial ports, these are known as /dev/tty* — where the asterisk is a letter or number value or both. When you add a board you have to run the MAKEDEV shell script to install the special files for it in /dev. If you are connecting to available ports on a particular board already in use, then the MAKEDEV has already been run and you can ignore the following information.

The MAKEDEV command takes an argument that is the device name for the device in the kernel. In the case of a serial port board, one /dev/tty* file is created for each terminal interface on the board. The table shows examples of the MAKEDEV command for the boards that Sun supports, and the names of the entries MAKEDEV creates in /dev.

Table 11-8 *Using* MAKEDEV *For New Boards*

| *Type* | *Command* | *Devices* |
|---|---|---|
| Sun CPU Board | MAKEDEV std | ttya,ttyb |
| 1st Multibus SCSI Board | MAKEDEV ttys | ttys0 — ttys3 |
| 2nd Multibus SCSI Board | MAKEDEV ttyt | ttyt0 — ttyt3 |
| Systech MTI-800 Board | MAKEDEV mti0 | tty00 — tty07 |
| Systech MTI-1600 Board | MAKEDEV mti0 | tty00 — tty0f |

Next you will need to edit /etc/ttytab, the file that tells the system about the new terminal. This file is read by the init program and specifies which serial ports will have a login process created for them.

There should be one line in /etc/ttytab for every serial port.* The name of the port in /etc/ttytab is the same as its name in /dev. The /etc/ttytab file looks in part like this:

There are also ttyp*, ttyq*, ttyr*, and ttys* lines. These are required for the window system and other programs and must appear as installed by the installation procedure, except that the "secure" status may be changed.

```
#
# name   getty                              type          status  comments
#
console "/usr/etc/getty std.9600"           sun           on secure
ttya    "/usr/etc/getty std.9600"           925           on
ttyb    "/usr/etc/getty std.9600"           unknown       off
   .
   .
   .
ttyd0   "/usr/etc/getty Fast-Dial-1200" dialup            on
```

A comment begins with the character "#" and continues to the end of the line.

The first field of a line is the name of the device in /dev.

The second field is the program that init should run on this line, together with all its arguments. Normally, /usr/etc/getty, which performs such tasks as setting the baud rate, reading the login name, and calling login, is run; its argument is the name of the baud rate code from /etc/gettytab to be used. In the file /etc/gettytab you will find the values your system recognizes for different baud rates. For example:

```
c|std.300|300-baud:\
        :nd#1:cd#1:sp#300:
f|std.1200|1200-baud:\
        :fd#1:sp#1200:
2|std.9600|9600-baud:\
        :sp#9600:
```

are the /etc/gettytab entries for initialization of terminals at three different baud rates. Therefore, "std.9600" as the argument of getty would set the rate to 9600, and "std.1200" would set it to to 1200.

The third field is the type of terminal on that port. To find the kind of terminal types your system recognizes, look in the file /usr/share/lib/termcap (see termcap(5)). This is a database describing the capabilities of terminals and the way operations are performed on them. If your terminal type is found in /usr/share/lib/termcap, use it. If your new terminal has an emulation mode for one of the terminal types found in your termcap file, put that name into /etc/ttytab and set the corresponding emulation mode on the terminal. If your terminal type is not in /usr/share/lib/termcap, you may create your own termcap entry. The type "dialup" should be used for phone-in lines. The information in /etc/ttytab is read at login time by login to initialize the TERM environment variable.

The fourth field is either "on" or "off", optionally followed by some flag values. If it is "off", init will ignore that line, and no logins will be able to occur on that line; if it is "on", init will create a login process for that line. The flag fields are separated from "on" or "off" and from each other by spaces. The flag "secure" indicates that the terminal is "secure"; the super-user ("root") may only log in on "secure" terminals. (Note that if the console terminal is note flagged as "secure", when entering single-user mode the system will require the super-user password to be typed before the shell is started.) The flag "`window=`*program*" indicates that the specified *program* together with its arguments is to be run by `init` before it runs the program (such as `getty`) that it would normally run; the *program* and its arguments are specified by a quoted string. This is used, for example, if the program that `init` would normally run requires a window system to be started.

The sample `/etc/ttytab` file above indicates that the console is a Sun Workstation console, and is a "secure" terminal. The terminal attached to `/dev/ttya` is a Televideo TVI-925, running at 9600 baud, and is not "secure". The terminal type of the terminal attached to `/dev/ttyb` is unknown, and logins are disabled on that terminal. If you were putting a Televideo TVI-925 terminal on `/dev/ttys2` at baud rate 9600, you would add the following line to the example `/etc/ttytab` file shown above:

```
ttys2      "/usr/etc/getty std.9600"    925     on
```

at the end of the file. This terminal will have a login process started for it at 9600 baud. After editing the `/etc/ttytab` file you **must** notify `init`, which then reads the new information in `/etc/ttytab`. The command to do this is:

```
# kill -1  1
```

Finally, you need to set some switches on the terminal so it can communicate with the Sun. Check to see if you have changed any settings on the Sun from the defaults given here: 7 data bits, even parity, flow control enabled (xon/xoff); set baud rate to the same rate you set in the `/etc/ttytab` file.

If possible, set up the terminal to ignore the parity bit.

### Problems

If you have problems after installation, check the cabling first, as outlined above in the section *Connecting Devices To Asynchronous Serial Ports*. Next, check the information in the `/etc/ttytab` file, and make sure you have restarted `init` as explained above.

If the terminal behaves strangely, check the `/usr/share/lib/termcap` file. You may have unexpected whitespace, or you may have forgotten to escape the newline character in your entry: a backslash (\) followed immediately by a newline (carriage return) will continue a line onto the next.

If you have not correctly set the tty type in the `/etc/ttytab` file, you can set it at your terminal with the command:

```
# set term=sometype
```

There are several ways to check aspects of your terminal environment. The `stty` command typed with no argument will tell you the baud rate of the terminal and the settings of options that are different from their defaults. The `all` argument reports all normally used option settings. The `everything` argument reports everything that `stty` knows about.

The `printenv` command with no arguments prints out the variables in the environment.

You can type `set` with no arguments, and it will report the value of all shell variables.

## 11.4. Adding a Modem to Your System

Please read the section above called *Connecting Devices To Asynchronous Serial Ports* before proceeding further. The general discussion there will prepare you to install a modem on your system.

This section explains the steps necessary for adding a modem, and gives some hints on what to do if you run into problems.

Sun supports three currently manufactured modems: the Ven-Tel 1200-Plus, the Ven-Tel 1200-32, and the Hayes Smartmodem 1200. Ven-Tel modems are run in Hayes compatibility mode. Sun will continue to support the discontinued Ven-Tel MD 212 series, although it alone is not discussed below.

Where this section refers to systems that have both auto-dial and auto-answer capabilities, the references are only to systems after Sun release 1.2. If you are running Sun release 3.2, the software is better suited to use with 2400-baud modems.

### Serial Ports, Cable Connection and Modem Switches

**Serial Ports** — Make sure you have an available serial port on your workstation. The number of serial ports varies from one Sun model to another. Each Sun workstation provides two asynchronous serial ports controlled by the Serial Communications Controller (SCC) on the Sun CPU board. If you also have a Multibus SCSI board in your system, two additional SCC's make four additional serial ports available. If all the existing serial ports are in use, and you are not planning to disconnect any current peripheral devices to make a port available, you will need to add a new Multibus board to the card cage to make more ports available. The instructions for adding a Multibus board are given above in the section *Adding A Board To Your System.*

**Cable Connection** — If you do have a port available, connect the modem to the workstation with an RS-232 cable that has pins 2 through 8 and pin 20 wired straight through. You may also use a 25 pin ribbon cable to connect the ribbon to the system. Make sure all the connections are secure.

The Systech MTI board requires a special cable with pins wired in the following fashion:

```
Systech    Modem
 2 ------------------ 2
 3 ------------------ 3
 4 -|                 4
 5 _|                 5
 6 ---\               6
 7-----\------------- 7
 8      \------------ 8
20 -----------------20
```

**Modem Switches** — There are differences between the Ven-Tel modems and the Hayes Smartmodem 1200. However, with either brand the switch settings shown below work for use with `tip`(1) and `uucp`(1). Always read the manufacturer's manual before attempting to adjust equipment.

First the Ven-Tel 1200-plus models. (Note that this information does not apply to the old Ven-Tel MD 212.) There are two variations of the Ven-Tel 1200-Plus. The older model has two blocks of four switches, one internal, and the other external. Newer models have a single internal 10-switch block. (**`Note:`** The manufacturer may have revised the configuration setting switches.) There is one panel of external switches and one of internal switches on each of the Ven-Tel models discussed here. For the external panel, set all switches to the open or up position. When you open the modem box you will see another panel of switches, which should be set as follows:

- Switch 1 up for hardware DTR.
- Switch 2 up for WECO control signals.
- Switch 3 down to get Hayes protocol.
- Switch 4 can be set according to your preference. Set it up to disable the speaker so you will not hear the high-pitched carrier noise when connecting; set it down to enable the speaker.

The Ven-Tel 1200-32 is somewhat different. Here are the settings for its internal 10-section dip switch:

- Switch 1 off for hardware DTR.
- Switch 2 on for numeric result codes.
- Switch 3 on to send result codes.
- Switch 4 on to not echo commands.
- Switch 5 off to answer incoming calls.
- Switch 6 off for hardware Carrier Detect.
- Switch 7 off — not used.
- Switch 8 off — not used.
- Switch 9 off for no hardware DSR.
- Switch 10 controls the speaker as with the Ven-Tel 1200-plus.

Next the Hayes Smartmodem 1200. The proper switch settings are given below. There is only one switch panel on the Hayes; down is `on` and up is `off`.

- Switch 1 up for hardware DTR.
- Switch 2 down for numeric result codes.
- Switch 3 down to send result codes.
- Switch 4 down to not echo commands.

- Switch 5 up to answer incoming calls.
- Switch 6 up for hardware Carrier Detect.
- Switch 7 up for connection to RJ11 modular jack.
- Switch 8 down enable command recognition.

After changing the switch settings on any model, you must power cycle the modem by unplugging it from the electrical outlet, waiting a few seconds, and then plugging it back in. Looking at the front of a properly installed modem, you should see lights AA and TR lit up when the modem is not in use; and the lights should be lit or blinking when the modem is in use.

Other modems that use the Hayes `AT` command set *may* work with Sun `tip` and `uucp` software. Configure the modem as follows:

- Hardware DTR, that is, when the Sun drops DTR (as when someone logs off) the model should hang up.
- Hardware Carrier Detect, that is, the modem only raises the Carrier Detect (CD) line when there is an active carrier signal on the phone connection, when carrier drops, either when the other end of the connection terminated or in the event the phone connection is broken, the Sun will be notified and act appropriately. The CD signal is also used for coordinating dial-in *and* dial-out use on a single serial port and modem.
- Respond with numeric result codes.
- Sends result codes.
- Does not echo commands.

### Kernel Modification

You must be superuser to make the following kernel modifications.

The kernel contains routines called device drivers that allow the operating system to pass information between system hardware and system software. Users may specify which devices they want to include in the kernel through a process called configuring the kernel. To specify which devices should be included in the system, and exactly how those devices should operate, the user edits a configuration file.

You must edit the device driver specification for the serial communications controller in your configuration file, to enable carrier detect on the line where the modem is attached. This is referred to as turning `off` the software carrier, which then allows the system to detect a hardware carrier on the line. When turned `on` the system treats each line as hard-wired with carrier always present. Turning `on` or `off` is done by specifying a parameter in the `flags` field. For example, below is a sample entry from a kernel configuration file for the serial communications controller on the standard Sun-3 CPU board:

```
device  zs0 at obio ? csr 0x20000 flags 3 priority 3
```

To make the device in this example into a modem-compatible driver, the hexadecimal value 0x3 after `flags` will have to be changed. The same field will have to be changed for other device drivers if your modem is going to be attached to a board other than the Sun CPU board.

Specifically, you need to make the following changes to the device driver specification line in the kernel configuration file. The changes for each of the three boards to which a modem can be attached are given below.

**The Standard Sun CPU Board** — The serial communications controller on the standard Sun CPU board, device `zs0`, provides two lines with full modem control. For both dial-in and dial-out on the same serial port, the `flags` bit in the kernel corresponding to that serial port has to be set to zero. This enables hardware carrier detect so that the Sun can tell when someone dials in or hangs up. The default value of `flags` for `zs0` in the GENERIC kernel is 3, indicating software carrier for both ports a and b. To permit hardware carrier detect on `ttya`, `flags` should be changed to 0x2, on ttyb to 0x1, and on both to 0x0.

**A Multibus SCSI Board** — Each Multibus SCSI board has two serial communications controllers identical to the one on the CPU board, for a total of four lines with modem control on each SCSI board. These are devices `zs[2,3]` for the first board and `zs[4,5]` for the second. Set the `flags` value for each driver according to the rules explained above for the CPU board. Remember that `zs2` is for the lines `ttys[0,1]` and that `zs3` is for the lines `ttys[2,3]`; if you have a second SCSI board the ttys are, by convention, `ttyt[0-3]`.

**A Systech MTI Board** — The Systech MTI-800 and MTI-1600 boards require a kernel configuration line for device `mti0`. As above, the `flags` value is set in hexadecimal. The default `flags` value on installation of either MTI board is 0xffff. This value selects software carrier detect for all lines. For lines with modems, you need hardware carrier detect and must set the corresponding `flags` bit in the kernel to zero, just as in the examples above.

The following example shows how to determine the proper hexadecimal values for the `flag` bits on a a Systech MTI-1600 board where ports zero through 7 are turned on, software carrier detect, and ports 8 through f are turned off, hardware carrier detect for modem operation. The logic here can be applied to all serial ports.

| port | f | e | d | c | b | a | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| on | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| off | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | |

If you take the values shown here, '0000' '0000' '1111' '1111', and convert to hexadecimal, you arrive at the entry for the `flags` parameter: 0x00ff.

For more information on the serial communications driver and the multi-terminal interface, see the pages `zs`(4s) and `mti`(4s) in the *SunOS Reference Manual.*

Here are the steps to edit and install the new kernel on a system named CHAOS. (The following example configures a kernel for a Sun-3 to use a modem on the ttya serial port.)

1. Change to /usr/share/sys/sun*[2,3,4]*/conf directory and make a copy of the current kernel configuration file to keep until the new one is installed and running.

```
# cd /usr/share/sys/sun[2,3,4]/conf
# cp CHAOS old.CHAOS
```

   If you have not built your own kernel, you will have to base your kernel configuration on the GENERIC kernel configuration file and make preparations for building your new kernel:

```
# cd /usr/share/sys/sun[2,3,4]/conf
# cp GENERIC CHAOS
# chmod +w CHAOS
```

2. Edit the kernel configuration file, changing the flags bit, as explained above, for the board to which you will attach the modem. For example, if you want to put the modem on ttya, change the entry that looks like this:

```
device  zs0 at obio ? csr 0x20000 flags 3 priority 3
```

   to an entry that looks like this:

```
device  zs0 at obio ? csr 0x20000 flags 0x2 priority 3
```

3. Run /usr/etc/config on the new kernel configuration file.

```
# /usr/etc/config CHAOS
```

4. Build the new kernel.

```
# cd ../CHAOS
# make
```

5. Now install the new kernel and try it out.

   First move the original working kernel to another (safe) place, then copy the new kernel to the place of the original, and finally boot up the system with this new kernel.

```
# mv /vmunix /vmunix.old
# mv vmunix /vmunix
# /etc/halt
```

The system goes through the halt sequence, then the monitor displays its prompt, at which point you can boot the system by typing **b**.

The system boots up multiuser, and then you can try things out.

6. If the new kernel does not seem to be functioning properly, halt the system and boot from the original kernel. Then reinstall the original kernel. Once you are booted up on the original, you can go about trying to fix the faulty kernel.

Remember to remove the `old.CHAOS` kernel configuration file you created as a backup.

For more information about kernel building, see Chapters 9 and 16.

## System Modification

You must be superuser to make the following system modifications.

**Using mknod** — First create an alternate device for your modem in the `/dev` directory by using `mknod`. This alternate device allows a single tty line to be connected to a modem and used for both incoming and outgoing calls.

There are several arguments which you must give to `mknod`. The first is the name of the device you are making. It will be `cua*`, where the asterisk is the logical value of this alternate device on your system — 0 for the first alternate device, 1 for the second and so forth. The second argument will always be `c`. The last two arguments are the major and minor device numbers. To determine what numbers to use here do an `ls -l` on the `/dev/tty*` device you are making the alternate device for. If you are going to use the first port on your CPU board — `ttya`— do the following:

```
# ls -l /dev/ttya
crw--w--w-  1 root       12,   0 Sep 17 18:27 /dev/ttya
```

Here the major device number is 12 and the minor device number is 0. When running `mknod` to create an alternate device for a modem you always give the actual major device number and you always add 128 to the minor device number. Thus, in this example, the `mknod` command would take as its last two arguments 12 and 128. You also change the mode and ownership of this device. The whole process would look like this:

```
    # cd /dev
    # mknod cua0 c 12 128
    # chmod 600 cua0
    # chown uucp cua0
```

In this example you created alternate device `cua0`, the special file for the first modem attached to a system. The second modem attached would be `cua1`, and so forth.

While still in the `/dev` directory, move the tty device whose software carrier you turned off in the `flags` field of your device driver specification in the kernel above, to a dialing device. This is the same one for which you created the alternate device with `mknod`. In this example you move the first port on the CPU board to the first modem device.

```
# mv ttya ttyd0
```

The second modem device would be ttyd1, and so forth.

**Updating ttytab** — Now edit the /etc/ttytab file (called /etc/ttys in pre-4.0 releases), and add an entry for the new ttyd1 line, and disable logins for the no longer existing ttya device.

A modem suitable for use at both 1200 and 300 baud could use a gettytab entry similar to this:

```
# fast dialup terminals, 1200/300 rotary (can start either way)
#
3|D1200|Fast-Dial-1200:\
    :nx=D300:fd@:tc=1200-baud:
5|D300|Fast-Dial-300:\
    :nx=D1200:tc=300-baud:
```

The resulting /etc/ttys file might look like this:

```
+8u*48u
#
# name  getty                             type          status  comments
#
console "/usr/etc/getty std.9600"         sun           on secure
ttya    "/usr/etc/getty std.9600"         925           on
ttyb    "/usr/etc/getty std.9600"         unknown       off
   .
   .
   .
ttyd0   "/usr/etc/getty 1200-baud"        dialup        on
```

The ttya line has logins disabled. The ttyd0 line has logins enabled at either 1200 or 300 baud — sending a break through the modem causes getty to switch to the other baud rate. If you want to use a modem at a single baud rate only, set up /etc/ttytab as you would a directly connected terminal. For example, this gettytab entry

```
f|std.1200|1200-baud:\
     :fd#1:sp#1200:
```

can be used for a fixed 1200-baud line. Modify /etc/ttytab accordingly:

```
+8u*48u
#
# name  getty                          type          status  comments
#
console "/usr/etc/getty std.9600"      sun           on secure
ttya    "/usr/etc/getty std.9600"      925           on
ttyb    "/usr/etc/getty std.9600"      unknown       off
   .
   .
   .
```

Do not remove the `ttyp*` or `ttyq*` entries in the `/etc/ttytab` file; they are required for the window system and other programs.

After you have finished editing `/etc/ttytab`, you must reinitialize the file by notifying `init` with the `kill` command:

```
# kill -1 1
```

The `/etc/ttys` file is automatically updated to reflect information in `/etc/ttytype`.

**The `/etc/remote` File** — The file `/etc/remote` stores the attributes of systems that you dial out to using `tip`(1). See the "Tip" section in Chapter 15. It is structured somewhat like `termcap`. You must now edit that file. Each line in the file provides a description for a single known system. Fields are separated by a colon (:). Lines ending in a backslash (\) and followed immediately by a newline (carriage return) are continued on the next line. Look in the `/etc/remote` file or see the `remote`(5) manual page in the *SunOS Reference Manual* for further explanation. The example given below would work to connect to a system named "sunphone," at phone number 7630927, at baud-rate 1200, using the Hayes auto call unit protocol, on dialer cua0:

```
sunphone:\
      :pn=7630927%:tc=UNIX-1200:
UNIX-1200:\
      :el=^D^U^C^S^Q^O@:du:at=hayes:ie=#$%:oe=^D:br#1200:tc=dialers:
dialers:\
      :dv=/dev/cua0:
```

**Install** `uucp` — Once your modem is installed and working, you may install `uucp`. See the appropriate sections in Chapter 15, "Electronic Mail and Communications."

**Set Up the Mail System** — Finally you must check to make sure that your machine's mailer configuration is set up properly. See *Setting Up The Mail System* in Chapter 15 for a complete discussion of the mail system. If you are adding a modem to connect a main machine to phone lines, or if you are a standalone machine not on a local network, install the `main` file in `sendmail.cf`. If you are a subsidiary machine on a local network (standalone or client) and planning to stay that way, despite the attachment of phone lines, install the `subsidiary` file in `sendmail.cf`.

## Hayes Specific Considerations

**tip Support** — To use a Hayes modem with tip, you should specify the modem type in the /etc/remote file as either at=hayes or at=at. The phone number attribute (pn) can contain any valid dial commands; see your modem manual for details. The most common commands to the phone number attribute are:

□ A phone number of numeric characters 0-9.

□ A comma (,) will cause a 2 second pause to wait for a secondary dial tone. For example to dial an outside line from a local phone network.

□ A P or T will switch to pulse or tone dialing. By default tip will use tone dialing.

□ You may set parameters for the dialer by starting the phone number with an “S” (for set) flag. Read the Hayes manual for an explanation.

□ Note that tip can cycle through a set of phone numbers, dialing each one until a connection is made. Previously the numbers were often separated with a comma, although this feature was never documented. Now, since comma is a valid dial character, phone numbers must be separated with a vertical bar (|), just like the separator in the name field. For example, the /etc/remote line for pn would look something like:

```
    :pn=2138896565|2138896564|2138896563:
```

**uucp Support** — To use a Hayes (or AT) modem with uucp the modem type in the /usr/lib/uucp/L.sys file should be ACUHAYES.

□ The uucp default is for tone dialing.

□ The phone number may contain any valid dial commands, however uucp uses any alphabetic prefix of a phone number to look up a translation in the L-devices file. Therefore, you must insert a minus (-) before a phone number that begins with a letter. An L.sys line that uses a Hayes modem to call with pulse dialing might look like:

```
adiron Any ACUHAYES 300 -P7620883 login:-EOT-login: uucp ssword: junk
```

To use the default of tone dialing, simply omit the -P.

□ As in tip, you may set parameters for the dialer by starting the phone number with an ‘S’ (for set) flag. Read the Hayes manual for an explanation.

## Problems

□ If you have problems after installation, check the cabling first: as outlined above in the section *Connecting Devices To Asynchronous Serial Ports*, and make sure you have pins 2 through 8 and 20 wired straight through. (If MTI, wire 2, 3, 7, and 20 straight through and have pin 8 on the modem go to pin 6 on the MTI port also you need to tie 4 to 5 on the MTI side. See page 257 for example.) Next, check the information in the /etc/ttys and the /etc/remote files, and make sure you have restarted init as explained above.

- If you cannot access a port, and find a process running on it when you do a `ps -ax`, then make sure you have the 8 pin connected in your cable. If that does not work, check to make sure your device driver is configured properly to set the correct flag for the line to `off`.

- Sometimes even when both the hardware and software are correct, the device driver will get into a state where it will not let the alternate port be opened. You must do a `kill -1 1` to notify `init` and reset the flags on the device driver.

- If you get a

```
can't synchronize with hayes
```

  error message when using a Hayes-compatible modem, check internal and external modem switch settings, and check the the cable connection. Power cycle the modem if necessary.

- If you get a

```
can't synchronize with ventel
```

  error message when using a Hayes-compatible modem, look in the `/etc/remote` file and make sure you have changed `at=ventel` to `at=hayes`.

- The message

```
tip: /dev/cua0: No such device or address
```

  usually means that the device special file is missing from `/dev` (or is incorrect there), or the device driver is missing from your kernel.

- The message

```
all ports busy
```

  may mean that the port is actually busy running `tip` or a dial-in user. You can do a `ps -ax` to see what is running. You should not see a `getty` on the port, except momentarily as the port receives a call.

  If you do see a `getty` with no user dialed in, you have not set up carrier detect properly. Check: the flags bit in the kernel; the cable; the modem settings; the `/etc/ttytype` file for correct device entry. If no process is currently using the serial port, there may be a leftover lock file. Look for a lock file in `/var/spool` and`/var/spool/uucp` and remove it. It will look something like `LCK.cua0`. If you change the switch, unplug and power cycle the modem as explained above. If you get the message when `tip` is not running, no one is dialed in, and there is no lock file, try unplugging the modem cable and/or power cycling the modem. Finally, you can do `ps -ax` and look for a process tying up the port.

- If you get a

```
tip: /dev/cua0: Permission denied
```

or

```
link down
```

error message, make sure you have `:dv=/dev/cua0:` in `/etc/remote`. Check for a lock file in `/var/spool` or `/var/spool/uucp` called `LCK.*` where `*` is the name of the dial out destination. Make sure permission modes on `/dev/cua0` are 600, and it is owned by `uucp`. Try turning off the modem, unplugging it for a minute, and plugging it back in again. Also check permissions on the `/var/spool/uucp` directory. It must exist and be readable, writable, and executable by everybody (777 mode).

## 11.5. Adding a Printer to Your System

The line printer spooling system implemented in SunOS can handle a variety of printers and configurations. It handles local and remote printers, printers attached to serial lines, raster output devices such as Varian or Versatec, and laser printers such as an Imagen or the Sun LaserWriter. It can handle multiple printers and multiple spooling queues.

Some printers require customer software for handling communication between a Sun workstation and printer or for converting data into an acceptable form required by the printer. The Imagen and Sun LaserWriter are examples of such printers. Contact the printer vendor for information on this software; for example, contact Sun for the Transcript software that provides the necessary support for printing on the Sun LaserWriter.

Generally, no special software is required for printing plain text files on printers that are character- or line-oriented and use the standard ASCII character set.

Special filter programs *are* required for printing non-plain text; for example, rasterfiles or `troff(1)` output. Special printer features, for example, special character sets, may also require special software for turning on or off at the appropriate times.

### Serial Port and Cable Connection

This section begins with the hardware configuration. Read the manufacturer's printer manual and check the following on the new printer.

- Determine whether the printer is configured as DTE (data terminal equipment) or DCE (data communication equipment). Set it to DTE if you can.
- If possible, disable the printer's use of modem control signals like CTS (clear-to-send) and DSR (data-set-ready). If the printer requires CTS and DSR, loop back the following lines on the printer: connect line 4 to line 5, and line 6 to line 20.
- If possible, enable xon/xoff flow control.
- At least for the installation phase, set the baud rate at a low rate for testing, 1200 for example.

Now make sure you have an available serial port on your workstation. The number of serial ports varies from one Sun model to another. Each Sun workstation provides at least two asynchronous serial ports controlled by the Serial Communications Controller on the Sun CPU board. If all the existing serial ports are in use, and you are not planning to disconnect any current peripheral devices to make a port available, you will need to add a new Multibus board to the card cage to increase the number of serial ports. The instructions for adding a board are given above in the section *Adding A Board To Your System.*

When you do have a port available, connect the printer to the workstation with a three-line cable. If the printer is a DTE device, use a null-modem cable. Null modem cables have line 7 wired straight through and lines 2 and 3 swapped so that proper transmit and receive signals are communicated between two DTE devices. If the printer is a DCE device, then connect 2, 3, and 7 straight through. Make sure all the connections are tight.

When you have connected the printer as explained above, you should verify that cabling and hardware are performing before proceeding. Obviously, if there is a problem with connection or faulty hardware, none of the later software installation will work.

- □ Consult the operation manual for the printer and run the printer's standalone diagnostics. If these do not run, call the printer manufacturer. Remember to set the switch on the printer back to operate mode after the self-test.

- □ Verify that the printer switch settings are correct. Check for the correct settings in the printer operation manual, and make sure they correspond with the parameters given above for the Sun.

- □ Send something over the tty line to the printer. To do this, set the characteristics of the printer using `stty` (1), and then `cat` (1) a file to the `/dev/tty*` serial port where the printer is attached. If the printer is attached to the first serial port on the CPU board, type:

```
# (stty 1200; cat /etc/passwd) > /dev/ttya
```

to print a copy of the password file.

If the printer is "dead," your cable may be bad, or you may need a null modem cable as described above.

If something all garbled prints, `stty` may not have set all the terminal characteristics properly for the printer. Read the manufacturer's manual and check the switches on the printer. In particular check the baud rate, 1200 in this example. Make sure the printer and the `stty` setting match. You may have to set additional options with `stty` to match those on your printer. The most likely are: to set parity to `even`, `odd` or neither (`space` parity); to set tab expansion with `-tabs`; and to allow carriage return for newline, and output CR-LF for carriage return or newline with `-nl`. These parameters are set for the line printer spooling system software in the `/etc/printcap` file by the `ms` capability. See below for an explanation. Also see `printcap` (4) in the *SunOS Reference Manual.*

## File Modification

You must be superuser to make the following file system modifications.

The Sun software includes the necessary programs to run the line printer system, and a default line printer queue is included. You need to create a /etc/printcap file to make the printer work properly.

## Editing the printcap File

printcap(5) is a database describing printers. It describes line printers directly attached to a machine and printers accessible across a network. Each printer should have an entry, and it is a good idea to remove entries for printers not in your system. Typically, a system will have just a single printer. The syntax of entries in printcap can seem torturous; check carefully after you have modified the file. Explanations and examples are given below.

printcap entries consist of fields that (except for the name field) **must** be preceded and followed by colons ( : ); the last field specified must terminate with a colon. The first field of each entry **must** be the name(s) the printer is known by, where these names are separated by '|' characters, and the field begins at the left margin without a leading colon. Every system **must** have one, and only one printer that uses lp as one of its aliases. In a single printer environment, you must include lp in the name field. Here is a sample printcap entry for a printer attached to a serial port, and then proceed with further explanations.

```
# printcap entry for DecWriter on serial port
0|lp|DEC|decwriter|LA-180 DecWriter III:\
	:lp=/dev/ttya:sd=/var/spool/lpd:br#1200:fs#06020:fc#0300:\
	:tr=\f:of=/usr/lib/lpf:lf=/var/adm/lpd.errs:
```

You can learn several general things from this example. First, notice that comments are allowed if a line begins with a '#'. Next something not so obvious to see, an entry is continued onto another line with the '\' character followed, without blank space, by a carriage return. This is a **must**. Blank space accidently typed at the end of a printcap line will cause severe problems. Notice that the continuation lines of this example are tabbed over from the left margin; all continuation lines in an entry **must** begin with blank space, normally a tab character. When entries do continue onto second and subsequent lines, a colon **must** appear at the end of one line and the beginning of the next.

Now, consider the fields within each entry. The printcap(5) manual page gives a table of all the capabilities available. You will probably need just the ones shown here to run most line printers from a serial port.

As mentioned above, the first field of each entry gives the names the printer is known by. One of the names must be lp; on a machine with more than one printer, one printer must use lp as one of its names — but only one printer.

Notice that each subsequent field is introduced by a two character code. Numeric capabilities take the form: *character_code#number_value*; for example, br#1200. String capabilities take the form: *character_code=sequence*; for example, lf=/var/adm/lpd-errs. The capabilities shown in this entry are:

lp — Specifies the file name to be opened for output. In this case it is the first serial port /dev/ttya. The default is /dev/lp.

sd

Specifies the name of a spooling directory. Make sure the directory exists with proper permissions before attempting to run the printer. When you install the operating system, it includes a correct spool directory: /var/spool/lpd.

br

Sets the baud rate for the tty line to the value given here.

fs

Sets flag bits. See tty(4) for an explanation of these sg_flags. The example here, fs#06020, is a good one to use on a serial printer. The 6000 value expands tabs, that is, sends blank spaces to a printer that cannot use a tab character. The 20 value puts out both a carriage return and a line-feed at the end of each line of print. Use the fc capability to clear flag bits.

fc

Clears flag bits. Similar to fs, but turns off bits instead of turning them on. The fc#0300 turns off the even and odd parity bits, resulting in space parity.

tr

A trailer string, to be sent to the printer at the end of a series of print jobs. In this example the trailer is a form feed.

of

This is the name of an output filtering program, a standard post-processor for nroff in this example. It is supposed to make underlined text come out properly on line printers. It may or may not be appropriate on your printer. Try running your printer without it. If the format looks fine, you probably do not need the filter.

lf

This is where spooler errors are logged. It can be created anywhere, but must exist with write permissions before errors can be sent to it. To assure that the spooler daemon can write on the error log file, become superuser, create the log file, and do a chmod 666 on it.

xs

Sets local mode bits. See tty(4) for a description of the local mode bits. For example, to set the LLITOUT bit, use xs#040. Use the xc capability to clear local mode bits.

### Other File Modifications

After your printcap file has been edited for your printer(s), modify a few more files.

□ Check to make sure the proper permissions and ownerships exist on the files /usr/lib/lpd, /usr/ucb/lpr, and on the directory, /var/spool/lpd. Note that you may have given a different name to your /var/spool/lpd directory, and that you will have one spooling directory with a unique name for each printer accessible on your system. For the files named above type ls -lg to check permissions, ownership and group. To check the same things on the spooling directory, type ls

-lgd. In addition, check the files in the spooling directory with ls -lg. The permissions, ownership, and group should match those shown below:

```
# ls -lg /usr/lib/lpd /usr/ucb/lpr
-rws--s--x  1 root    daemon   53248 Oct 14 09:19 /usr/lib/lpd
-rws--s--x  1 root    daemon   30720 Oct 14 09:19 /usr/ucb/lpr
# ls -lgd /var/spool/lpd
drwxrwx---  2 daemon  daemon     512 Nov 09 11:00 /var/spool/lpd
# ls -lg /var/spool/lpd
-rw-r--r--  1 root     daemon         22 Mar  1 18:25 lock
-rw-rw-r--  1 root     daemon         29 Mar  1 17:28 status
# ls -lg /dev/ttya
crw-rw----  1 daemon   daemon  12,     0 Oct 21 11:57 /dev/ttya
```

□ Make sure that init(8) does not create a login process on the port you are using for your printer. To do this, edit the /etc/ttytab file, making sure that the "status" field is "off" for the tty pot that you printer is attached to. The example printcap attaches a serial printer to the first CPU serial port, ttya, /etc/ttytab should have an entry like:

```
    ttya    "/usr/etc/getty std.9600"   925 off
```

If it was necessary to edit the /etc/ttytab file, you must notify init to make the system aware of the changes made. That is done by the following kill command while you are superuser:

```
    # kill -HUP 1
```

Now you can send something to the printer with the lpr command. If the printer does not work now, but did when you sent it output with the cat command to /dev/tty*, make sure the permissions and characteristics of all the lpr–related files are correct. They must be as described here. Make sure that your output filter (if you have one) is executable and that it is located where /etc/printcap says it is. Finally, check all the fields and the syntax in the /etc/printcap file.

## Hooking Up a Printer to a VPC-2200 Multibus Board

In this configuration, you hook-up your printer to a Multibus printer controller. Sun currently provides support in the standard software distribution for a single Systech VPC-2200 board per workstation. This section explains the installation of that board and a printer.

If you want to use a different printer controller board you will have to write your own device driver for the kernel. This requires some expertise; the procedure is described in *Writing Device Drivers*. Due to the difficulty of writing a driver, this course is not recommended.

The remaining discussion deals with the Systech VPC-2200 board.

First install the board in the card cage of your Sun workstation and connect the cable. If necessary, configure the new board into the system kernel. Finally modify the necessary files to enable the workstation to queue jobs and send them to the printer.

These steps are discussed in the sections below.

### Card Cage Installation And Cable Hook-up

Read the manufacturer's manual for the VPC-2200. Check all recommendations and settings carefully. When installing a printer controller board, you may place it in any slot that does not share a P2 section with the Sun-2 CPU or Sun-2 Memory boards. A basic ribbon cable can connect the board to the printer.

You should note the section *Systech VPC-2200 Versatec Printer/Plotter Controller* in the hardware manual for your workstation. There you will find a detailed description of how to install and test the Systech board.

Follow the self test instructions for the board once it is installed. (Note that some printers, such as the Imagen, will not work with self test.) Remember to set switches back to operate mode after self test is complete. If you have trouble with self test, double check all the recommended switch settings, check the cabling from the controller to the printer, and make sure the board is connected snugly in the card cage. Consult the manufacturer's manual for any recommended procedures. Call the manufacturer if the board still seems flaky.

### Kernel Modification

You must be superuser to make kernel modifications.

Here is a brief outline of the steps for building a new kernel after installing a Systech VPC-2200 printer interface board. For a general discussion of software modifications after adding a new board, see the section *Adding a Board to Your System* earlier in this chapter. (If you use a different board and write your own device driver, the process is slightly different; see the *Writing Device Drivers.*

Follow the steps shown here to reconfigure the kernel after the installation of a Systech printer controller board. These steps explain the process on a system named `GRENDEL`:

1. Change to `/usr/share/sys/sun`*[2,3,4]*`/conf` directory and make a copy of the current kernel configuration file to keep until the new one is installed and running.

```
# cd /usr/share/sys/sun[2,3,4]/conf
# cp GRENDEL old.GRENDEL
```

   If you have not built your own kernel, you will have to base your kernel configuration on the GENERIC kernel configuration file and make preparations for building your new kernel:

```
# cd /usr/share/sys/sun[2,3,4]/conf
# cp GENERIC GRENDEL
# chmod +w GRENDEL
# mkdir ../GRENDEL
```

2. Edit the kernel configuration file, adding an entry for the Systech board you have installed into the card cage. Look in *Section 4* of the *SunOS Reference Manual* for a description of this device. The entry you make in the kernel configuration file looks like this:

```
device   vpc0 at vme16d16 ? csr 0x480 priority 2 vector vpcintr 0x80
```

3. Run `/usr/etc/config` on the new kernel configuration file.

```
    # /usr/etc/config GRENDEL
```

4. Build the new kernel.

```
    # cd ../GRENDEL
    # make
```

5. Now install the new kernel and try it out.

   First move the original working kernel to another (safe) place, then copy the new kernel to the place of the original, and finally boot up the system with this new kernel.

```
    # mv /vmunix /vmunix.old
    # mv vmunix /vmunix
    # /etc/halt
```

   The system goes through the halt sequence, then the monitor displays its prompt, at which point you can boot the system by typing **b**.

   The system boots up multiuser, and then you can try things out.

6. If the new kernel does not seem to be functioning properly, halt the system and boot from the original kernel. Then reinstall the original kernel. Once you are booted up on the original, you can go about trying to fix the faulty kernel.

Remember to remove the `old.GRENDEL` kernel configuration file you created as a backup.

For more information about kernel building, see Chapters 9 and 16.

### File Modification

You must be superuser to make the following file modifications.

### Using MAKEDEV To Create Special Files

The kernel communicates with the printer through special files in the directory `/dev`. When a Systech board is added to the system, these new entries must be made in the `/dev` directory. This is relatively easy to do with a shell script called `MAKEDEV` (8), located in `/dev`. `MAKEDEV` takes the argument `vpc0`, and automatically installs the files `/dev/vp0` and `/dev/lp0` (`vp0` is the special device file for the Versatec interface; `lp0` is the special device file for the Centronics/Dataproducts interface.)

```
    # cd /dev
    # MAKEDEV vpc0
```

is what you type. If the system has had a printer in the past, there may be a

/dev/vp0, or even a /dev/lp0, existing already. If they exist, you **should** remove these files before running MAKEDEV. An error message from MAKEDEV, mknod: File exists, will be returned if you try to make a file on an existing one.

### Editing the /etc/printcap File

Refer to the earlier section of the same title for this information.

### Other File Modifications

After your printcap file has been edited for your printer(s), you will need to adjust files.

□ Check to make sure the proper permissions and ownerships exist on the files /usr/lib/lpd, /usr/ucb/lpr, and on the directory, /var/spool/lpd. Note that you may have given a different name to your /var/spool/lpd directory, and that you will have one spooling directory with a unique name for each printer accessible on your system. For the files named above type ls -lg to check permissions, ownership and group. To check the same things on the spooling directory, type ls -lgd. In addition, check the files in the spooling directory with ls -lg. The permissions, ownership, and group should match those shown below, for example:

```
# ls -lg /usr/lib/lpd /usr/ucb/lpr
-rws--s--x  1 root    daemon   53248 Oct 14 09:19 /usr/lib/lpd
-rws--s--x  1 root    daemon   30720 Oct 14 09:19 /usr/ucb/lpr
# ls -lgd /var/spool/lpd
drwxrwx---  2 daemon  daemon     512 Nov 09 11:00 /var/spool/lpd
# ls -lg /var/spool/lpd
-rw-r--r--  1 root     daemon         22 Mar  1 18:25 lock
-rw-rw-r--  1 root     daemon         29 Mar  1 17:28 status
```

Now you can send something to the printer with the lpr command. If the printer does not work now, make sure the permissions and characteristics of all the lpr–related files are correct. They must be as described here. Make sure that your output filter (if you have one) is executable and that it is located where /etc/printcap says it is. Finally, check all the fields and the syntax in the /etc/printcap file.

Note that printers may come with their own software and installation instructions. These are similar to the above, but not identical. Follow the manufacturer's instructions.

## Printing on Remote Machines

Remote spooling via the network is handled with two spooling queues, one on the local machine and one on the remote machine with the printer connected. 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 remote (or receiving) machine must have an entry for the local (or sending) machine in both its /etc/hosts and its /etc/hosts.equiv files to enable the transfer.

The local machine must know about the remote machine by having an entry for it (remote) in the local's /etc/hosts file.

Machines that use the printer on a remote machine should have an empty field for the lp capability in the /etc/printcap file. For example, the following printcap entry would send output to the printer named versatec on the remote machine venus.

```
    lp|1|versatec:\
        :lp=:rm=venus:rp=versatec:sd=/var/spool/vpd:
```

- The name field in this example happens to match the name of the printer on the remote machine. It need not necessarily.
- lp – is an empty field. Don't forget the equal sign.
- rm – is the name of the remote machine to print on. As mentioned, this name must appear in the /etc/hosts database on the local machine.
- rp – indicates the name of the printer on the remote machine is 'versatec'; the default for this capability is the 'lp' printer on remote.
- sd – specifies /var/spool/vpd is the spooling directory instead of the default value of /var/spool/lpd.

## Output Filters

The printcap examples above show various types of output filters. This section gives more details on their uses and specifications.

Filters are used to handle device dependencies and to perform accounting functions. The output filter **of** is used to filter 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 names, identifying the beginning and ending of jobs, etc. The other filters, such as if, (if specified) are started for each job printed and perform accounting if there is an af entry. The af entry designates the file where if puts its accounting informations — see the second example below. If entries for both of and one of the other filters are specified, the output filter is used only to print the banner page; it is then stopped to allow other filters access to the printer. An example of a printer that requires output filters is the Benson-Varian.

```
va|varian|Benson-Varian:\
    :lp=/dev/va0:sd=/var/spool/vad:of=/usr/lib/vpf:\
    :tf=/usr/lib/rvcat:mx#2000:pl#66:tr=\f:
```

The tf filter (invoked with lpr -t) takes a troff output file and converts it to Versatec output. It is used by vtroff (1). Note that the page length is set to 66 lines by the pl entry for 8.5" by 11" fan-fold paper. To enable accounting, the varian entry would be augmented with an af file as shown below.

```
va|varian|Benson-Varian:\
    :lp=/dev/va0:sd=/var/spool/vad:of=/usr/lib/vpf:\
    :if=/usr/lib/vpf:tf=/usr/lib/rvcat:af=/var/adm/vaacct:\
    :mx#2000:pl#58:tr=\f:
```

## Output Filter Specifications

Sun software provides several filters that are listed as `?f` under CAPABILITIES in `printcap(5)`. For many devices or accounting methods, it is probably necessary to create a new filter.

Filters are spawned by `lpd` with their standard input the data to be printed, and standard output the printer. The standard error is attached to the `lf` file for logging errors. A filter must return a zero exit code if there were no errors, 1 if the job should be reprinted, and 2 if the job should be thrown away. When `lprm` sends a kill signal to the `lpd` process controlling printing, it sends a SIGINT signal to all filters and descendents of filters. This signal can be trapped by filters that need to perform cleanup operations such as deleting temporary files.

Arguments passed to a filter depend on its type. The `of` filter is called with the following arguments.

```
ofilter -wwidth -llength
```

The *width* and *length* values come from the `pw` and `pl` entries in the `printcap` database. The `if` filter is passed the following parameters.

```
filter [ -c ] -wwidth -llength -iindent -n login -h host accounting_file
```

The –c flag is optional, and only supplied when control characters are to be passed uninterpreted to the printer (when the `–l` option of `lpr` is used to print the file). The –w and `–l` parameters are the same as for the `of` filter. The –n and –h parameters specify the login name and host name of the job owner. The last argument is the name of the accounting file from `printcap`.

All other filters are called with the following arguments:

```
filter -xwidth -ylength -n login -h host accounting_file
```

The –x and –y options specify the horizontal and vertical page size in pixels (from the `px` and `py` entries in the printcap file). The rest of the arguments are the same as for the `if` filter.

# Part Three: Network and Communications Administration

This part contains important procedures and theoretical information for administering local and wide area networks on the Sun workstation. The information in this part is applicable to any Sun workstation that is on a network.

## Who Should Read This Part

If your machine is on a network, you should familiarize yourself with the procedures in a chapter that apply specifically to your system, be it a server or client, including a networked standalone. (Chapter 3 defines these terms from a networking perspective.) If you have a standalone system that is not attached to a network, Part Three does not apply to you.

## What Is in This Part

Part Three discusses the following:

□ What network protocols are, steps you need to perform to get a network up and running, and suggestions for troubleshooting network-related problems.

□ How to set up and administer servers and clients to use the Sun Network File System.

□ How to administer the Sun Yellow Pages services.

□ How to administer electronic mail on your system, and how to set up the `uucp` wide area network.

# 12

# The SunOS Network Environment

# 12

# The SunOS Network Environment

This chapter discusses networking within the SunOS environment. It explains how information is sent and received across a network, how to maintain the network using SunOS commands and files, and how to expand an existing network. Topics covered are:

- □ Understanding the ISO communications protocol model and how the Sun networking software has implemented it
- □ Understanding and maintaining local area network hardware
- □ Setting up and administering network software
- □ Expanding an existing network
- □ Troubleshooting network problems (for advanced administrators)

The information presented in this chapter assumes that you have a basic knowledge of networking concepts as they apply to Sun hardware and software. Refer to Chapter 3, "Introducing Networks," if you need an overview of networking on your Sun computer.

## 12.1. SunOS Networking Protocols

Networks accept information from a sender, transmit that information along a communications line, and finally deliver the information in a form that is intelligent to the recipient. In order to accomplish these tasks, different types of networking software need to interact with one another. Thus, network designers developed a family of communications *protocols*. These protocols are a set of formal rules explaining how software and hardware should interact within the network in order to transmit information.

This section briefly explains a model for communications protocols developed by the International Standardization Organization (ISO). It also explains how SunOS software and hardware use this model as a basis for network communication.

### The ISO Model for Communications Protocols

The International Standardization Organization is a body that develops standards for various areas of computing. The ISO developed their communications protocol model in an effort provide a consistent structure for people who were designing protocol families. This model consists of seven parts, or *layers*, each of which may consist of a type of protocol and an addressing method. The protocol for each level specifies the functions available at that level, as shown in Table 12-1.

Table 12-1 *The ISO Protocol Model*

| *ISO Layer* | *Function* |
|---|---|
| **Application** | Network services |
| **Presentation** | Data presentation, such as message formatting |
| **Session** | Makes the communications link between sender and receiver |
| **Transport** | Connects processes at different locations to each other |
| **Network** | Provides the address of a machine on the network |
| **Data Link** | Groups bits together for transmission |
| **Physical** | Actual hardware comprising a network |

Each protocol layer performs functions for the protocol layer above it. The addressing method helps a particular layer to pass information and call other programs at other layers.

## SunOS Protocols

The ISO interpretation of the protocol layers is loose enough so that designers have some leeway in how they implement the protocols. For example, some Sun applications bypass the Presentation and Session layers to interact with the Transport layer.

Table 13-2 shows how Sun has implemented the ISO layered model.

Table 12-2 *Communications Protocol Layers in SunOS*

| *ISO Layer* | *Sun RPC Services* | *Sun Non-RPC Services* |
|---|---|---|
| **Application** | NFS, YP, `mount`, etc. | `rlogin`,`rcp`,`tftp`, etc. |
| | | \| \| |
| **Presentation** | XDR | \| \| *(bypass)* |
| | | \| \| |
| **Session** | Remote procedure calls (RPCs) | v v |
| **Transport** | TCP, UDP protocols | TCP, UDP protocols |
| **Network** | IP protocol | IP protocol |
| **Data Link** | Ethernet (or other) controller | Ethernet (or other) controller |
| **Physical** | Ethernet (or other) cable | Ethernet (or other) cable |

When information is generated or received, it conceptually travels through each layer of the protocol until it arrives at its proper destination. The next subsections explain each layer in detail and show how a message or request travels through the protocol layers until it goes out over the transmission media.

### Level 7: Application Layer

In SunOS, this layer contains network applications, or *services*. Some network services at this level, such as NFS, YP, the `mount` daemon, lock manager, and `rstatd` program, are known as RPC services. These services have an identifying, fixed program number as their addressing method.

Other services at the Application layer, such as `tftp`, `rlogin`, `rcp`, `telnet`, `ftp`, and `rsh`, are known as non-RPC services. Non-RPC services have an identifying *port number*, which is their addressing method.

As an example, assume you issue a `mount` command from a client machine. The request begins at Layer 7, then travels through the succeeding layers of the network protocol model. Software at each layer interacts with the request, formatting it and adding information to it until it is ready to go out over the communications media. (Chapter 13, "The Sun Network File System," contains more information about mounting file systems through NFS.)

### Level 6: Presentation Layer

The standard protocol XDR (eXternal Data Representation) exists at this layer. XDR translates data so that different kinds of hardware can communicate with each other. The RPC services, such as NFS, YP, and `mount` use XDR. However, non-RPC services, such as `tftp` and `telnet` bypass this layer.

Continuing the example from the previous level, a mount request issued at Layer 7 is passed to this level, where it is translated into XDR format.

### Level 5: Session Layer

This layer contains *remote procedure calls*, or RPC. RPC is a standard mechanism for communication between two remote programs. RPC services such as NFS communicate with this layer, though non-RPC services bypass it.

You can use the program `rpcinfo` to get a list of registered RPC programs on a particular host, and to perform other RPC-related functions, such as making RPC calls to lower layer protocols on other machines. Refer to the `rpcinfo`(8c) man page for more information.

A program called `portmap`, or the portmapper, translates the fixed program number of an RPC service to a variable port number, which is then passed to the following lower level.

The manual *Network Programming* contains detailed information about RPCs, XDR, and RPC programming.

### Level 4: Transport Layer

The Sun local area network uses two sets of protocols at the Transport layer: TCP (transmission control protocol) and UDP (user datagram protocol). The addressing method used at this level is the port number. Network services that skip the above two protocol layers communicate directly with this layer. The nature of the request handled by the service determines whether TCP or UDP is used.

A request reaches the Transport layer as a sequence of characters. TCP handles virtual circuit (connection-oriented) service while UDP provides datagram (connectionless) service. For example, print requests are handled by the TCP protocol while NFS requests are handled by UDP. If a message is too long, TCP or UDP breaks it down into segments that can be handled over the network.

The file `/etc/inetd.conf` describes both RPC and non-RPC services. If you administer a server, you may have to update this file if you add various services. Refer to the `inetd.conf`(5) man page for more information.

The file `/etc/services` lists non-RPC services, their port numbers, and associated protocols. It is described in the `services`(5) man page. Usually you do not have to maintain this file.

Continuing the mount example, when the `mount` request reaches the Transport layer, it is handled by the UDP protocol, which resembles a a datagram.

### Level 3: Network Layer

On a Sun network, this layer typically contains the Internet (IP) protocol; software at the Internet layer determines the route over the network that a message must take to arrive at the destination machine.

The addressing method used at this level is the IP address. The IP addresses of all machines and networks known to a server are maintained in the `/etc/hosts` and `/etc/networks` files. If you have a server, you will have to maintain these files. They are described in more detail later in this chapter and in the `hosts`(5) and `networks`(5) man pages.

Because the address of the recipient is now known, the network level protocols can now connect a process on one machine to a process on another, regardless of their location. In the mount request example, when the datagram containing the request reaches the Network layer, the IP protocol determines the route the message must take on the network to go to the recipient server.

### Level 2: Data Link Layer

On this level, information is prepared for transmission. The Ethernet (or similar) software organizes the bits comprising the message into groups of appropriate size for transmission and attaches an identifying header to each group. Note that the communications controller with an Ethernet (or similar) chip and buffer is involved at this level.

The Ethernet address is the addressing method used at this level. Each machine on the network has its own Ethernet address. Ethernet can only send a message to a host on its own local area network. If the recipient machine is not on the local network, the message is sent to a router on that local network, which then forwards the message to another network.

When a datagram containing a mount request reaches the Data Link layer, software at this level formats the request for transmission and attach a header containing the Ethernet address of the destination server.

### Level 1: Physical Layer

This is the hardware level of the protocol model. On Sun systems, it consists of the connector to the network transmission media and the cables. In the example of the mount request, the controller at the Data Link level sends the now-formatted mount request over the attached cable, where it is received by the communications controller on the recipient machine or router.

SunOS enables programs to treat the network in a fashion similar to files through mechanisms known as *sockets*. Sockets allow programs to communicate with each other remotely or on a local system, in a similar fashion to reading or writing to files.

### How Data Is Grouped for Transmission

Networking software in the upper layers of the protocol model, groups data into either *datagrams* or *virtual circuits*, depending upon the particular protocol. Virtual circuits appear to be transmitted in a character-by-character fashion, similar to the way a workstation keyboard sends a byte of data representing a character to the CPU each time you press a key. A transmission consists of a starting point, which opens the connection, the entire message in byte order, and an ending point, which closes the connection. Conversely, datagrams are groups of information transmitted as a unit by the upper layer protocols.

Datagrams and virtual circuits exist at the Transport layer and all layers above it. At the data link level, both datagrams and virtual circuits are encapsulated into a format called a *packet*. A packet contains a source address, the message itself, and a destination address, all transmitted over the network media in one or more units. Long transmissions often are made up of several packets, in which case each packet also contains information about the packet to follow it. Ethernet and thin-wire Ethernet are called *packet switching networks* because their communications channels are only occupied for the duration of the transmission of a packet. The telephone network is an example of a *circuit switching network*.

Note that it is not crucial for you to know about packets, datagrams, and virtual circuits in detail. These terms have been presented here because you will see them in network administration files, statistics displayed by various commands, and in error messages displayed on the system console. If you require more information about these concepts, refer to the `intro`(2) man page in the *SunOS Reference Manual* and the *Network Programming* manual.

## 12.2. Understanding and Debugging Network Hardware

Local area networks facilitate inter-computer communications in a limited area—within a building or between neighboring buildings. This section briefly describes the hardware that comprises a basic local area network, using the Ethernet technology as an example. If you are using a different technology, some of this information may not apply. This section does not describe how to physically assemble the hardware in any great detail. If you have this responsibility, refer to the hardware manuals that came with your equipment.

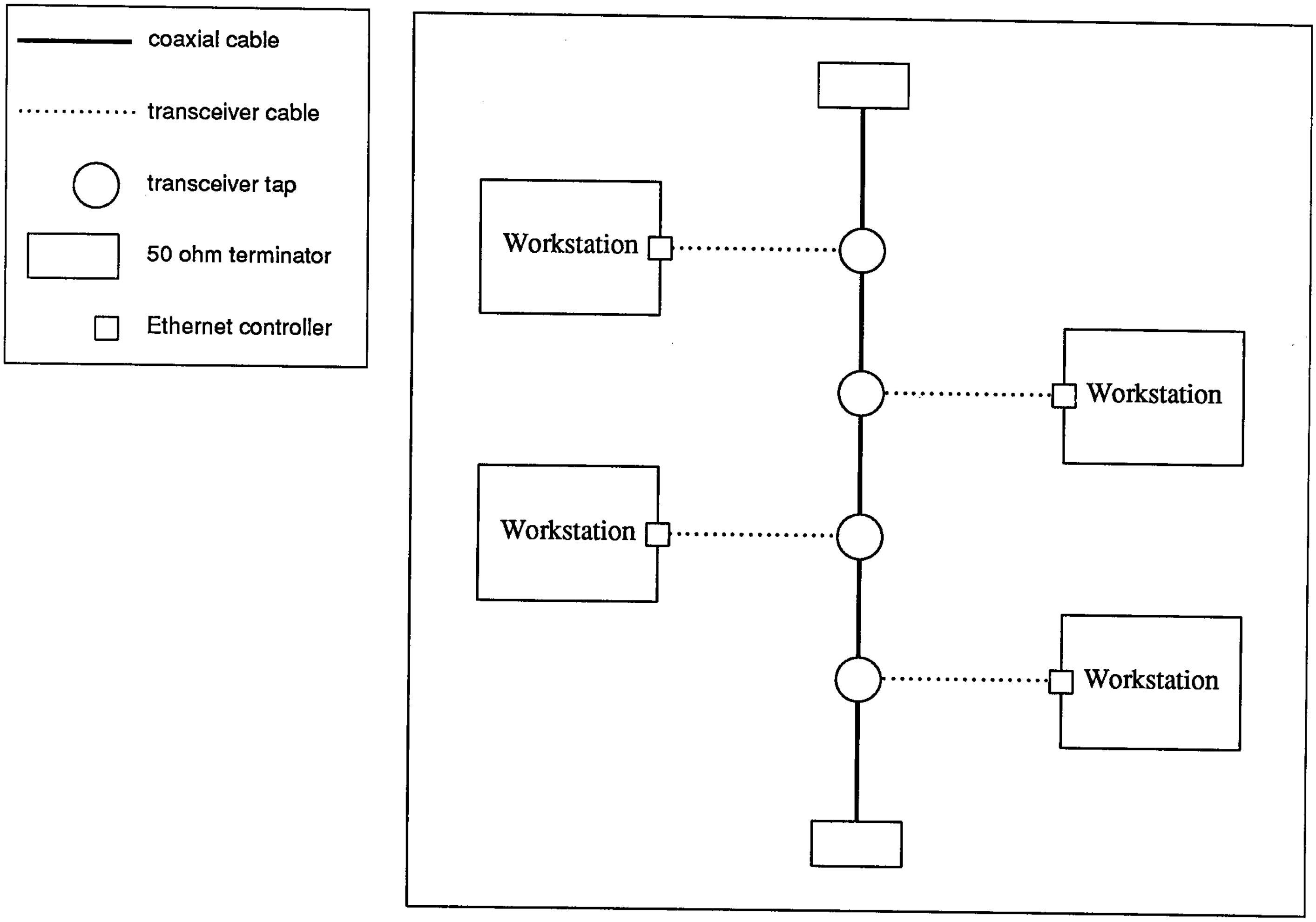
Individual workstations and other devices are plugged into the Ethernet's cable system. The Ethernet controller inside each machine has its own, unique Ethernet address. This is displayed as soon as you attach the system to cable and power it up. You can move the machine around and plug it into any convenient outlet on a single Ethernet cable. You can attach different types of equipment to an Ethernet cable, but they cannot always communicate with one another. For example, VMS systems using DECNET and Suns can be attached to the same Ethernet cable. But they cannot communicate with one another without special software on one or both of the systems.

The diagram on the following page shows a typical Ethernet setup.

The basic parts of a typical Ethernet system, as shown in the diagram, include the following:

- □ The Ethernet controller in the machine.
- □ The *transceiver* cable that connects the controller board to the transceiver taps on the coaxial cable. Optionally, you can connect the transceiver cable to an Ethernet multiplexor box, instead of to the transceiver on the coaxial cable.
- □ An optional Ethernet multiplexor box. A *multiplexor box* is equivalent to several transceivers stacked end to end and connected by coaxial cable. You can run up to eight transceiver cables from a single multiplexor box. The multiplexor box allows you to attach several devices to a single transceiver tap into your coaxial cable.
- □ The coaxial cable forms the backbone of the Ethernet system. All workstations or other devices are attached to it with transceiver cable and taps. The maximum operative length of joined coaxial cables in an Ethernet system is 1500 meters.
- □ Both ends of a coaxial cable are terminated by terminators with insulated outside covers.

Figure 12-1 *A Local Physical Network*



### File Transfer over the Ethernet

Here is a breakdown of file transfer, from the hardware perspective, over the Ethernet local area network.

1. A workstation user typically uses a command to specify that a file be transferred between a sending and receiving machine.
2. The command makes requests to the operating system. SunOS then copies the data through a *virtual circuit* or as a series of *datagrams*.
3. The virtual circuit or datagram software breaks the character stream into packets for transmission.
4. The packets are passed to the Ethernet driver software.
5. The Ethernet device driver copies the packet into a packet buffer and tells the controller to transmit it.
6. The controller waits until the coaxial cable is not in use and then transmits the packet.

7. The Ethernet transceiver receives the packet's bit stream and injects it into the coaxial cable.

8. The receiving machine recognizes its address in the packets, and reverses the above procedure: bits are received by the transceiver, fed to the controller, passed to software that reassembles the packets, maps the characters, and stores the data.

### Hardware Devices for Expanding the Local Network

Below is a description of the hardware necessary to expand your local area network to an internetwork consisting of two or more local networks. It is slanted towards Ethernets, but the general concepts apply to other local area network technologies, as well.

- **Repeater:** This is a device used at the physical layer of the network that copies each bit of a packet from one segment of a network to another. In Ethernet terminology, a single length of coaxial cable is considered a physical segment. Therefore, a physical Ethernet local area network consists of several segments connected by repeaters. It is possible to implement a repeater in two halves and connect them, possibly with a pair of fibers. This gives wider geographical coverage. The Ethernet allows 1000 meters of fiber inside the repeaters.

- **Bridge:** This is a device used at the data link level of the network protocol model. It selectively copies packets from one segment of the network to another segment of the same type. This is reasonable to do, in theory, on an Ethernet, because the first few bytes of an Ethernet packet contain its destination address. A bridge that is watching all traffic on an Ethernet can determine which hosts are sending which packets. Therefore, it will know which packets to copy from one segment to another.

  Bridges differ from repeaters in several ways. Because copying done by bridges is selective, this reduces traffic on each section of the network. Since bridges copy whole packets, the geographical or timing constraints of the basic physical network can be extended.

  You can split a bridge as you would a repeater. Like repeaters, bridges copy raw packets, a scheme which works for all higher lever protocols. Since bridges copy whole packets, In theory, connections like this are "invisible" to the software on all the machines on the network.

- **Router** (sometimes called a gateway): This is a device that forwards packets of a particular protocol family, for example, IP, from one logical network to another. A logical network makes sense only to a particular protocol, and has a single network number. Usually there is a one-to-one mapping between physical network and logical network, but things get blurred by subnets (explained later in this chapter) and bridges. The internetwork is a collection of logical networks (all speaking the same protocol) connected via routers.

  The router may forward packets between different physical types of networks, for example, from an Ethernet to a ring local area network, or from the ARPANET to a phone line. It can also forward packets between two logical networks of the same type, for example, between two Ethernets on different floors of a building.

During forwarding, a router looks inside the packet to find the destination address, then consults its routing table, which is normally kept up to date by having routers talk to each other via some routing protocol. Note that it is possible to have a multilingual router—a single device that forwards packets for several protocol types, for example, IP, ISO, or XNS.

- **Gateway** (sometimes called a relay or forwarder): This is a device and its associated software that enable networks using different protocols to communicate with each other. Because it translates protocols existing at all layers of the protocol model, you can use gateways to connect networks that differ on all layers from each other.

## Solving Ethernet Problems

From time to time, most networks will have problems. This section discusses some techniques for handling them.

Always check your network connections first. On networks such as the Ethernet a loose cable tap or misplaced transceiver cable can result in deteriorated service. If you believe you have a faulty coaxial Ethernet cable, test it to make sure it delivers 50 ohms — see *Hardware Checks* below.

Below are some suggested corrective actions. After each step, the network should be tested again.

### Hardware Checks

Here are some actions you can take to diagnose and fix hardware-related problems.

1. With its host system powered on, after awhile, each transceiver should feel slightly warm to the touch. A cold transceiver probably is not receiving power. This could indicate one of several things: a loose connection on either end of the transceiver cable, a loose connection of the internal Ethernet cable to the Ethernet board (on Multibus or second Ethernet board only), or a faulty cable, transceiver (less likely), or Ethernet board (even less likely).

2. Check the solidity of all Ethernet coaxial and transceiver cable connections. If your machine has a second Ethernet board or is a Sun-2 with Multibus, make sure that the internal connector (from the back panel) is plugged into the Ethernet board.

3. The network must be terminated on both ends. Ensure that it is by unscrewing one of the terminators and using an ohm meter to test resistance across the coaxial connector where you just unscrewed the terminator (use the pin "inside" the N-connector for signal, and the housing for ground). You should measure about 50 ohms. If you get something other than 50 ohms, your cable may be damaged. This check is particularly pertinent if you use a clamp-on ("vampire clamp") type of transceiver; they tend to short-circuit the Ethernet coaxial cable. The threaded connectors on the type of transceivers supplied by Sun can also develop shorts at the connectors.

4. Remove one of the terminators and try operating the network. You should get error messages on every machine, something like:

```
ie0: no carrier
ie0: Ethernet jammed
```

If a machine just continues to give its previous error

```
Connection timed out
```

or

```
no server
```

then it may not be connected correctly to its transceiver. As in Check 3 above, the problem could be loose connections or a faulty connector, cable, transceiver, or Ethernet board.

## 12.3. Setting Up Network Software

This section explains how to set up and maintain network software, including:

- Obtaining an Internet number
- Setting up an administrative domain
- Modifying network-related files
- Connecting two local networks to form an internetwork.
- Dividing an Internet network into subnets

### Obtaining an Internet Number

The Sun networking software uses Internet protocols for sending information across a network. Every site needs its own unique Internet number, which you assign to the server during the `suninstall` process. Even if your site does not plan to join the Internet, you must obtain an Internet address from the the Network Information Center (NIC) at SRI in Menlo Park, California. This organization handles all assignments of network numbers and autonomous system numbers (AS) for the Internet.

To obtain your Internet address, you must use the following questionnaire provided by the Hostmaster at SRI-NIC. When you complete it, send it via electronic mail to HOSTMASTER@SRI-NIC.ARPA, or, if electronic mail is not available to you, send it to:

DDN Network Information Center
SRI International
Room EJ217
333 Ravenswood Avenue
Menlo Park, CA 94025

1. If the network will be connected to the DARPA Internet or the DDN Internet, you must provide the name of the sponsoring organization, and the name, title, mailing address, phone number, net mailbox, and NIC Handle (if any) of the contact person at that organization. This is the contact point for

administrative and policy questions about the authorization for this network to join the DARPA Internet or the DDN Internet.

**Note:** If the network will NOT be connected to either the DARPA Internet or the DDN Internet, then you do not need to provide this information.

For example, your response may look like the following:

Sponsor

| | |
|---|---|
| Organization | DARPA |
| Name | Dr. Robert E. Kahn |
| Title | Director, IPTO |
| Mail Address | DARPA<br>1400 Wilson Bl.<br>Arlington, VA. 22209 |
| Phone Number | (202) 694-5922 |
| Net Mailbox | Kahn@ISI.EDU |
| NIC Handle | REK2 |

2. The name, title, mailing address, phone number, and organization of the administrative head of the organization. This is the contact point for administrative and policy questions about the network. In the case of a research project this should be the Principal Investigator. The online mailbox and NIC Handle (if any) of this person should also be included.

   Here is a sample response.

   Administrator

| | |
|---|---|
| Organization | USC/Information Sciences Institute |
| Name | Keith Uncapher |
| Title | Executive Director |
| Mail Address | USC/ISI<br>4676 Admiralty Way, Suite 1001<br>Marina del Rey, CA. 90292-6695 |
| Phone Number | (213) 822-1511 |
| Net Mailbox | Uncapher@ISI.EDU |
| NIC Handle | KU |

3. The name, title, mailing address, phone number, and organization of the technical contact. The online mailbox and NIC Handle (if any) of the technical contact should also be included. This is the contact point for problems with the network and for updating information about the network. Also, the technical contact may be responsible for hosts attached to this network. Here is a sample response.

   Technical Contact

| | |
|---|---|
| Organization | USC/Information Sciences Institute |
| Name | Craig Milo Rogers |
| Title | Researcher |
| Mail Address | USC/ISI<br>4676 Admiralty Way, Suite 1001<br>Marina del Rey, CA. 90292-6695 |
| Phone Number | (213) 822-1511 |

| | |
|---|---|
| Net Mailbox | Rogers@ISI.EDU |
| NIC Handle | CMR |

4. The short name of the network (up to 12 characters). This name will be used in tables and lists associating networks and addresses, for example:

   "ALPHA-BETA"

5. The long name of the network (up to 20 characters). This name should be descriptive of the network. It might be used to clarify the ownership, location, or purpose of the network, for example:

   "Greek Alphabet Net"

6. Geographically, where is this network located?

   USC/ISI
   4676 Admiralty Way
   Marina del Rey, CA. 90292-6695

7. A citation to a document that describes the network. This should identify a document that describes the network in a technical sense. Here is an example:

   "The Ethernet, a Local Area Network: Data Link
   Layer and Physical Layer Specification", X3T51/80-50 Xerox,
   Stamford Connecticut, October 1980.

8. Gateway information:

   If the network is to be connected to the DARPA Internet or the DDN Internet, answer questions 8a and 8b. If the network will not be connected to the DARPA Internet or the DDN Internet, answer question 8c.

a. A description of the gateway that connects the new network to the DARPA Internet or the DDN Internet, and the date it will be operational. The gateway must be either a core gateway supplied and operated by BBN, or a gateway of another Autonomous System. If this gateway is not a core gateway, then some gateway in this gateway's Autonomous System must exchange routing information with some core gateway via EGP.

   A good way to answer this question is to say "Our gateway is supplied by person or company X and does whatever their standard issue gateway does." For example, you might respond: "Our gateway is the standard issue supplied and operated by BBN, and will be installed and made operational on 1-April-88."

b. A description of the gateway machine, including

   (a) Hardware (Sun-3/60, LSI-11/23, VAX-11/750, etc. interfaces)
   (b) Addresses (what host on what net for each connected net)
   (c) Software (operating system and programming language)

   A sample response might be:

(a) hardware

Sun-4/110; Ethernet and 1822/HDH interfaces

(b) address

10.9.0.193 on ARPANET

(c) software

SunOS 4.0

c. A description of the gateway used in the system. For example, does the gateway:

forward IP datagrams?
send ICMP redirects?
implement GGP?
implement EGP?

A good way to answer this question is to say "Our gateway is supplied by person or company X and does whatever their standard issue gateway does." For example, you might respond:

Our gateway runs SunOS supplied by Sun Microsystems.

9. An estimate of the number of hosts that will be on the network

(a) initially,
(b) within one year,
(c) two years, and
(d) five years.

For example, you might respond as follows:

(a) initially = 5
(b) one year = 25
(c) two years = 50
(d) five years = 200

10. Unless a strong and convincing reason is presented, the network (if it qualifies at all) will be assigned a Class C network number. Is a Class C network number acceptable for your purposes, and if not why not? (If your site plans for more than a few local networks and over a 100 hosts, you should strongly consider subnetting, as described further in this chapter.) In most cases, you can respond:

Class C is fine.

11. Networks are characterized as being either Research, Defense, Government - Non Defense, or Commercial, and the network address space is shared between these three areas. Describe the network type, for example:

Research

12. What is the purpose of the network. For example, you might respond as follows:

> To economically connect computers used in DARPA
> sponsored research project FROB-BRAF to the DARPA Internet or the
> DDN Internet to provide communication capability with other
> similar projects at UNIV-X and CORP-Y.

## Establishing a Domain

If you are setting up machines on a network that will not be part of an existing domain, you must create a new administrative domain. This involves the following activities:

- Establishing a name for the domain
- Registering the domain with outside upper level networks
- Specifying the domain name during `suninstall` to update the appropriate SunOS programs and YP maps

### Deciding on a Domain Name

The first steps to take when you set up a domain is to determine its name and which machines are part of it. As explained in Chapter 3, domain names should correspond to administrative divisions, not necessarily to how networks are connected. Domain names are hierarchical, with components separated by periods. The top level domain is the component on the furthest right; lower level domains follow from right to left. This hierarchical system is similar to full pathnames in UNIX, except that you read pathnames from the top level root on the left, followed by subdirectories read from left to right.

Your domain full name must be unique within the world. For example, you could use "podunk.EDU" as a base name for a domain consisting of all machines at Podunk University, provided that another university has not used this name first. Upper and lowercase letters are not significant in domain names. The traditional UNIX convention is to use all lowercase; many other systems use uppercase.

You also have to add a valid top-level domain name as an extension to the domain base name. Top level domains often define the type of administrative entity they apply to. Some valid top-level domains are:

| | |
|---|---|
| `.COM` | Commercial Companies |
| `.EDU` | Educational Institutions |
| `.GOV` | Government Agencies |

For example, Podunk University's administrator would add the `.EDU` top-level domain name to their base name, resulting in the domain name "podunk.EDU."

You can give domains either visible or invisible first components, depending on whether or not the `sendmail` mail routing program is to include that component in the headers of outgoing mail. To make the domain name entirely appear in the header, begin it with a dot or a plus sign, as in +podunk.EDU or .sun.COM. Otherwise the first component of the domain name will be removed before being appended to the host name.

If your domain uses YP services, you might prefer starting the domain name with a plus sign. This is because YP stores domains as directories in `/var/yp`. If you run the `ls` command on `/var/yp`, it will not display any directory names beginning with a dot, unless you use the `-a` option. `sendmail` converts the plus sign to a dot before appending it to the host name.

As an example, machine fred at Podunk University has the fully-qualified domain name "fred.podunk.EDU," where "fred" is the first component in the domain name. The plus sign in +podunk.EDU indicates that the first component names for machines in that domain are to appear in outgoing mail. Therefore, `sendmail` prints the name fred.podunk.EDU in the headers of all mail sent by machine fred.

On the other hand, consider an example where the Spacely Sprocket Corporation has given their domain the name "sprocket.COM." Notice that the domain name given to `suninstall` does not begin with a period or plus sign. As a result, Spacely Sprocket Company can administer their engineering machines separately from marketing machines by creating invisible domains. For example, they might divide sprocket.COM into invisible domains called "eng.sprocket.COM" and "mktg.sprocket.COM." Here the first components of the domain names, eng and mktg, will not appear in outgoing mail. Host names for all machines within sprocket.COM must still be unique within the YP map `hosts.byname`, but other YP maps like the YP password map can be maintained separately. (Refer to Chapter 14 for information about YP maps.)

The following table shows how pluses and periods in domain names affect the name printed on a host machine's outgoing mail.

| *Host name* | *YP domain* | *Name on outgoing mail* |
|---|---|---|
| `stefania` | wseng.sun.COM | stefania.sun.COM |
| `stefania` | +sun.COM | stefania.sun.COM |
| `bill` | +spd.sun.COM | bill.spd.sun.COM |
| `bill` | .spd.sun.COM | bill.spd.sun.COM |

Domains are not limited to any fixed number of levels. For example, the Amalgamated Widgit Company might have a domain called "eng.Japan.Widgit.COM" for the engineering group within their Japan subsidiary, and another called "mktg.Japan.Widgit.COM" for marketing in Japan. However, machines at company headquarters might belong to the domain called "+HQ.Widigit.COM."

You can simplify administration if a YP domain (as set by the `domainname` command) can also be used by the mail transport system. By default, the sendmail mail routing program in SunOS Release 4.0 takes as its default domain name the one set by the `domainname` command in `/etc/rc.local`. This is the same name that you specify during `suninstall`. Old sendmail configuration files set up in previous releases of SunOS should also still work, if you need to explicitly set the mail domain.

A single machine with no network connections might not need YP at all, so it would be considered its own domain by itself. Small networks might only need to refer to names within a single domain. Very large networks might need to use the inter-domain name resolution feature that is enabled by the -b option to the makedbm program, and the name server described in Chapter 22.

## Registering a Domain

Before setting up your domain, you need to register it with a higher level domain. To do this, you should get in touch with the administrator of that domain. The top-level domains are administered by the Hostmaster at NIC at SRI International, whose address is given in the previous section. The NIC can answer your questions and send you the appropriate form for registering your domain.

If you want to register your domain on the UUCP, BITNET, or CSNET, you should contact the following people:

CSNET - CSNET Coordination and Information Center
(617) 873-2777
Netmail: CIC@SH.CS.NET

UUCP - Mark Horton
Netmail: registry@stargate.COM

BITNET - BITNET Network Information Center
(619) 734-1878
Netmail: info%bitnic@CUNYVM.CUNY.EDU

The administrator of each of these organizations processes applications, then sends them to the NIC, which will then send you a special form to fill out.

Once your domain is registered, you can divide it into subdomains as the need arises.

## Maintaining Network-Related Files

SunOS includes a number of administrative files that are used during network operations. If YP is running on your network, it automatically maintains certain network-related files. Others you have to maintain yourself. If you do not have YP on your network, you should take care to keep the files discussed in this next section up-to-date.

## The /etc/hosts File

The /etc/hosts file is one of two that help locate other machines on the local area network. It lists, by Internet address and name, all machines in the local domain that your server needs to recognize. (From an NFS perspective, a domain is the name selected during suninstall for a local area network.)

When you first run suninstall, it generates a default /etc/hosts file containing the Internet address of the server, plus the Internet addresses of any clients you described on the Client Form. This file also contains the localhost and loghost addresses, which are internal addresses on the machine.

Suppose you set up a server named dancer using suninstall and gave it an Internet address 192.9.200.100. Then you defined three clients for dancer--ballet, raks, and samba--on the Client Form. The resulting /etc/hosts file would look like the following:

```
# If the yellow pages is running, this file is only consulted when booting
#
# These lines added by the Suninstall Program
#
192.9.200.1         ballet
192.9.200.2         raks
192.9.200.3         samba

192.9.200.100       dancer

127.0.0.1 localhost loghost
#
# End of lines added by the Suninstall Program
#
```

The numbers in the first column are Internet numbers for each host. The next column lists the official host names. You can also add any aliases for a host in this column. An *alias* is another name by which the host can receive information, such as mail or NFS services, over the network. You can also add a comment to each entry. After the alias(es) for the host, type a pound sign and follow it with the comment text.

If you are not running YP, you must keep `/etc/hosts` up-to-date, so that your server can recognize the machines' calls for service and send them their requested file systems, if they have the proper permissions. To update `/etc/hosts` you must log in as superuser, then edit the file as you would any ASCII file, using your preferred text editor.

You will need to update `/etc/hosts` after you have done the following:

- □ Just installed your server in a domain that includes other servers besides yours. In this case, you need to add the Internet addresses of the server(s) and clients in your domain that you want your server to know about.
- □ Added a new client to a server already running SunOS 4.0, or a new router to an existing local network. You need to make an entry in `/etc/hosts` for this machine.
- □ Found out that a new client has been added to a different server on your network, and that you have to provide services for it. You need to make an entry to `/etc/hosts` file for this machine.

You can use secondary names for a machine in `/etc/hosts`. People use secondary names for machines for a number of reasons. Sometimes users want two machine names that they can use interchangeably for a system. Another reason is for specifying a machine's function. For example, suppose everyone at a site accesses a telephone database from the machine named "registry." This is a secondary name for a machine; you can change the registry from machine to machine with only minor changes. People going to that machine to get the phone list don't have to know anything about the change. This is also convenient for datehost, loghost, and mailhost.

For more information about /etc/hosts, refer to the hosts(5) man page in the *SunOS Reference Manual.*

### The /etc/ethers File

The other file used by your server to locate hosts on the network is /etc/ethers. It lists, by Ethernet address and name, all machines in the local domain that your server needs to recognize. The initial /etc/ethers file is set up by suninstall from the Ethernet addresses you provide on the Client Form. A default /etc/ethers file for the server dancer mentioned above might look like the following:

```
#
# Entries in this file should be in alphabetical order by machine name.
#
8:0:20:1:40:14  ballet
8:0:20:1:2h:7   dancer
8:0:20:1:40:15  raks
8:0:20:1:40:16  samba
```

The first column of the file contains the Ethernet address of each machine. (Recall that after a machine is physically connected to a local area network, you can find out its Ethernet address simply by powering it up.) The second column contains the official host names of each machine. It is important that these names correspond to the names in /etc/hosts.

For the same reasons that you must maintain /etc/hosts, you must keep /etc/ethers up to date if your network is not running YP. To update /etc/ethers you must log in as superuser, then edit the file, as you would any ASCII file, using your preferred text editor. For a network that is running YP, you will have to propagate YP maps that correspond to /etc/ethers, as explained in Chapter 14. Refer to the ethers(5) man page in the *SunOS Reference Manual* for additional information.

### The /etc/netgroup File

You use the /etc/netgroup file to define network-wide groups of users; suninstall does not create this file during installation. If your network does not run YP, and you want to allow access to certain directories only to specific network-wide user groups, you have to create /etc/netgroup. Below is a sample /etc/netgroup file.

```
everyone          (,,)
aswan             (raks,amina,sun) (masr,shamira,sun)
justmachines      (raks,-,sun) (ballet,-,sun) (samba,-,sun)
justpeople        (-,amina,sun) (-,stefania,sun) (-,ernest,sun)
```

A line in the netgroup file has the following overall syntax:

```
groupname member member ....
```

where *member* is another group name or what is known as a triple. A triple has the format

```
(hostname, username, domainname)
```

where *hostname* is the name of a machine, *username* is the name a person uses to log in, and *domainname* is the name of the domain in which the triple is valid. Any of the three fields within the parenthesis can be empty. You can also substitute a character other than a letter, digit, or underscore, for example, a dash (-), in a field to give it the opposite meaning. Below are four examples showing the triple syntax.

To define a group to which all users on the network belong, you simply leave all fields of the triple empty. In the following example,

```
everyone (,,)
```

everyone on the network belongs to the everyone netgroup. For example, if, in an `/etc/exports` file, you give access to a directory to netgroup everyone, then all users on the network can access this directory, not just all users on your server's clients.

To define a group to which specific users on specific machines belong, you might use a line like the following:

```
aswan    (raks,amina,sun) (masr,shamira,sun)
```

In this example, when user amina is logged in to host raks in domain sun, she can access any directories restricted to members of netgroup aswan. Another user, ernest, may have permission to log in to host raks in domain sun. Yet he would not be able to access any directories restricted to members of netgroup aswan. Moreover, user amina is only a member of netgroup aswan while logged in on host raks. Should she log in to host machine masr, she would not be able to access any directories restricted to netgroup aswan.

To define a netgroup consisting only of hosts in a particular domain, you might have an entry:

```
justmachines (raks,-,sun) (ballet,-,sun) (samba,-,sun)
```

This means that any users logged in to hosts raks, ballet, or samba in domain sun can access any directories restricted to members of the netgroup justmachines.

You can also define a netgroup consisting only of users in a particular domain. Consider the following line:

```
justpeople (-,amina,sun) (-,stefania,sun) (-,ernest,sun)
```

The three users--amina, stefania, and ernest--can access any directories restricted to netgroup justpeople, regardless of the machines within domain sun that they are logged in to. However, suppose user stefania logged in to a machine within domain moon, which communicates with domain sun over a router. She could not access any directories restricted to justpeople.

Create `/etc/netgroup`, by logging in as superuser and using your favorite text editor. Once you have created `/etc/netgroup`, you can use netgroup names when assigning mount access privileges in the `/etc/exports` file and for permission purposes in other files. You can also use the netgroup name when defining who is able to remote log in or set up a remote shell. This type of usage is explained in the the later section,

### The /etc/networks File

This file lists all the networks available to your server. The release tape contains a default /etc/networks

file, which lists two networks on the Internet—arpanet and ucb-ether—plus three provided by Sun. Here is a sample default /etc/networks file:

```
#
# Sun customer networks
# This file is never consulted when the yellow pages are running
#
loopback        127
sun-ether       192.9.200       sunether ethernet localnet
sun-oldether    125             sunoldether
#
# Internet networks
#
arpanet         10              arpa
ucb-ether       46              ucbether
```

loopback, sun-ether, and sun-oldether are the networks provided by Sun. *sun-ether* lists the Internet address you gave to suninstall for your local area network. *loopback* refers to the loopback file system provided by Sun. *sun-oldether* is an earlier Ethernet provided by Sun. After you install SunOS 4.0, your server automatically knows about these networks.

If you are running YP, you only need to update /etc/networks if you change your network configuration. Then you must change the master copy of /etc/networks then build its corresponding YP map, as explained in Chapter 14. However, if you are not running YP, you may need to update /etc/networks to reflect all networks that your server can use. This is particularly important because the netstat program uses the information in /etc/networks for the status tables it produces.

You need to update /etc/networks on the following occasions:

□ You just installed a router and you want to tell your server about the other network to which the router is attached.

□ You join a wide-area network, such as TELNET or BITNET, which is not listed in the file.

To modify /etc/networks,

1. Log in as superuser.
2. Edit /etc/networks using your preferred text editor.
3. Represent each network you want to add with a single line entry in the file. Use the following syntax

```
    official_network_name   network_number    aliases
```

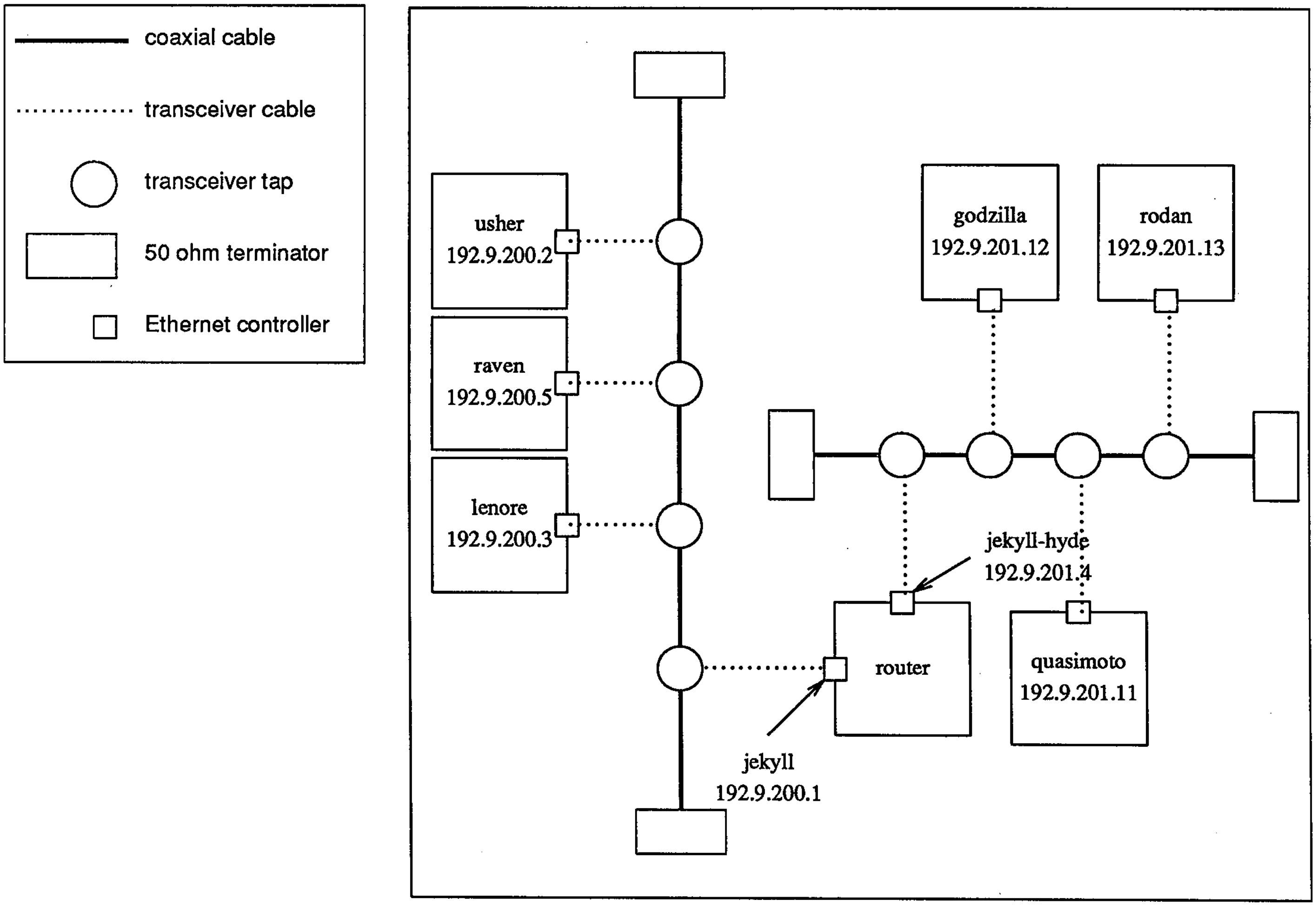
*official network name* is the name by which the network is known. *network number* is the network's Internet address. *aliases* are any aliases by which the network is also known. You can also add a comment to the line by typing a # and then the comment text.

### Setting Up a Router to Join Two Networks

A router machine enables you to attach two networks that are in close proximity to one another. If you have a single local area network that is not connected to any other network, you do not need a router. However, if you have more than one terminated coaxial cable in your complete physical Ethernet, you need a router to join the local area networks on each coaxial cable. A router often will have two Ethernet controllers, one to communicate with each local area network.[1] Because each machine must have a unique hostname for each Internet address, a router must be assigned a hostname and Internet address for each network it is on. Figure 12-2 below depicts two local networks joined by a router.

[1] Note that the second Ethernet board in a machine uses the same Ethernet address as the first board. The second board gets its address from the PROM of the first board and uses it to masquerade as the machine on the second network.

Figure 12-2 *Two Local Networks Joined By a Router*



To set up a proper configuration for a router, you need to acquire a registered IP network number for each network, as explained in the previous subsection. You then need to name the network, and put an entry for it in the `/etc/networks` file. (Refer to Chapter 13 and the `networks`(5) man page for more information.)

To set up a proper configuration for a router, you must follow these steps:

1. Edit the `/etc/hosts` file on the router. If you are running YP, edit the `/etc/hosts` database on the YP master server,

   Add a line for the router's hostname and address on the "second" (or other) network. In the example below, "jekyll" is a router between two local area networks. Remember, jekyll must have two Ethernet interfaces. Before you edit it, the `/etc/hosts` file might resemble the following:

```
# 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
192.9.200.5       raven
[ etc. ]
# Local Net 192.9.201 -- 10Mb/s Ethernet -- Marketing
#
192.9.201.11       quasimoto
192.9.201.12       godzilla
192.9.201.13       rodan
[ etc. ]
```

2. To make jekyll a router to the Marketing local area network 192.9.201, add an entry for its address on that network:

```
# 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
192.9.200.5       raven
[ etc. ]
# Local Net 192.9.201 -- 10Mb/s Ethernet -- Marketing
#
192.9.201.11      quasimoto
192.9.201.12      godzilla
192.9.201.13      rodan
192.9.201.4       jekyll-hyde

[etc. ]
```

Notice that on the second network jekyll has an alter-ego name, "jekyll-hyde." The primary name here is jekyll, but for the network software to work properly, the router must have a unique hostname in the /etc/hosts file for each interface. For ease of administration, use similar hostnames for each network— people on both networks can address the machine by the primary name (in this case, jekyll), while the system administrator can tell the difference between the two.

3. If your network uses YP, run ypmake(8) on the new /etc/hosts database to propagate it to all machines that are using YP. Go to the /usr/etc/yp directory and type the following:

```
# make hosts
```

This updates the /etc/hosts database, then performs a yppush to propagate the change to all the slave servers on the network. For details see ypmake(8).

4. On the router, edit `/etc/hosts` and add the same hosts entry as you did in the YP master server's `/etc/hosts` file. The resulting `/etc/hosts` file on the router should now contain both Internet addresses and host names for this router:

```
192.9.200.1 jekyll loghost
192.9.201.4 jekyll-hyde
```

5. On the new router, edit `/usr/etc/rc.boot`, and add an for`ifconfig`-line second hostname (in the example above, `loghost`), and Ethernet controller. The relevant line from the `/usr/etc/rc.boot` script looks like the following before adding router jekyll:

```
/usr/etc/ifconfig xx0 jekyll -trailers up
```

Here is how it looks when you add jekyll as a router:

```
/usr/etc/ifconfig xx0 jekyll -trailers up
/usr/etc/ifconfig xx1 jekyll-hyde -trailers up
```

For *xx* above, you substitute the proper Ethernet controller type, either `ec` for a 3Com controller, `ie` for an Intel controller, or `le` for a LANCE controller.

## Setting Up Subnets

Subnets are logical subsections of a single Internet network. For administrative or technical reasons, many organizations have chosen to divide one Internet network into several subnets. The standard subnetting scheme is called the "Address Mask" approach.

SunOS 4.0 includes support for Internet standard subnets. Subnets allow more flexibility in the assignment of network addresses. Network numbers are subdivided into three categories: Class A, Class B, and Class C. There are 127 Class A networks with 24 bit host fields, 16,383 Class B networks with 16 bit host fields, and over two million Class C networks with eight bit host fields.

Routing can get very complicated as the number of networks grows. For example, a small organization might give each Ethernet a Class C number. As the organization grows, the administration of network numbers could get out of hand. A better idea is to allocate a few Class B network numbers: one for Engineering, one for Operations, one for Support, one for Sales, and so on. Then, divide each Class B network into physical Ethernets using subnets. In this way, machines are isolated from topology changes in remote parts of the organization.

### Network Masks

When setting up your network, a network-wide *network mask* must be selected. The network mask determines which bits will be the subnet number, and the rest is the host within the subnet. For example, an organization could be one Class B network, with each Ethernet (or SunLink connection) assigned a subnet number within that network. The 16 bits could be allocated as eight for subnet and eight for host, or nine for subnet and seven for host, and so on. However, this decision would be transparent to everyone outside that organization.

You can express network masks as a single hexadecimal number, or as a sequence of four decimal numbers in the IP "dotted notation." The default is a mask of 0xFF000000 (255.0.0.0) for Class A networks, 0xFFFF0000 (255.255.0.0) for Class B networks, and 0xFFFFFF00 (255.255.255.0) for Class C networks. Network masks must only be explicitly specified when they are "wider" (that is, have more one-bits) than the default values. One common case is a Class C mask on a Class B network.

Sun systems previously used a non-standard broadcast address (all zeros in host part). Release 4.0 is more liberal about accepting all broadcast addresses that are in use, but still uses the all-zeros-host for its own broadcasts. SunOS Release 4.0 provides a mechanism to set the broadcast address. The main implication of this is: DO NOT USE ALL ONES in the host part for any host addresses. You should reconfigure systems attached to networks with machines that run earlier SunOS releases or that do not support standard broadcast addresses, so that they use the old, non-standard broadcast address. When the network includes exclusively machines that support the standard broadcast address, you can configure all of the systems to use it.

To set up the netmask, you need to edit the file `/etc/netmasks.` which is described in the `netmasks`(5) man page.

```
#
# Network masks database
#
# only non-standard subnet masks need to be defined here
#
# Network       netmask
128.32.0.0      255.255.255.0
```

When you edit this file, you need to provide the network number and network mask on a separate line for each network that is subnetted. If you are using YP, you should type the following on the YP master server:

```
% cd /etc/yp
% make
```

This propagates the `netmasks.byaddr` map. The map is used by the + argument to `ifconfig` in `/etc/rc.local` files during booting. You can also use `ifconfig` to manually override the network masks. For more information about `ifconfig`, refer to the `ifconfig`(8c) man page.

As an example, consider the Class B network 128.32 with an eight-bit wide subnet field (and, therefore, an eight-bit wide host field). The `/etc/netmasks` entry could be as follows:

```
128.32.0.0  255.255.255.0
```

Symbolic names defined for subnet addresses can be defined in the `/etc/hosts` file. These subnet names can be used instead of numbers as parameters to commands.

### Changing from a Non-subnetted to a Subnetted Network

Follow these steps to change from an internetwork that does not use subnets to one that uses subnets.

1. Decide on the new subnet topology, including considerations for subnet routers and locations of hosts on the subnets.
2. Assign all subnet and host addresses.
3. Edit `/etc/netmasks` as mentioned previously.
4. Edit `/etc/hosts` files to change host address. If YP is running, change the YP database.
5. NFS servers should be sure that `rarpd` runs on the interface connected to the NFS client's subnet. `rarpd` is started by `/etc/rc.local.` Change the IP address of NFS clients in the `/tftpboot` directory.
6. YP servers should also follow the setup procedure described in Chapter 14, "The Sun Yellow Pages Service."
7. Connect the physical networks and reboot all machines.

### Examples of Subnets

The following examples show network installations where subnets are (and are not) in use:

| | | | |
|---|---|---|---|
| 128.32.0.0 | Berkeley class B network | (subnetted) | netmask 255.255.255.0 |
| 36.0.0.0 | Stanford class A network | (subnetted) | netmask 255.255.0.0 |
| 10.0.0.0 | ARPAnet class A network | (non-subnetted) | netmask 255.0.0.0 |

All of the University of California at Berkeley is assigned the network number 128.32.0.0, so that any outside user only needs to know one route to access Berkeley. Within the campus, a class C subnet mask is used to give each individual Ethernet a subnet number, with 254 hosts on each of the 254 possible subnets. (Zero and all ones are reserved.) Stanford University uses a class A network number with a class B network mask, for 254 subnets of 65534 hosts each. The ARPAnet itself is a class A network without subnets; therefore, the default class A netmask is used.

## 12.4. Diagnosing Network Problems

This next section assumes you are an experienced network administrator. It deals with trouble-shooting problems in the networking software.

### Using a Local Disk to Offload the Network

There are several benefits to using a local disk to offload the network. A possible configuration would be to have many workstations, each with a local disk, use a central NFS server for their root and home directories. You would then use the local disk for swap and read-only executables (`/usr`), thereby avoiding network traffic for accesses to them. This puts all the volatile files on the server where backups are done, and allows you to disconnect a broken disk and replace it with another, standard disk, without worrying about restoring files.

## Software Checks

Here are some actions you can take to diagnose and fix software-related problems.

1. Check the /etc/hosts file (or its corresponding YP map, if your network has YP) to make sure that the entries are correct and up-to-date. If you are running YP, make sure a correct copy has been sent to all YP slave servers. Check the Ethernet addresses in /etc/ethers (or its corresponding YP map, if applicable) to make sure that the entries are correct and up to date. If you are running YP, make sure a correct copy has been sent to all YP slave servers.

2. Check the accuracy of the /ifconfig line(s) added to the /etc/rc.boot file. It should have a line like the following:

```
ifconfig ie0 $hostname -trailers up
```

   If your machine has more than one network interface you will have more than one ifconfig line.

3. On a standalone host or a server, try to rlogin to yourself. Make sure the network daemon inetd is running on the machines that want to communicate with each other.

4. Check the kernel configuration file to make sure the proper device description line for your communications controller has been enabled.

5. Use the netstat command to determine network status. netstat graphically displays the status of network traffic in table format, including routing table information (available routes and their status), and interface information. Refer to the netstat(8c) man page for more information.

   When you use netstat -i, you can find out how many packets a machine thinks it is transmitting and receiving on each local network. For example, on a server you may see the input packet count increasing each time a client tries to boot, while the output packet count remains steady. This suggests that the server is seeing the boot request packets from the client, but does not realize it is supposed to respond to them. This might be caused by an incorrect address in /etc/hosts or /etc/ethers. On the other hand, if the input packet count is steady, the machine does not see the packets at all, which suggests a different type of failure, possibly a hardware problem.

6. To find out if a machine on the network is receiving packets, go to another machine on the same network and issue the ping command as follows:

```
% ping hostname
```

   where *hostname* is the machine you suspect may not be receiving packets. If this condition does exist, you will get the message

```
no answer from hostname
```

7. Run the `spray` program to troubleshoot problems between two machines on the network. `spray` sends a sequence of RPC calls to a given host. You can use it to find out whether a machine is operational at the RPC layer, and to get some idea of whether the machine is experiencing a high-packet error rate. `spray` is similar to `ping`, but it works at a higher level in the networking hierarchy. Refer to the `spray`(8c) man page for more information.

8. Run the `etherfind` command, which is an invaluable tool for diagnosing network problems. It allows you to: inspect packet traffic between two hosts; examine packet traffic for certain types of packets, such as ARP and IP; select specified amounts of network traffic for later analysis. These capabilities are very useful in tracking down what has gone wrong in a malfunctioning network activity. Refer to the `etherfind`(8c) man page for more information.

9. Run the `traffic` command. `traffic` examines Ethernet activity, as does `etherfind`. But instead of reporting information on individual packets, `traffic` generates graphs giving statistical information on network activity as a whole. For more information, refer to the `traffic`(1c) man page.

10. Run `perfmon`, a program that displays a graphic representation of general system statistics for a workstation. These statistics are:

| | |
|---|---|
| user | percentage of total CPU time spent in user processes |
| system | percentage of total CPU time attributed to system calls and overhead |
| idle | percentage of total CPU time spent idle |
| free | amount of available real memory in Kbytes |
| disk | total amount of disk transfers performed |
| interrupts | total number of interrupts serviced |
| input | total number of input packets received |
| output | total number of output packets transmitted |
| collision | total number of collisions between packets observed on the network |

Refer to `perfmon`(1) for more information.

11. Run the `perfmeter`, a program that provides a meter display of system performance values. These values include: percent of CPU being utilized, Ethernet packets, paging activity in pages per second, and many other options described in .`perfmeter`(1)

### Performance Considerations

You can have most of the advantages of living in a network environment without running the network daemons `in.routed` and `in.rwhod`. The `rwho` daemon (`in.rwhod`) is not run by default because it has a severe impact on performance. You are strongly advised not to run it. If you need to run it, edit `/etc/rc.local` and remove the pound sign at the beginning of the line that starts `in.rwhod`. (Refer to `rwhod`(8c) if you need more information.)

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 to keep accurate information about the topology of the local network.

Whether or not where you should run routed depends on your system configuration. With limited memory, consider which one of the following suggestions best applies to you, and follow through:

□ If there are no routers on your local network then you do not need to run routed. You can disable the daemon by removing (or commenting out by inserting a pound sign (#) before each line) the following 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
```

□ If your machine has limited memory **and** only one router in your local network, then you can run routed only on the router, and disable it on all other machines. All machines on the local network redirect traffic for other networks through this router. To do this, edit the /etc/rc.local file on all machines except the router (*router_name*, in the example). Find the following line:

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

Insert the following two lines just before it:

```
/usr/etc/route add default router_name 1
echo '  /usr/etc/route add default router_name 1'>/dev/console
```

Then, find the following 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
```

## Logging Network Problems

If you suspect a routing daemon malfunction, you may log its actions — and even all the packet transfers. To create a log file of routing daemon actions, just supply a file name when you start up the in.routed daemon, for example:

```
# /usr/etc/in.routed /etc/routerlog
```

(Refer to the routed(8c) man page for more information about in.routed.)

Whenever a route is added, deleted, or modified, a log of the action and a history of the previous packets sent and received will be printed in the log file. To force full packet tracing, specify the -t option when the daemon is started up. Beware though — on a busy network this generates almost constant output.

# 13

# The Sun Network File System

# 13

# The Sun Network File System

This chapter explains how to administer the Sun Network File System (NFS) service. It also tells you how to diagnose and fix NFS-related problems. The text explains:

- How NFS works, as an overview
- How to administer an NFS server
- How to administer an NFS client
- How to set up network security
- How to troubleshoot NFS-related problems

You need to read at least part of this chapter, unless you administer a non-networked standalone. If you administer a diskless client, you should read the NFS overview and sections that explain how to mount directories, set up network security, and diagnose problems on an NFS client. If you administer an NFS server, read the entire chapter. You may have to fix problems on clients as well as on your server. If you administer a client machine with a disk (dataless or networked standalone) that will share files, you should read the information about exporting files in the section, "How to Set Up and Administer an NFS Server," plus the sections recommended for a diskless client.

## 13.1. How NFS Works-an Overview

NFS is an RPC service that enables machines to share file systems across a network. It performs two major functions: mounting and exporting. This section presents general concepts concerning these functions. It also explains how various daemons interact to enable exporting and mounting.

### Servers and Exporting

When a server *exports* file systems, it is essentially advertising the directories on its disk(s) that it will permit other computers to access. The server's `/etc/exports` file lists these available directories, which clients are allowed to access them, and any access restrictions that must be applied. When you boot an NFS server, the `/etc/rc.local` script automatically starts up a program called `exportfs`. This program then looks at the `/etc/exports` file and informs the server's kernel about the permissions applicable to each exported file system.

You can also explicitly export and "un-export" a file system after the server is up by using the `exportfs` command. Explicit exporting is explained in "Setting

Up and Maintaining an NFS Server."

### Clients and Mounting

Clients access files on the server by *mounting* the server's exported directories. When a client mounts a directory, it does not make a copy of that directory. (Do this by using the `rcp` or `tftp` commands.) Rather, the mounting process uses a series of remote procedure calls to enable a client to transparently access the directories on the server's disk.

RPCs running on the server receive the information in XDR format, and translate it to a form recognizable to the server's kernel. The kernel then processes the information and permits (or does not permit) the client to mount the file system.

Once a client mounts a remote directory, it appears to the users that all they have to do to go to a mounted directory is to type `cd` and the directory's pathname, just as they would for a local directory.

The process by which a client locates a server that exports the information it wants, then sets up communication between itself and that server, is called *binding*. NFS binding occurs during an NFS mount. (Clients also initiate binding to YP servers, as explained in Chapter 14, "The Sun Yellow Pages Service.")

A client can mount a directory when it boots, or can explicitly mount a directory when a user issues a `mount` command. The `/etc/fstab` file, first introduced in Chapter 6, "The SunOS File System," lists all file systems that the client mounts at boot time. You can also explicitly mount a file system during a work session by using the `mount` and `umount` commands.

### What Happens When a Client Mounts a Directory

The following overview briefly summarizes the activities that take place when an NFS client mounts a directory. Its purpose is to introduce you to the daemons and system programs that handle NFS requests. On a server, NFS service is controlled by the `exportfs` system program and the `rpc.mountd` and `nfsd` daemons. On a client, NFS service is handled by the `mount` system program and the eight `biod` daemons.

Note that it is not critical for you to understand these daemons in depth. You simply need to know that they exist, because you may have to restart them if they stop after a crash. Should you want to know more about a particular daemon, refer to its man page in the *SunOS Reference Manual.*

This summary begins as the server and clients boot up:

1. When the server reboots, `rc.local` executes `exportfs`, which reads the server's `/etc/exports` file, then tells the kernel which directories the server can export and what access restrictions—if any—are on these files.
2. The `rpc.mountd` daemon and several `nfsd` daemons (usually about four) are started automatically by `rc.local` when the server boots.
3. When the client reboots, its `fstab` file is automatically read by the `mount` program.
4. The `mount` program then requests the server to allow the client to access the directories in the client's `/etc/fstab` file.

5. The server's `rpc.mountd` daemon handles the client's mount requests.
6. If the requested directory is available to that client (or to the public), the `rpc.mountd` daemon sends the client's kernel an identifier called a *file handle*.
7. When it does a file operation, the client kernel then sends the file handle to the server, where it is read by one of the `nfsd` daemons to process the file request.
8. The `nfsd` daemons know how a directory is exported from the information sent to the server's kernel by `exportfs`. These daemons allow the client to access the directory according to its permissions, by way of the file handle.
9. The client's `biod` daemons improve client performance while the directory is accessed. They are not necessary for NFS to work, but they do increase performance.

## 13.2. Setting Up and Maintaining an NFS Server

This section explains the files you need to edit or create to set up and successfully maintain an NFS server, and other actions you must take after its initial installation. Recall that when you set up the NFS server during `suninstall`, you did the following:

□ Identified the machine as an NFS server.

□ Defined the server's disk or disks, indicating which partitions would hold the file systems to be shared among the clients.

□ Defined parameters for each of the server's clients on the Client Form.

After you reconfigure the kernel and boot up your NFS server, it has dedicated root directories, home directories, and swap space for each client on its disk, as defined through `suninstall`. Furthermore, it has a default `/etc/exports` file that automatically exports these directories, which clients need as soon as they boot up.

Nevertheless, at this phase you have only established basic NFS service. You still need to perform many other NFS-related activities, mostly in the area of file creation and modification. This is particularly important if you do not plan to use the Yellow Pages (YP) service.

Other activities you may need to perform include:

□ Add additional directories to the default `/etc/exports` file, if needed.

□ Change access permissions to `/etc/exports`, as needed.

□ Directly export and unexport directories during a server's work session.

□ Modify the server's `/etc/fstab` file, if more than one NFS server is on your network. Your server can then mount exported file systems from the other servers.

- Create mount points in the server's file systems for directories that it will mount from other servers.
- Upgrade a homogeneous server to heterogeneous, as required by your site.
- Upgrade a standalone workstation to NFS server, if required.
- Edit the following files if you are not going to use YP:

  ```
  /etc/hosts
  /etc/ethers
  /etc/netgroups
  ```

  They are described in the previous chapter.
- Check network status on a regular basis, and update related files, as necessary.
- Set up security for your network, as determined by your site's requirements.
- Diagnose and fix NFS related problems as they arise.

### Exporting Directories

You can control how a server exports directories in two ways:

- At boot time, through the `/etc/exports file.`
- As needed, by using the `exportfs` command.

### The `/etc/exports` File

The `/etc/exports` file advertises all file systems that a server exports to its clients. Mounting is initiated by the client. Through `/etc/exports`, servers can control which client may mount a directory by limiting access to it to a desired client or netgroup.

When you first boot up a server after installing Release 4.0, its `/etc/exports` file resembles the following:

```
/                access=clients
/usr             access=clients
/home            access=clients
#
/export/root/ballet -root=ballet,access=ballet
/export/swap/ballet -root=ballet,access=ballet
#
/export/root/raks -root=raks,access=raks
/export/swap/raks -root=raks,access=raks
#
/export/root/samba -root=samba,access=samba
/export/swap/samba -root=samba,access=samba
```

Each line in the file has the syntax:

*directory -option[,option]*

where *directory* is the pathname of a directory, a file, or, in some cases, the name of an entire file system. *option* represents a list of options that indicates which clients are allowed to access the directory, and if read-only restrictions are on that directory.

The *SunOS Reference Manual* fully describes all options in the man page `exports`(5). More frequently used options are defined below.

`ro`

This option specifies that clients are limited to read-only access to the specified directory. Therefore, they can read the directory's files but cannot change them. If `ro` is not specified for a directory, clients are given read-write (`rw`) access by default.

`root=`*hostnames*

This option indicates that root access to the directory is given only to superusers from a specified *hostname*. In the previous example, the line

```
/export/root/raks -root=raks
```

indicates that root access to `/export/root/raks` is given only to those who can run superuser on client raks. Note above that root access to the other default client directory `/export/swap/raks` is also given to superusers on client raks. A superuser on client samba, for example, cannot access `/export/root/raks` unless local files on client raks grant samba this type of permission.

`access=`*client*

This option indicates that access to the particular directory should be given to the named client, clients, or netgroup. To limit access to a directory to a single client, use the parameter `access=`*client_name*. The parameter

```
-access=raks
```

grants mount access to `/export/root/raks` and `/export/swap/raks` to all users on client raks.

You can use the word *clients* to enable all the NFS server's clients to mount a directory. In the example above, the line

```
/               access=clients
```

indicates that all the server's clients are allowed to mount `/`.

Finally, you can limit mount access to a directory to a designated netgroup. A netgroup is a network-wide group allowed access to certain network resources for security and organizational reasons. You set up netgroups either through YP, as described in Chapter 14 or through the `/etc/netgroup` file described later in this section.

### Editing /etc/exports

When you bring up your server after installing Release 4.0, you probably will want to modify the existing contents of etc/exports. For example, you might want to designate that /usr be exported read-only, designate access of a directory to a netgroup, or add more directories for exporting. Once you have modified /etc/exports, the server will automatically export these files every time you boot it up. Therefore, you should modify /etc/exports so that it contains directories that you want the server to export all the time.

Below is a set of instructions that explain how to edit /etc/exports.

1. Run superuser and edit the /etc/exports file, using your preferred text editor, for example, vi.

```
# vi /etc/exports
```

2. Enter the mount point pathname of the directory you want to add into the file. For example, suppose you want all members of the accounting netgroup to mount a new billing program that you have installed in its own file system, /billing. Because this information is confidential, you do not want network users who aren't in accounting to mount this directory. The entry in /etc/exports should look like the following:

```
/billing        -access=accounting
```

3. Write the modified file.

4. Run exportfs as follows, to update the exports information in the server's kernel.

```
# /usr/etc/exportfs -a
```

   The -a option tells exportfs to send all information in /etc/exports to the kernel.

You cannot export a directory that is either a parent or subdirectory of a directory *within the same file system* that is already exported. It would be illegal, for example, to export both /usr and /usr/local if both directories resided on the same disk partition.

### Explicitly Exporting Directories with exportfs

SunOS uses /usr/etc/exportfs in a number of ways to export and unexport directories to NFS clients. You have seen how rc.local runs exportfs at boot time, and how you can use exportfs during a work session to update /etc/exports. You can also run exportfs at any time to explicitly export and unexport directories. You probably will want to use exportfs in this fashion for directories that you only want to export for a finite amount of time, then unexport.

**Note:** You must be superuser to use exportfs.

exportfs has the following syntax:

```
/usr/etc/exportfs [ -avu ] [ -o options ] [ directory ]
```

Its arguments are completely described in `exportfs(8)` in the *SunOS Reference Manual.* The more commonly used arguments are defined below:

`-a` Export all the directories in `/etc/exports`, as illustrated in the previous subsection.

`-u` Unexport the indicated directories.

`-o` *options* Execute the list of options following `-o`. The arguments to *options* are the same as the options you can add to `/etc/exports`. For example, you can specify

```
/usr/etc/exportfs -o ro directory
```

to export a directory with read-only access. You can also use the

```
root=hostname
```

and

```
access=client
```

parameters in the same fashion as in `/etc/exports`.

*directory* This is the full pathname of the directory that you want to export or unexport.

If you type `/usr/etc/exportfs` without an argument, you receive a display of currently exported directories.

## The `/etc/xtab` File

Whenever `exportfs` runs, it updates the `/etc/xtab` file, which lists currently exported directories. `/etc/xtab` has an identical format to `/etc/exports`, but its contents change whenever you run `exportfs`. On the other hand, `/etc/exports`' contents remain static until you go in and change them.

Suppose you explicitly exported a directory with `exportfs`, then checked the `/etc/xtab` file. That directory would be listed, although it would not be listed in `/etc/exports`. Moreover, if you then unexported the directory using `exportfs -u`, then checked `/etc/xtab`, the unexported directory would, of course, no longer be listed.

## Upgrading an NFS Server from Homogeneous to Heterogeneous

As your site's needs grow, you may need to add clients of a different architecture to a network served by a homogeneous NFS server. The `/usr` file system of a homogeneous NFS server contains programs that can only be used by machines with the same architecture as the server. Therefore, these executable programs are referred to as "architecture dependent." If the existing server now must support clients of several architectures, you have to install the `/usr` executables for those architectures from the 4.0 Release tape. The `/usr/etc/install/setup_exec` command extracts these files from the tape loaded in a local or remote tape drive, then places the files in the directory you specify.

**Note:** You can no longer use extract_release to extract additional software if you do not want to rerun suninstall after installing a system. You must use setup_exec to extract software from the same or other architectures or rerun suninstall.

Here is the syntax of setup_exec:

```
# /usr/etc/install/setup_exec arch execpath
```

where *arch* indicates the architecture type of the executables to be extracted. Its values are sun2, sun3, or sun4. *execpath* is the directory where the executables are to be installed.

You can use setup_exec to install as many architectures as you need on the server as long as there is enough space available on the server's disk. Run the command once for each architecture to be extracted. You do not need to reload the executables for the server's architecture type.

For example, suppose you load the 4.0 release tape in the local drive of a Sun-3 server and type the following:

```
    # /usr/etc/install/setup_exec sun4 /export/exec
```

setup_exec loads Sun-4 executables from the local tape into a directory called /export/exec/sun4. After the executables are installed, the server's Sun-4 NFS clients mount /export/exec/sun4 as their /usr file systems. setup_exec also updates the server's /etc/exports file.

For instructions for using setup_exec to load files from a remote tape drive, refer to *Installing the SunOS*.

## Converting a Standalone System to NFS Server

You can convert a standalone system running Release 4.0 to an NFS server by running /usr/etc/install/setup_exec and /usr/etc/install/script/setup_client. (The latter command adds new clients to an already existing server, as described in the next section, "Setting Up and Maintaining and NFS Client.") Use the same setup_exec syntax for upgdrading a standalone as you would for upgrading a server.

For example, suppose you have a Sun-2 standalone system running Release 4.0, and you want to upgrade it to a heterogeneous NFS server supporting both Sun-2 and Sun-3 4.0 clients. You need to run setup_exec to install Sun-3 executable files onto the system. You do not need to install Sun-2 executable files, since suninstall has already placed these files in your standalone's /usr file system. If you want to load the Sun-3 executable files into /export/exec/sun3, specify exec for *execpath*. Then run setup_client to add clients to the new server.

## The NFS Server's /etc/fstab File

The /etc/fstab file shows all the directories that a machine mounts when it boots up. It is completely described in the fstab(5) man page. You need to modify a server's fstab when you want it to access directories from another server on the network.

When you first set up a Sun computer, suninstall automatically creates a default /etc/fstab for it. This default file is slightly different for a server than for a client. Below is a sample default fstab for a server.

```
/dev/xd0a / 4.2 rw,nosuid 1 1
/dev/xd0h /usr 4.2 rw 1 2
/dev/xd4c /home 4.2 rw 1 3
/dev/xd0g /export/dump 4.2 rw 1 4
/dev/xd0e /export/swap 4.2 rw 1 5
/dev/xd0f /export/root 4.2 rw 1 6
```

Lines in the file have the following syntax:

```
file_system  mount_point  type  options  freq  pass
```

Here is the meaning of each parameter.

*file system* This is the name of the file system that the server mounts. In the first line of the example above, it is `/dev/xd0a`—the server's root file system. Notice that all entries in the first column correspond to partitions on the local disk designated during `suninstall`.

*mount_point* This is the location within the directory tree where the server accesses the files designated by the *file system* parameter. In the first entry, the server mounts `/`, which is in file system `/dev/xd0a` (Partition a). Mount points are fully explained in the next section, "Setting Up and Maintaining an NFS Client."

*type* Type indicates the type of mount used to access the particular directory, either 4.2 or NFS. In the default `fstab` the server as yet only does local, 4.2 mounts to its own disk. However, if another server is on the local area network, you can also add remote NFS mounts to your server's `fstab` file, as explained in the next major section.

*options* This is a list of options that you can choose for the type of mount; they are fully described in the `fstab`(5) man page. For example, with a 4.2 mount you can specify a `quota` option to enable the quota system. In the first line of the sample `fstab` file, the `rw` option is given, which enables the server to mount the specified directory with read-write permission—the default.

*freq* This parameter indicates the interval in days between dumps.

*pass* This parameter indicates the pass in the `fsck` program during which this file system will be checked. If you indicate a 0 for this parameter, the file system is not checked during `fsck`. (Refer to Chapter 17, "File System Check Program" for more information about `fsck` passes.)

The contents of the `fstab` file do not change unless you explicitly modify it. You will want to do this for your server if it is one of several on a local area network. To mount files from a remote server, you add NFS mounts to your server's `fstab` file. These are explained in the next section, "Setting Up and Maintaining an NFS Client."

## 13.3. Setting Up and Maintaining an NFS Client

This section contains information related to setting up and maintaining any type of NFS client machine on a network. You should read this if you are responsible for administering your client machine, or if you are administering both servers and clients on your network.

Activities you need to perform to administer a client include:

- Making mount points in your client's directory tree for the directories mounted through the `/etc/fstab` file.
- Modifying the client's `/etc/fstab` file to include directories the client is to mount
- Directly mounting and unmounting directories as needed
- Adding and removing clients of an already existing NFS server.
- Updating local files on the client.
- Diagnosing and fixing network-related problems on the client.

### Mounting Files from an NFS Server

The most important service NFS provides to clients is the ability to access directories on a disk on a remote machine, a process called an NFS `mount`. Both NFS and 4.2 mounting occur during booting, when the `mount` program reads the `/etc/fstab` file. You can also dynamically mount and unmount directories at any time by using the `mount` and `umount` commands. This next section explains how to set up clients to perform NFS mounts.

#### Making Mount Points

Mount points are locations within a directory tree through which your computer accesses directories. These directories can be mounted locally from a disk or tape, or remotely from another computer on the network. For example, when you use the `restore` command as described in Chapter 7, you mount the files on the backup tape through `/mnt`, an already existing mount point in the root file system. Furthermore, when `suninstall` creates a client's `root`, `swap`, and `home` directories in a server's `/export` file system, it also creates the mount points `/`, `/swap`, and `/home` for them in the client's directory tree. If the client must mount additional directories, you need to create mount points for them.

Mount points are essentially empty directories, which you create through the `mkdir` command. Simply type

```
% mkdir mount_point
```

where *mount_point* is the full pathname of the empty directory you want to create. You need to create mount points for any non-default directories mounted through the `/etc/fstab` file or through the `mount` command.

## An NFS Client's fstab File

When a server is first set up, suninstall creates default fstab files for each of the server's clients. The default client /etc/fstab file resembles the following, set up by server dancer for client raks:

```
dancer:/export/root/raks / nfs rw 0 0
dancer:/export/swap/raks / nfs rw 0 0
dancer:/usr /usr nfs ro 0 0
dancer:/home/dancer /home/dancer nfs rw 0 0
```

Entries in the file have the following syntax:

```
server:server_dir client_mountpoint type options freq pass
```

Here is how these parameters apply to the sample fstab file above.

*server* This is the name of the server exporting the directory the client wants to mount. In the sample file, dancer is the only server listed.

*server_dir* This is the name of the directory to be mounted from the indicated server. In the first entry, the directory is /export/root/raks, client raks' individual root directory. Note that in the default fstab file for the client, the only mounted directories are those that the client requires to operate: its individual root, swap, and home, plus /usr, which, along with the client root, contains all essential SunOS programs.

*client_mountpoint*
This is the mount point on the client through which it accesses the directories mounted from the server. In the first entry above, the mount point is the client's root directory.

*type* This is the type of mount taking place. In the example above, this is an nfs mount. If the client has its own disk, you can also set up its fstab file so that it can perform 4.2 mounts of file systems on its local disk.

**Note:** An NFS server can also be a client of another NFS server on a local network. In this case, you edit the server's fstab to include both 4.2 and nfs mounts.

*options* This can be any one of a number of options provided for nfs or 4.2 mounts. Refer to the fstab(5) man page for a complete list. For example, if you are running secure SunOS, you may want to select the secure option for nfs mounts.

The example file shows the client performing nfs mounts with the common options rw or ro. The client can mount its own root, swap, and home directories with read-write access. However, it accesses the server's /usr directory with read-only permission; thus a user on the client cannot add or modify a file in /usr.

*freq* This is the interval in days between dumps of the directory.

*pass* — This indicates the `fsck` pass in which the listed file system is checked. If a zero is indicated, the file system is not checked during `fsck`.

The contents of `/etc/fstab` remain the same until you change them.

## Mounting and Unmounting Directories

This subsection shows how to perform two different mount procedures: statically, through the `fstab` file, or explicitly, through the `mount` and `umount` commands. You can use them to enable both nfs and 4.2 mounts.

### Procedures for Modifying the Client's `fstab`

If you want the client always to mount a particular file system, you should add an entry to its `fstab` file.

1. Log in as superuser.
2. Edit `/etc/fstab` using your preferred text editor.

**Note:** If you are editing a client's `fstab` file while logged in to your server, this file is located in `/export/root/`*client_name*`/etc/fstab`.

3. Create an entry in the file, using the syntax shown in the previous subsection. List the server exporting the directory you wish to mount, that directory's pathname, the mount point on the client, and so on.
4. Create the mount point in the client's directory tree using the `mkdir` command.
5. Type the following

```
# /usr/etc/mount -a
```

to mount everything in the current `fstab` file.

After you have finished these steps, your client will always mount the files in its `/etc/fstab` file until you modify it again.

### Explicitly Mounting and Unmounting Directories

Use the `mount` and `umount` commands during the course of client operations. It is advisable to use these commands for directories that you only want to access for a finite amount of time.

### Using `mount`

`mount` explicitly mounts a directory. It only requires that you can reach the directory's server over the network, and that the server's `/etc/exports` file allows you access to the directory. You must be superuser to use `mount` and `umount`.

Here is a form of the `mount` command that you can use for most mount operations. (Refer to the `mount(8)` man pages for the complete syntax of `mount`.)

```
# /usr/etc/mount -t type [-rv] -o [options] server_name: /directory /mount_point
```

These parameters are explained below.

`-t` *type* — This is the type of mount you want to perform, `nfs` or `4.2`.

*[-rv]* These are two options that you can supply with `mount`. `-r` specifies that you want to mount the directory read-only. `-v` specifies the verbose option, in which `mount` displays messages as it operates.

`-o` *options* This is a list of options specified after the `-o` flag, fully described in the `mount`(8) man page.

One important `-o` option is `soft/hard`. In a *hard* mount, the client continually tries to mount the directory, even if the server does not respond to its request. In a *soft* mount, the client does not retry the mount operation if the server does not respond to its first request.

*server_name* This is the name of the server exporting the directory.

*directory* This is the pathname on the server of the directory you want to mount

*mount_point* This is the mount point on the client through which the directory is mounted.

Directories accessed through the `mount` command stay mounted during a work session unless you unmount them with the `umount` command. Moreover, if you reboot the machine, the directory is automatically unmounted (unless you also edited the `fstab` file to include the mount.)

For example, you type the following:

```
# /usr/etc/mount -o soft dancer:/usr/man /usr/man
```

to mount the man pages from remote machine `dancer` to the local empty directory `/usr/man`:

Here are some final points to remember about `mount`:

- Do not forget to make the applicable mount point before issuing the `mount` command.
- Consider soft mounting directories such as the man pages so that if the server exporting them goes down, your client will not hang.
- Use the *hard* option with any file systems you mount read-write.

### Using `umount`

You use `umount` to explicitly unmount a currently mounted file system. A simple form of `umount` is shown below. (Refer to the `mount`(8) man page for more variations of this command.)

```
/usr/etc/umount [ -v ] mount_point
```

The `-v` option selects verbose mode, wherein `umount` displays messages as it runs. `mount_point` is the name of the mount point on the client where you have mounted the files from the server.

## The /etc/mtab File

Whenever you explicitly mount or unmount a file system, SunOS modifies the /etc/mtab file. This file has the same format as the fstab for a client or for a server, whichever applies. You can display /etc/mtab by using the cat or more commands, but you cannot edit it, as you would /etc/fstab.

## Adding and Removing Clients of an NFS Server

As your network grows, you may need to add or remove client machines. You can do this as required by running the program setup_client. (When you initially set up your NFS server and its clients, suninstall runs setup_client, to create clients according to the specifications on the Client Form.)

The syntax of setup_client is

```
/usr/etc/install/script/setup_client
op clientname yp_type swapsize rootpath swappath dumppath
homepath execpath arch
```

The setup_client(8) man page completely describes all arguments to the command. Briefly, they are:

*op* Operation to perform. You must specify **add** or **remove**, depending on the operation you want to perform.

*clientname* Hostname of the client.

*yp_type* Type of YP server or service to provide. Specify one of the following: **master,** if the machine is to be a YP master server; **slave**, for a YP slave server; **client**, for a YP client; **none**, for no YP service.

*swap_size* Number of bytes for the client's swap file. 10 Mbytes of swap space per client is recommended.

**Note:** This argument enables you to change swap size for an already existing client.

*rootpath* Pathname of the parent directory where the client's root directory is to reside. For example, if you want the client's root directory to be in /export/root/*client_name*, then specify **/export/root**.

*swappath* Pathname of the parent directory where swap files reside. If you want the client's swap files to be in /export/swap/*client_name*, specify **/export/swap**.

*dumppath* Pathname of the parent directory where dump files are to reside. suninstall does not automatically create a file system for client dumps, as it does for client root and swap files. If you have already set up a dump directory for client dump files, you should specify it here.

*homepath* Pathname of the parent directory that contains the client home directories. Specify **/home**.

*execpath* Pathname of the directory where the executables for the client's architecture type are stored. Specify **/export/exec/sun***n*, where *n* is 2, 3, or 4.

*arch* Architecture type of the client, either **sun2**, **sun3**, or **sun4**.

setup_client creates or removes one client at a time. The next three subsections contain sample procedures for using setup_client.

### Adding a Client to a Homogeneous Server

**Note:** Refer to Chapter 14 for instructions on adding a new client to a network with YP services.

This example shows how to add a Sun-2 client called "ballet" to a network without YP that is supported by a Sun-2 NFS server. You follow this procedure:

1. Become superuser on the server.
2. Add the client's Internet address to /etc/hosts.
3. Add the client's Ethernet address to /etc/ethers.
4. Type the following to set up the new client:

```
# /usr/etc/install/script/setup_client add ballet none 16M \
  /export/root /export/swap /export/dump /home /export/exec sun2
```

This command adds a machine called "ballet," which does not use YP. 16 Mbytes of swap space are reserved for client ballet in /export/swap/ballet. The client's root is /export/root/ballet; its dump files will go into /export/dump, a directory you previously created for your server with suninstall. The client's home directory will be in /home; its executables will be in /export/exec/sun2. The machine is specified as having Sun-2 architecture.

5. When setup_client is finished, you can go to the client machine and boot it up.

### Adding a Client to a Heterogeneous Server

This example shows how to add a Sun-3 client called "samba" to a network without YP that is supported by a Sun-4 NFS server. You do the following:

1. Perform Steps 1-3 shown in the previous subsection.
2. Type the following to set up the new client:

```
# /usr/etc/install/script/setup_client add samba none 10M \
  /export/root /export/swap /home /export/exec sun3
```

This command adds client machine samba, which does not use YP. It has 10 Mbytes of swap space reserved for it in /export/swap/samba. Its root is /export/root/samba; no directory is specified for its dump files. The client's home directory will be in /home; its executables will be in /export/exec/sun3.

3. Go to the subsection “Booting a New Client over a Heterogeneous Network“ in Chapter 5 for instructions for booting a new Sun-3 client.
4. Boot up client samba according to these instructions.

### Removing a Client

This example shows how to remove an existing client called “raks” from an NFS server.

1. Become superuser on the server and type the following:

```
# /usr/etc/install/script/setup_client remove raks none 10M \
  /export/root /export/swap /home /export/exec sun4
```

2. Remove entries for client raks in /etc/hosts and /etc/ethers.

### Files to Set Up for a New Client

When you add new clients to a new or existing network, you have to edit certain administrative files to enable the client to operate smoothly and securely. Become superuser on the client and do the following:

1. Edit the /etc/passwd file using your preferred text editor's standard editing command, as explained shortly in the section “Making the Network More Secure.”

**CAUTION:** Do not use /usr/etc/vipw to edit the password file. Using this command at this point brings in the server's password file, rather than the new client's, into the editor.

2. If user groups are in existence at your site, edit the client's /etc/group file. If YP is not present, move a copy of the NFS server's /etc/group file into the client's root directory. If YP is present, make the same changes to /etc/netgroup on the client's YP master server.
3. To enable the client to use printers on the network, copy the NFS server's version of this file to the client's root directory:

```
# rcp server_name:/etc/printcap /etc/printcap
```

4. To enable the mail facility on the new client, type the following on the client:

```
# rcp server_name:/usr/lib/sendmail.subsidiary.cf \
/usr/lib/sendmail.cf
```

Note that you may enter the above command on a single line. This display wraps to a second line only because of page size limitations. If you do want to wrap your command, you must escape the carriage return with a backslash (\) as shown above.

5. Edit the client's /etc/rc.boot file, changing the line near the beginning from

```
hostname=server_name
```

to

```
hostname=new_client_name
```

6. Check that the new client's domain name is specified on the first line of the `/etc/rc.local` file:

```
/domainname domain_name
```

7. Update the `/etc/hosts.equiv` and `.rhosts` files to specify who is allowed to log in to your machine over the network.

8. You may define the new users' environment on login in several ways. For example you may give them copies of such files as `.login` and `.cshrc` if they use `/bin/csh`, or `.profile` if they use `/bin/sh`. These files are completely described in *Setting Up Your SunOS Environment: Beginner's Guide* .

9. You may also want to give a user `.suntools` and `.mailrc` files. Remember to change ownership of these files to the user.

   If the user should belong to any mailing lists, add him or her to `/etc/aliases`. See Chapter 15, "Electronic Mail and Communications."

## 13.4. Handling NFS Problems

This section describes typical problems that occur on machines using NFS services. Topics discussed include:

- Strategies for tracking NFS problems
- NFS-related error messages

Before trying to clear NFS problems, you might want to reread the first section of this chapter, which describes how NFS works. Consider familiarizing yourself with the following man pages, if you have not done so already:

```
mount (8)
nfsd (8)
biod (8)
mountd (8)
inetd (8)
inetd.conf (8)
```

The information in this section contains enough technical details to give experienced network administrators a thorough picture of what is happen with their machines. If you do not yet have this level of expertise, note that it is not important to understand these daemons, system calls, and files fully. However, you should be able to at least recognize their names and functions.

## Determining Where NFS Service Has Failed

When tracking down an NFS problem, keep in mind that, like all network services, there are three main points of failure: the server, the client, or the network itself. The strategy outlined below tries to isolate each individual component to find the one that is not working.

For example, consider this sample mount request made from an NFS client machine:

```
% mount dancer:/usr/src /dancer.src
```

Below is a summary of how this command works and where it can fail.

1. The command requests server dancer to send a file handle (`fhandle`) for the directory `/usr/src`. This file handle is sent to the client kernel by the `mount` program.

2. The client kernel looks up the directory `/dancer.src` and, if everything is okay, it ties the file handle to the directory in a mount record. From now on all file system requests to that directory and its subdirectories go through the file handle to server dancer.

3. The `mountd` daemon must be present for a remote mount to succeed. Make sure `mountd` will be available for an `rpc` call by checking `/etc/inetd.conf` on the NFS server for this line:

```
mountd/1       dgram   rpc/udp wait root /usr/etc/rpc.mountd   rpc.mountd
```

If it is not there add it. For details, see `inetd.conf`(5).

4. Remote mount also needs some number, typically 8, of `nfsd` daemons to execute on NFS servers. Check `/etc/rc.local` for the following lines:

```
if [ -f /usr/etc/nfsd -a -f /etc/exports ]; then
/usr/etc/nfsd 8 & echo -n ' nfsd'       >/dev/console
```

Add these lines, or your own version of them, if the new NFS server's `/etc/rc.local` script does not enable `nfsd` daemons. You can enable these daemons without rebooting. Become superuser and type:

```
# /usr/etc/nfsd 8
```

### Debugging Hints

Below are some general pointers for debugging, followed by a list of possible errors and their probable causes.

When the network or server has problems, programs that access hard mounted remote files will fail differently than those that access soft mounted remote files. Hard mounted remote file systems cause programs to retry until the server responds again. Soft mounted remote file systems return an error after trying for a while. `mount` is like any other program: if the server for a remote file system fails to respond, it retries the mount request until it succeeds. When you use mount with the bg option, it retries the mount in the background if the first mount attempt fails.

Once a hard mount succeeds, programs that access hard mounted files hang as long as the server fails to respond. In this case, NFS should print

```
  NFS server not responding
```

on the console. On a soft mounted file system programs get the message

```
Connection timed out (ETIMEDOUT)
```

when they access a file whose server is dead. Unfortunately, many programs do not check return conditions on file system operations, so you may not see this error message when accessing soft mounted files. An NFS error message should be printed on the console in this case also.

If a client is having NFS trouble, check first to make sure the server is up and running. From a client you can type

```
  % /usr/etc/rpcinfo -p server_name
```

to see if the server is up at all. It should print out a list of program, version, protocol, and port numbers that looks something like this:

```
program vers proto   port
   100000    2   tcp    111  portmapper
   100000    2   udp    111  portmapper
   100004    2   udp    648  ypserv
   100004    2   tcp    649  ypserv
   100004    1   udp    648  ypserv
   100004    1   tcp    649  ypserv
   100007    2   tcp   1024  ypbind
   100007    2   udp   1026  ypbind
   100007    1   tcp   1024  ypbind
   100007    1   udp   1026  ypbind
   100029    1   udp    652  keyserv
   100028    1   tcp    658  ypupdated
   100028    1   udp    660  ypupdated
   100009    1   udp   1023  yppasswdd
   100035    1   udp    719
   100035    1   tcp    722
   100005    1   udp    724  mountd
                 .
                 .
                 .
```

If that works, you can also use rpcinfo o check if the mountd daemon is running:

```
% /usr/etc/rpcinfo -u server_name mount
```

This should come back with the response:

```
program 100005 version 1 ready and waiting
```

If these fail, try logging in to the server's console and see if it is okay.

If the server is up but your machine cannot communicate with it, you should check the Ethernet connections between your machine and the server.

If the server is okay and the network is okay, use `ps` to check your client daemons. You should have a `portmap` and several `biod` daemons running. For example, running `ps` should result in output similar to the following:

```
% ps ax
  PID TT STAT  TIME COMMAND
    0 ?  D     0:00 swapper
    1 ?  I     0:00 /single/init -
    2 ?  D     0:00 pagedaemon
   35 ?  I     0:11 portmap
   38 ?  I     0:00 ypbind
   41 ?  I     0:00 keyserv
   57 ?  S     9:27 in.routed
   59 ?  I     0:00  (biod)
   60 ?  I     0:00  (biod)
   61 ?  I     0:00  (biod)
   62 ?  I     0:00  (biod)
```

## Clearing Remote Mounting Problems

This section deals with problems related to mounting. If mount fails for any reason, check the sections below for specific details about what to do. They are arranged according to where they occur in the mounting sequence and are labeled with the error message you are likely to see.

`mount` can get its parameters explicitly from the command line or from `/etc/fstab`. The example below assumes command line arguments, but the same debugging techniques work if `/etc/fstab` is used in the `mount -a` command.

Keep in mind the interaction of the various players in the mount request. If you understand this, the problem descriptions below will make a lot more sense.

Consider the following `mount` request from a client on a network where YP is running:

```
% /usr/etc/mount dancer:/usr/src /dancer.src
```

Here are the steps `mount` uses to mount a remote file system.

Note This explanation is written

**Note** This explanation is written especially for advanced system administrators and programmers. If you do not have this type of experience, be aware that you do not have to fully understand the entire mount process to clear problems involving remote mounts.

1. `mount` opens `/etc/mtab` and checks that this mount has not already been done.
2. `mount` parses the first argument into host dancer and remote directory `/usr/src`.
3. `mount` calls the yellow pages binder daemon `ypbind` to determine which server machine to find the yellow pages server on. It then calls the `ypserv` daemon on that machine to get the Internet protocol (IP) address of dancers.
4. `mount` calls dancer's portmapper to get the port number of `mountd`.
5. `mount` calls dancer's `mountd` daemon and passes it `/usr/src`.
6. dancer's `mountd` reads its `/etc/exports` file and looks for the exported file system that contains `/usr/src`.
7. dancer's `mountd` daemon calls the yellow pages server `ypserv` to expand the host names and netgroups in the export list for `/usr/src`.
8. dancer's `mountd` does a `getfh`(2) system call on `/usr/src` to get the `fhandle`.
9. dancer's `mountd` returns the `fhandle`.
10. `mount` does an `mount`(2) system call with the `fhandle` and `/dancer.src`.
11. `mount` checks if the caller is superuser and if `/dancer.src` is a directory.
12. `mount` does a `statfs`(2) call to dancer's NFS server (`nfsd`).
13. `mount` opens `/etc/mtab` and adds an entry to the end.

## Error Messages Related to Remote Mounts

Any step in the remote mounting process can fail—some of them in more than one way. The sections below give detailed descriptions of the failures associated with specific error messages.

```
/etc/mtab: No such file or directory
```

The mounted file system table is kept in the file `/etc/mtab`. This file must exist before mount can succeed.

```
mount: ... already mounted
```

The file system that you are trying to mount is already mounted or there is a erroneous entry for it in `/etc/mtab`.

```
mount: ...  Block device required
```

You probably did not specify the remote machine name in the following command:

```
# /usr/etc/mount remote.machine:/usr/src /dancer.src
```

The mount command assumes you are doing a local mount unless it sees a colon in the file system name, or unless the /etc/fstab specifies that the file system is mounted via an nfs mount.

```
mount: ... not found in /etc/fstab
```

If you issue the mount command with only a directory or file system name but not both, it looks in /etc/fstab for an entry whose file system or directory field matches the argument. For example, the following command:

```
# /usr/etc/mount /dancer.src
```

searches /etc/fstab for a line that has a directory name field of /dancer.src. If it finds an entry, such as:

```
dancer:/usr/src /dancer.src nfs rw,hard 0 0
```

it will do the mount as if you had typed

```
# /usr/etc/mount -o rw,hard dancer:/usr/src /dancer.src
```

The default options are read-write, hard, and suid.

```
/etc/fstab: No such file or directory
```

This message indicates that mount tried to look up the directory given it as an argument in the /etc/fstab file, but could not find /etc/fstab.

```
 ... not in hosts database
```

On a network without YP, this message indicates that the host specified to mount is not in the /etc/hosts file. On a network running YP, the message indicates that YP could not find the host name in the /etc/hosts database or that the yellow pages daemon ypbind has died on your machine.

Check the spelling and the placement of the colon in your mount command. If the command is correct, your network does not run YP, and you only get this message for this host name, check the entry in /etc/hosts.

If your network is running YP, make sure that ypbind is running by typing:

```
# ps ax
```

Try to rlogin to another machine, or use rcp to remote copy something to another machine. If this also fails, your ypbind daemon is probably dead or hung. If you only get this message for this host name, you should check the /etc/hosts entry on the YP server. See Chapter 14 for more information about yellow pages problems.

```
mount: directory path must begin with '/'
```

The second argument to the mount is the path of the directory to be used as a mount point. This must be a full pathname starting at root (/).

```
mount: ... server not responding: RPC_PMAP_FAILURE - RPC_TIMED_OUT
```

Either the server you are trying to mount from is down, or its portmapper is dead or hung. Try rebooting the server to restart the inetd, portmap, and ypbind daemons. If you can't rlogin to the server but the server is up, you should check your Ethernet connection by trying to rlogin to some other machine. You should also check the server's Ethernet connection.

```
mount: ... server not responding: RPC_PROG_NOT_REGISTERED
```

This means that mount got through to the portmapper, but the NFS mount daemon rpc.mountd was not registered.

```
mount: ...: No such file or directory
```

Either the remote directory or the local directory does not exist. Check the spelling of the directory names. Try to use ls on both directories.

```
mount: not in export list for ...
```

Your machine name is not in the export list for the file system you want to mount from the server. You can get a list of the server's exported file systems by running

```
# showmount -e hostname
```

If the file system you want is not in the list, or your machine name or netgroup name is not in the user list for the file system, log in to the server and check the /etc/exports file for the correct file system entry. A file system name that appears in the /etc/exports file but not in the output from showmount, indicates a failure in mountd. Either it could not parse that line in the file, or it could not find the file system, or the file system name was not a local mounted file system. See the exports(5) man page for more information. If the /etc/exports file looks correct, and your network runs YP, check the server's ypbind daemon. It may be dead or hung.

```
mount: ...: Permission denied
```

This message is a generic indication that some authentication failed on the server. It may be that you are not in the export list (see above), that the server could not recognize your machine ypbind( is dead), or that the server does not believe you are who you say you are. Check the server's /etc/exports file, and, if applicable, ypbind. In this case you can just change your hostname with hostname(1) and retry the mount command.

```
mount: ...: Not a directory
```

Either the remote path or the local path is not a directory. Check the spelling in your command, and try to run on both directories.

```
mount: ...: Not owner
```

You have run mount as root on your machine because it affects the file system for the entire machine, not just you.

## Fixing Hung Programs

If programs hang doing file related work, your NFS server may be dead. You may see the following:

```
NFS server hostname not responding, still trying
```

on your console. The message indicates that NFS server *hostname* is down. This indicates a problem with your NFS server or with the Ethernet. Programs can also hang if a YP server dies.

If your machine hangs completely, check the server(s) from which you have mounted file systems. If one of them (or more) is down, do not be concerned. When the server comes back up, your programs continue automatically. No files are destroyed.

If a soft mounted server dies, other work should not be affected. Programs that time out trying to access soft mounted remote files will fail with errno ETIMEDOUT, but you should still be able to access your other file systems.

If all servers are running, go ask someone else using these same servers if they are having trouble. If more than one machine is having problems getting service, this indicates a problem with the server's nfsd daemons. Log in to the server; Run ps to see if nfsd is running and accumulating CPU time. If not, you may be able to kill and then restart nfsd. If this does not work, you will have to reboot the server.

If other systems seem to be up and running, check your Ethernet connection and the connection of the server.

### Fixing a Machine that Hung Part Way Through Boot

If your machine boots up normally until it tries to do remote mounts, probably one or more servers is down, or your network connection is bad. Try to reboot in single user mode. Then use the mount command to manually mount each directory normally mounted through the /etc/fstab file. See previous two subsections for more help.

### Speeding Up Slow Access Times

If access to remote files seems unusually slow, type:

```
# ps aux
```

on the server to be sure that it is not being adversely affected by a runaway daemon, bad tty line, etc. If the server seems okay and others are getting good response, make sure your biod daemons are running. Try the following steps:

1. Run ps ax and look for biod daemons in the display. If they are not running or are hung, continue with these steps.

2. Find the process IDs of the biod daemons by typing:

```
    # ps ax | grep biod
```

3. Kill these processes as follows:

```
    # kill -9 pid1 pid2 pid3 pid4
```

4. Restart them by typing:

```
    # /usr/etc/biod 4
```

   To determine if the biods are hung, run ps as above, then copy a large file from a remote system, then run ps again. If the biods do not accumulate CPU time, they are probably hung.

If the biods are okay, check your Ethernet connection. The command netstat-i introduced in Chapter 12 tells you if you are dropping packets. Also, you can use the commands nfsstat-c and nfsstat-s to tell if the client or server is doing a lot of retransmitting. A retransmission rate of 5 percent is considered high. Excessive retransmission usually indicates a bad Ethernet board, a bad Ethernet tap, a mismatch between board and tap, or a mismatch between your Ethernet board and the server's board.

## 13.5. Making the Network More Secure

This section explains how to implement network security through use of various administrative files. These files are

```
/etc/passwd
/etc/group
/etc/hosts.equiv
/.rhosts
```

For more information, refer to the man pages associated with these files, and to the *Security Features Guide.*

## The `/etc/passwd` File

This local password file contains entries for each user that has permission to log in to the machine to which the file applies. `/etc/passwd` is present on all categories of Sun computers, from diskless client to any type of server.

Here is a sample `passwd` file on a networked standalone machine:

```
root:uYU722Lg5xk/U:0:1:Operator:/:/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
uucp:eXs0qzRjUOS8Y:4:4::/usr/spool/uucppublic:
news:*:6:6::/usr/spool/news:/bin/csh
sync::1:1::/:/bin/sync
sysdiag::0:1:System Diagnostic:/usr/diag/sysdiag:/usr/diag/sysdiag/sysdiag
stefania:1Zm62edD0O8es:3747:20:Stephanie Brucker:/home/dancer/raks:/bin/csh
+::0:0:::
```

Each entry has the syntax:

*username:password:uid:gid:gcos-field:home-dir:login-shell*

The `passwd(5)` man page completely describes these parameters. Briefly, they are:

| | |
|---|---|
| *username* | The user's login name. |
| *password* | The user's encrypted password. |
| *uid* | User's numerical ID. |
| *gid* | Numerical ID of the group to which the user belongs. |
| *gcos-field* | User's real name and other identifying information to be printed in the user's mail message heading. |
| *home-dir* | Full pathname of the user's home directory. |
| *login-shell* | Shell the user accesses upon login. |

In the sample password file above, the users

```
root
nobody
daemon
bin
sync
```

are actually IDs created by SunOS for each machine. When you log in with the `root` user name, the system grants you special access rights to the administrative files in `/etc`. Depending on the circumstances, various SunOS programs also run with the user names `root`, `bin`, `nobody`, or `daemon` as their owners.

The user names

```
uucp:eXs0qzRjUOS8Y:4:4::/usr/spool/uucppublic:
news:*:6:6::/usr/spool/news:/bin/csh
```

in the sample above are used by uucp programs. The entry

```
stefania:1Zm62edD0O8es:3747:20:Stephanie Brucker:/home/dancer/stefania:/bin/csh
```

is the entry for a user allowed to log in to this machine.

### Setting Up a New User's Password

When you first add a new user, you have to create an entry for them in their machine's /etc/passwd file, then create a home directory for them. You should do the following:

1. Obtain basic information from the user, such as preferred user name (which should be unique within your network domain), full name as he or she wants it displayed in mail headers, and preferred login shell.

2. Become superuser and access the /etc directory of the person's machine, for example, raks.

```
#cd /export/root/raks/etc
```

3. Edit the local passwd file, using your preferred text editor.

4. Create an entry for the new user on a separate line.

5. Type the user's requested login name, for instance,

    **shamira**

    The colon after the user name is the delimiter used in the passwd file to indicate the end of a field.

6. Leave the password field blank by typing another colon, as in:

    **shamira::**

7. Add a user ID for shamira, according to your company's policies. Do not use a user ID that already exists in the local password file. Especially do not use 0 or 1, the user IDs for root and daemon. If your company does not have a policy for assigning user IDs, you have to create one. As one suggestion, some companies use a person's employee number in the user ID field of /etc/passwd. Follow the user ID with a colon.

8. Add a group ID number for the group you want the person to be in. If the user will not belong to a group, just type a colon to leave the field blank.

9. Type the person's name as he or she requested for the mail header. Follow it with a colon.

10. Type the full pathname of the person's home directory, which should be: /home/*server_name*/*user_name*. Follow it with the delimiting colon.

11. Finally, type the user's preferred login shell. Here is an example of a complete entry for new user shamira:

```
shamira::235:12:Marsha Wong:/home/dancer/shamira:/bin/csh
```

12. Close the /etc/passwd file. Now, create a home directory for the new user.

```
#cd /home/dancer
#mkdir shamira
# chown userid# shamira
```

13. Have the user log in to the client by supplying the new user name and then pressing RETURN when requested for a password.

14. Have the user run the passwd command to a password to the login name.

### The /etc/group File

The /etc/group file is a local file that you might want to alter for your server. The groups created consist only of users who are allowed to log in to your server.

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

The /etc/group file has several default groups. The wheel group, for instance, is the default group to which all local users belong. A second group that you might want to assign users to is operator. Members of the -operator group have special permissions that enable them to run certain commands without being superuser, and to run these commands on a remote machine without being in its .rhosts file. These commands are dump, restore, and shutdown. If your site employs certain individuals whose job it is to specifically run daily dumps and restores, you might want to add these people to the operator group.

For more information on groups, refer to the group(5) and chgrp(1) man pages.

### The /etc/hosts.equiv File

The /etc/hosts.equiv file is a list of machines, or *hosts*, whose users are permitted to remote login (rlogin) to your machine without supplying a password. It is a local file, pertaining only to your system, which you modify by logging in as superuser and using your text editor to make changes.

A typical hosts.equiv file has the following structure:

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

When you place a simple entry for a host in `hosts.equiv`, such as the entry above for `host1`, this means that anyone logging in to your machine from host1 is trusted. Furthermore, if the user logging in from host1 also has an entry in the `/etc/passwd` file for your machine, he or she will be granted immediate access and not requested for a password.

Here are some simple guidelines for understanding how the protections in `/etc/hosts.equiv` operate:

- If the user has an entry in the remote machine's `passwd` file and the user's machine is in the remote machine's `hosts.equiv` file, then the user can `rlogin` to the remote machine without a password.
- If the user is in the remote machine's `passwd` file but the user's machine is not in the remote machine's `hosts.equiv` file, then the user must supply a password when logging in to the remote machine.
- If the user does not have an entry in the remote machine's `passwd` file but the user's machine is in the remote machine's `host.equiv`, the user cannot `rlogin` in to the remote machine.
- If the user does not have an entry in the remote machine's `passwd` file, and there is no entry for the user's machine in the remote machine's `hosts.equiv` file, then the user cannot `rlogin` to the remote machine.

A single + on a line in your `hosts.equiv` file means that everyone logging in over the network is trusted. The `+@` in front of a group name means that all members of that group or netgroup are trusted. The `-@` before a group name means that no members of that group are to be trusted.

If your machine is a print server, its `hosts.equiv` file has to have real entries in it in order for it to print files from remote machines. The same is true if your machine is a mail server, as discussed in Chapter 15.

Refer to the `hosts.equiv`(5) man page for more information.

### The `/.rhosts` File

The `.rhosts` is located in the user's home directory. It has the same format as `/etc/hosts.equiv`, that is,

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

It is used for additional permission checking when trying to access a remote machine using `rlogin` or `rsh`.

When you use `rlogin` or `rsh`, the `.rhosts` file in the remote machine's home directory is added to the end of the `hosts.equiv` file for additional permission checking. If the `host.equiv` file denies you access due to a minus entry, but your machine is listed as trusted in `.rhosts`, then you will be able to access the remote machine. Additionally, if you are logged in as superuser on your machine and try to remote log in to another machine, only the `.rhosts` file of the remote machine is checked.

## How the Administrative Files Affect Network Security

Network security is implemented at two levels: first, at the machine level, and second, at the user level. The `/etc/hosts.equiv` and `/.rhosts` files, respectively, control access at these levels. If YP runs on your network, your machine consults the `hosts.equiv` and `yppasswd` files in the YP database on the YP server. If the machine does have a local copy of either file, the local copy is consulted first. The security-checking process goes like this:

- If you initiate a remote process on another machine, for example `rlogin`, the system first checks for an entry for your user name in `/etc/passwd` on the remote machine. If no entry is found and YP is running, the system then checks `yppasswd.` If the system still cannot find an entry, you are denied access. If attempting to `rlogin` to the machine, you will be prompted for a password and then get

```
Login incorrect
```

  You receive the same message if attempting an `rcp` or `rsh`.

- If an entry for your user name is found in the local `/etc/passwd`, the system next checks for your machine's hostname in the other machine's `/etc/hosts.equiv` file. If the hostname is found, you gain access.

- If no `/etc/hosts.equiv` entry is found, the system checks for a line with the your machine's hostname (and, optionally, your username) in the `.rhosts` file in the home directory on the machine you want to access. If the entry is found, you gain access.

- If no entry is found for your machine in either `/etc/hosts.equiv` or *user_name*/`.rhosts` , but an entry for your user name is in `/etc/passwd`, you can `rlogin` to the machine after giving the right password. However, you will get

```
Permission denied
```

  messages when attempting to run remote processes like `rcp` or `rsh`.

- The single exception to this security scenario is the superuser. When you `rlogin` as `root`, the system skips `/etc/hosts.equiv` (the second level check) and directly checks `.rhosts`.

To allow access to your machine by all users on another specific machine, do the following:

1. Include an entry in your machine's `/etc/passwd` file for each user on the remote machine.

2. Include the remote machine's hostname in your `/etc/hosts.equiv` file, or, if your network runs YP, make sure these are included in the appropriate

YP databases.

For example, suppose a machine's hostname is samba, and you want to allow anyone on host ballet to gain access to samba. Simply edit /etc/hosts.equiv on samba as follows. The file is just a list of hostnames, one per line:

```
raks
dancers
jazz
```

Add host ballet to the list:

```
raks
dancers
jazz
ballet
```

Now all users who can gain access to ballet can also freely rlogin to samba without having that machine request a password. Furthermore, users on ballet can rcp from and use rsh on samba, provided they are in samba's /etc/passwd file.

Suppose you want certain users on a particular machine to access your machine, but not everyone. Here you do not put the machine's hostname in your machine's /etc/hosts.equiv. Instead, put the machine's name in the .rhosts file in each user's home directory on your machine ( *user_name*/.rhosts ). Note that to avoid some security problems, this .rhosts file must be owned by either the home directory's owner or root, and may 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. Its format is best illustrated by an example. You can allow user chris at host samba to gain access to host raks, and keep other users of samba out by:

- Making sure samba is not in raks' /etc/hosts.equiv file.
- Adding an entry for chris on samba to ~chris/.rhosts. The entry looks like this:

```
samba chris
```

Note also that this means user chris can only remotely access raks from host samba. If he tries to remotely access raks from another host, he must supply a correct password to raks in order to rlogin and cannot complete remote processes. Of course, if he can rlogin, he can always modify his .rhosts file (if it is not owned by root, and you gave him a home directory on raks).

This brings up two points:

- The only way to achieve anything resembling security in a sensitive environment is to exclude users from the /etc/passwd file; once someone knows a password, he or she can access a machine. If your site requires tight security, also be especially careful to protect /.rhosts properly. Make sure only root has write permission. In addition, refer to the *Security Features Guide* for more information about security measures added with Release 4.0.

- If your site does not require stringent security measures, the easiest way to administer machine access at the user level is to give each "trusted" user an account in /etc/passwd and a home directory on your machine(s). Then ask the trusted users to create their own .rhosts files in their home directories on the machine(s).

Finally, note that an entry in /etc/hosts.equiv and .rhosts is invalid if it contains trailing white space— blanks or tabs. The presence of trailing white space may not be obvious. You can look at a file through the cat -e command or by running vi with the "set list" option. Both of these methods show a $ at the end of each line, before which you can look for trailing white space. Also try running grep on a file to search for tab_character$ or blank_space$ patterns to get a listing of any lines ending in tabs or blanks.

## Setting the User ID on Execution

When a program executes, the kernel changes the user ID of its owner to that of the person running it—unless the program's permissions have the suid (setuid) bit set. When the suid bit is set, the program 's user ID does not change when it is executed. This is an important security consideration. Many programs on your release tape come with the suid bit set, a condition you may want to modify. In addition, people creating programs may ask you to enable or disable this option for their own applications.

You set the user ID of a program by becoming superuser and running the chmod command. In the syntax statement

```
chmod [ -fR ] mode filename ...
```

specify setuid by substituting for *mode* the absolute mode 4*nnn*, where *nnn* are octal numbers representing the permissions for owner, group, and public. If you prefer, you can use the symbolic mode s with u and g to accomplish this. Then, if necessary, use the chown command to specify the login name that will always own the program, regardless of who runs it. Refer to chmod(1v) and chown(8) in the *SunOS Reference Manual* if you need help with these commands.

Here are the permissions for the /usr/bin/crontab program, produced by running ls -l on the directory /usr/bin.

```
-rwsr-xr-x  1 root        16384 Feb  5 12:42 crontab
```

Note the letter s listed as the execute permission bit for the program's owner. It indicates that the suid bit is set. Since the program's owner is root, crontab will always run with the root user ID as owner.

You should go through the permissions for the files installed from the release tape. Note any that are run suid, and note who is the owner. Some programs, such as daemons, always need to execute with root as user ID, regardless of who actually executed them. Depending on your site's security considerations, it is sometimes safer to change the ownership of other programs from root to daemon.

Shell scripts can prove dangerous security holes, if they run with root ownership and the suid bit on. This means that anyone can make changes to the script.

In SunOS Release 4.0, you can mount a directory from the server with the suid or nosuid option. For example, if you specify the command

```
# /usr/etc/mount -o /home/lauraf /home/lauraf
```

the home directory for lauraf will be mounted with the default options rw and suid. This is a wise precaution if you think the directory you are mounting might have laxly protected setuid programs. It prevents intruders from using these programs to gain access to your system.

You can mount a directory with the nosuid option, as in

```
# /usr/etc/mount -o rw,nosuid /home/lauraf /home/lauraf
```

If you enable this option, then any setuid programs in the mounted directory lose their setuid characteristics when executed on your machine. For example, a program such as /usr/etc/sendmail, which has the suid bit on and is owned by root, would run as if the user ID of its executor actually owned it, rather than root.

### Setting the Group ID

There is a difference in the way the group ID worked in previous releases and the way it works with Release 4.0. If the setgid bit on a directory is set, files created within that directory will be owned by the group ID of that directory. If the setgid bit is not set, files created in that directory will have the group ID of the creating process. This allows users to choose which type of group permission they want.

### Allowing Root Access over the Network

Under the NFS, a server exports file systems it owns so clients can remote mount them. When a client becomes superuser, it is denied access on remote mounted file systems. When a person logged in as root on one host requests access to a particular file from NFS, the user ID of the requester is changed to the user ID of the user name nobody. (Recall that nobody is one of the user names automatically created by SunOS in the default /etc/passwd file.) User nobody's access rights are the same as those given to the public for a particular file. For example, if the public only has execute permission for a file, then user nobody can only execute that file.

When you export a file system, you can permit root on a particular machine to have root access to that file system by editing /etc/exports on the server. For example, suppose you wanted the machine samba (but no others) to have superuser access to the exported directory /usr/src. You would enter the following line in /etc/exports.

```
/usr/src     -root=samba
```

If you want more than one client to have root access, you can specify a list as follows:

```
/usr/src     -root=samba:raks:jazz
```

You can also enable superuser access for all clients, again using the /etc/exports file. Suppose you wanted to allow superuser access for all clients accessing /usr/src. You add the following line to /etc/exports:

```
/usr/src     -anon=0
```

The anon is short for "anonymous." Its meaning is somewhat involved. An NFS server labels as anonymous any request from a root user (someone whose current user name has user ID 0) who is not in the root list in /etc/exports. Anonymous requests, by default, get their user ID changed from its previous value to -2, the user ID of nobody. The line in the sample /etc/exports:

```
/usr/src          -anon=0
```

tells SunOS to use the value 0 instead of -2 for anonymous requests. The result is that all root users retain their user ID of 0.

See exports(5) and exportfs(8) for more information on exporting directories.

## Privileged Ports

SunOS has some Internet domain source ports to which only privileged users can attach. These are known as *privileged ports*. Currently, NFS does not check to see if a client is bound to one of these. That is, an NFS server has no way of knowing whether a client's file request originated from the real client's kernel or from someone's user program. If you get the following error message:

```
NFS request from unprivileged port
```

you should also see the IP address of the client on console screen of the server. You can turn on server port checking by changing the following:

```
# adb -w /vmunix
nfs_portmon?W0
_nfs_portmon:    0x1          =          0x0
<Control-D>
```

The next time you boot your system, the source ports will be checked. If you can trust all of the root users on your network, then just doing the above is enough. But be warned: some non-UNIX systems do not enforce the privileged port convention (in particular, PCs with 3Com boards). Another warning: all of a server's clients should be running software release 3.0 or higher for this to work. (A server will reject all pre-3.0 requests.)

## The Lock Manager

Sun release 4.0 now supports a System V compatible record and file locking service. The lock manager functionality applies to local record and file locking as well as to record and file locking of remote file systems on the network. File locking in accomplished with the `lockf` and `fcntl` functions. Since the Lock Manager is an extensible serve, the user, program, or application can optionally invoke the service on a per-transaction, or per-application basis. Once a record is locked by one user or application, another user or application trying to lock the same record will either (a) receive an error message or (b) be placed in a queue and be appropriately notified when the lock becomes available.

The *Lock Manager* is a System V Interface Definition (SVID) compatible record and file locking service. When multiple users attempt to access the same record at the same time, only one will have access and the rest will be locked out.

System V compatible file and record locking `fcntl(` calls) supports both local and remote files. A new signal has been added to allow recovery in the event that a remote server crashes and locks are lost. Ordinarily, locks will be reclaimed when the remote server recovers and SIGLOST (the signal) will not be issued. The signal notifies the process in the event of a lost lock. The default action is to kill the process, and therefore existing System V programs need not be changed to run on SunOS, unless they choose to recognize this failure notification.

See the *SunOS Reference Manual* for more information.

## Clock Skew in User Programs

Because NFS architecture differs in some minor ways from earlier versions of the UNIX operating system, you should be aware of places where users programs could run up against these incompatibilities.

Because each workstation keeps its own time, the clocks are out of sync between the NFS server and client. Obviously this might introduce a problem in certain situations. The clock skew problems have been fixed. Here are examples of two problems and how they were fixed.

1. Many programs make the (reasonable) assumption that an existing file could not have been created in the future. For example, `ls` does this. The command `ls -l` has two basic forms of output, depending upon how old the file is:

```
# date
Jan 22 15:27:01 PST 1985
# touch file2
# ls -l file*
-rw-r--r--  1 root         0 Dec 27  1983 file
-rw-r--r--  1 root         0 Jan 22 15:27 file2
```

The first form of `ls` prints the year, month, and day of last file modification if the file is more than six months old. The second form prints the month, day, and minute of last file modification if the file is less than six months old.

`ls` calculates the age of a file by simply subtracting the modification time of the file from the current time. If the results are greater than six months worth of seconds, the file is considered old.

Now assume that the time on the server is Jan 22 15:30:31 (three minutes ahead of the local machine's time):

```
# date
Jan 22 15:27:31 PST 1985
# touch file3
# ls -l file*
-rw-r--r--  1 root          0 Dec 27  1983 file
-rw-r--r--  1 root          0 Jan 22 15:26 file2
-rw-r--r--  1 root          0 Jan 22  1985 file3
```

The problem is that the difference of the two times is huge:

> (now) - (modification_time) =
> (now) - (now + 180 seconds) =
> -180 seconds = huge unsigned number,
> which is greater than six months.

Thus, ls believes the new file was created long ago in the past. Sun modified ls to deal with files which are created a short time in the future.

2. The ranlib program was also modified to deal with clock skew. ranlib timestamps a library when it is produced, and ld compares that timestamp with the last modified date of the library. If the last modified date occurred after the timestamp, then ld instructs you to run andranlibagain,

   If the library is on a server whose clock is ahead of the client running ranlib, ld will always complain. Sun fixed ranlib to set the timestamp to the maximum value of the current time and the library's modify time.

Remember that if your application depends upon local time or the file system timestamps, then it will have to deal with clock skew problems if it uses remote files.

# 14

# The Sun Yellow Pages Service

# 14

# The Sun Yellow Pages Service

Yellow Pages (YP) is Sun's distributed network lookup service. It maintains a set of files that machines can query as they would a database. These files are known as YP maps. All YP maps are fully replicated on several systems known as YP servers, each of which runs a server process for the maps. It does not matter which server process answers a client request. The response is always the same because the set of YP maps is identical for each server. This allows you to have multiple servers per network and makes YP service highly reliable and available.

## 14.1. Basic YP Concepts

This section gives an overview of YP, its maps, and its associated commands.

### The YP Map

Each YP map contains a set of keys and associated values. For example, in a map called `hosts.byname`, all the host names within a domain are the keys, and the Internet addresses of these host names are the values. Each YP map has a mapname used by programs to access it. Many of the current YP maps are derived from ASCII files traditionally found in `/etc`, such as:

```
/etc/hosts
/etc/group
/etc/passwd
```

and a few others. The format of the data within the YP map is identical (in most cases) to the format within the ASCII file. dbm(3) files, which are located in the subdirectories of the directory `/var/yp` on YP servers, implement the YP maps.

A *YP domain* is a named set of YP maps, located in a subdirectory of `/var/yp`, the directory where a YP server holds all its maps. The name of this subdirectory is the name of the YP domain. For example, maps for the `literature` domain are located in `/var/yp/literature`. Refer to Chapters 3 and 12 for suggestions for setting up domain names.

Each machine belongs to a default domain set at boot time by the `/etc/rc.local` file with the `domainname` command. You must set the domain name on all machines (both servers and clients) that will access the maps of a particular domain. You can also display or change your YP domain name by using the `domainname` command.

In the YP environment, only a YP servers have a set of YP maps, which they make available to clients over the network. There are two kinds of YP servers: a

YP slave server and a YP master server. The master server updates the maps of the slave servers. Therefore, you should always modify maps on the YP master server. The changes propagate from the master server to the YP slave servers. If you create or change YP maps on a slave server instead of a master server, this may create two potentially different versions of the YP map. Furthermore, the YP master server is the only machine with all the necessary ASCII files.

The YP clients run processes that request data from maps on these machines. YP clients do not care which server is the master, since all YP servers should have the same information. The distinction between master and slave server only applies to where you make the updates.

A server may be a master with regard to one map, and a slave with regard to another. Randomly assigning maps to YP servers can cause a great deal of confusion. You are strongly urged to make a single server the master for all the maps you create by the `ypinit` command within a single domain. The examples in this chapter assume that one server is the master for all maps in the domain.

### Commands Used for Maintaining YP

YP offers a series of commands that you can use for setting up and editing maps and performing other YP-related functions. This subsection briefly describes them. These commands are described in detail later on in this section, in the *SunOS Reference Manual* and in the *Network Programming* manual.

`ypserv` — Describes the processes that comprise the YP service. These are `ypserv`, the YP map server daemon and `ypbind`, the YP binder daemon. `ypserv` must run on each YP server. `ypbind` must run on all clients.

`ypfiles` — Describes the file structure of the YP service.

`ypinit` — Automatically constructs maps from files located in `/etc`, such as `/etc/hosts`, `/etc/passwd`, and others. `ypinit` also constructs initial versions of required maps that are not built from files in `/etc`, for example, `ypservers`. Use `ypinit` to set up the master YP server and the slave YP servers for the first time. You typically do not use it as an administrative tool for running systems.

`ypmake` — Describes the use of `/var/yp/Makefile`, which builds several commonly-changed components of YP maps. These are the maps built from the files in `/etc` on the master YP server: `passwd`, `hosts`, `group`, `netgroup`, `networks`, `protocols`, and `services`.

`makedbm` — Takes an input file and converts it into a pair of `dbm` files, which then become valid YP maps. For example, `ypmaps.dir` and `ypmaps.pag` are both `dbm` files. You can use `makedbm` to build or rebuild maps not built from `/var/yp/Makefile`. You can also use makedbm to "disassemble" a map, so that you can see the key-value pairs that comprise it. You can also edit the disassembled form using editors such as `vi`, `emacs`, and `ex`, or text processing tools like `awk`, or`grep`, `cat`. The disasssembled form is in the format required for input back into `makedbm`.

`ypxfr` Moves a YP map from one YP server to another, using YP itself as the transport medium. You can run `ypxfr` interactively, or periodically from a `crontab` file. (See Chapter 8 for information about running `crontab`.)

`yppush` Requests each of the `ypserv` processes within a domain to transfer a particular map, waits for a summary response from the transfer agent, and prints out the results for each server. You run it on the master YP server.

`ypset` Tells a `ypbind` process (the local one, by default) to get YP services for a domain from a named YP server. This is not for casual use.

`yppoll` Asks any `ypserv` for the information it holds internally about a single map.

`ypcat` Displays the contents of a YP map. Use it when you do not care which server's version you are seeing. If you need to see a particular server's map, `rlogin` to that server (or use `rsh`) and use `makedbm`.

`ypmatch` Prints the value for one or more specified keys in a YP map. Again, you have no control over which YP server's version of the map you are seeing.

`ypwhich` Use this command to see which YP server a host is using at the moment for YP services, or which YP server is master of a particular map.

`ypupdated` Daemon used for changing YP information. This daemon is normally started up by `inetd`. `ypupdated` consults the file `updaters` in the `/var/yp` directory to determine which maps should be updated and how to change them. Note that `ypupdated` only works if the network is running secure RPC.

### How Administrative Files Are Consulted on a YP Network

YP can serve any number of maps. Typically these include some files in `/etc`. YP services make updating these files much simpler, since you do not have to make the same change to every machine on the network. For example, on networks that do not run YP, programs read the `/etc/hosts` file to find an Internet address. When you add a new machine, you have to add an entry for this machine to the `/etc/hosts` files on every machine on the network. On networks running YP, programs that need to consult `/etc/hosts` now do a remote procedure call to the YP servers for the same information.

SunOS programs do not consult the same system administrative on a network with YP than they would on a network without YP. This is because they consult YP maps instead. The following list describes how SunOS programs on a network running YP consult the administrative files you have previously learned about.

`/etc/passwd` Always consulted. If there are + or – entries, the YP password map is consulted, otherwise YP is not used. See `passwd`(5).

`/etc/group` Always consulted. If there are + or – entries, the YP group map is consulted, otherwise YP is not used. See `group`(5.

`/etc/services` Never consulted. The data that was formerly read from this file now comes from the YP services map.

`/etc/protocols` Never consulted. The data that was formerly read from this file now comes from the YP protocols map.

`/etc/networks` Never consulted. Data is taken from this file to create the YP networks map.

`/etc/netgroup` Never consulted. The data that was formerly read from this file now comes from the YP netgroup map.

`/etc/bootparams` Never consulted. The data that was formerly read from this file now comes from the YP bootparams map.

`/etc/ethers` Never consulted. The data read from this file comes from the YP netgroup map.

`/etc/hosts` Consulted only when booting (by the `ifconfig` command in the `/etc/rc.boot` file). After that the YP map is used instead.

`/etc/hosts.equiv`

(And similarly for `.rhosts`) Always consulted, though neither of these files is in the YP domain. (See the section below *How Security Is Changed with YP*, for a fuller explanation of these two files.) If there are + or – entries, whose arguments are netgroups, the YP netgroup map is consulted, otherwise YP is not used. See `hosts.equiv`(5).

`/etc/aliases` Always consulted. Local aliases take precedence over those in the YP database. See `/etc/aliases`.

`/etc/netmasks` Never consulted. The data that was formerly read from this file now comes from the YP netmasks map. See the man page `netmasks`(5).

## YP Installation and Administration

This section covers nine installation and administration topics:

1. Setting up a master YP server
2. Altering a YP client's database to use YP services
3. Setting up a slave YP server
4. Setting up a YP client
5. Modifying individual YP maps after installing YP
6. Propagating a YP map
7. Making new YP maps after installing YP
8. Adding an additional YP server
9. Changing the master server to a different machine

### How to Set up a Master YP Server

Before setting up the master YP server, there are several steps you must take. You need to set up the YP domain name, if it is to differ from the than the name selected for your network domain during `suninstall`. You also need to set up the hostname. By default, `/etc/rc.local` sets up `domainname` and `/etc/rc.boot` sets up `hostname`.

To create a new server on an existing network, you got to the `/var/yp` directory and run `/var/yp/ypinit`. You are asked whether you want the procedure to die at the first non-fatal error (in which case, you can fix the problem and restart `ypinit`, recommended if you haven't done the procedure before), or to continue despite non-fatal errors. In this second case you can try to fix all the problems by hand, or fix some, then restart `ypinit`. `ypinit` prompts you for a list of other hosts that will also be YP servers. (Initially, this is the set of YP slave servers, but at some future time any of them might become the YP master server.) You need not add any other hosts at this time, but if you know that you will be setting up more YP servers, add them now. You will save yourself some work later, and there is little runtime penalty for doing it. (However, do not name every host in the network.)

Before running `ypinit`, the following files in `/etc` should be complete and reflect an up-to-date picture of your system: `passwd`, `hosts`, `ethers`, `group`, `networks`, `protocols`, and `services`. Also, if you know how `/etc/netgroup` is going to be set up, do that before running `ypinit`. If you don't know, `ypinit` makes an empty netgroup map. Also, `/etc/aliases` should be complete.

For security reasons, you may restrict access to the master YP machine to a smaller set of users than that defined by the complete `/etc/passwd` file. To do this, copy the complete file to some place other than `/etc/passwd`, and edit out undesired users from the remaining `/etc/passwd`. For a security-conscious system, this smaller file should not include the YP escape entry discussed in the next section.

After performing these steps, you are ready to create a new master server. Become superuser and change directory to `/var/yp`. Then run `ypinit` with the -m option.

To start providing YP services, invoke /usr/etc/ypserv. It then starts up automatically from /etc/rc.local every time the server boots.

### Altering a YP Client's Files To Use YP Services

Once you decide to run YP at your site, you should have all hosts on the network access the YP maps, rather than potentially out-of-date information in their local administrative files. That policy is enforced by running a ypbind process on the client machine (including machines that may be running YP servers), and by abbreviating or eliminating the files that traditionally implemented the YP maps. The files in question are:

```
/etc/passwd
/etc/hosts
/etc/ethers
/etc/group
/etc/networks
/etc/protocols
/etc/services
/etc/netgroup
/etc/aliases
/etc/netmasks
/.rhosts
```

The treatment of each file is discussed in this section.

- /etc/networks, /etc/protocols, /etc/ethers, /etc/services, and /etc/netgroup need not exist on any YP clients.

- /etc/hosts.equiv is never served by YP. However, you can add escape sequences to reference YP. This reduces problems with rlogin or which are sometimes caused by different /etc/hosts.equiv files on the two machines.

  To let anyone log on to a machine, you can edit /etc/hosts.equiv to contain a single line, with only the character, + (plus) on it. A line with only a + means that all further entries are retrieved from YP rather than the local file.

  Alternatively, you can exercise more control over logins by using lines of the form:

```
+@trusted_group1
+@trusted_group2
-@distrusted_group
```

  Each of the names to the right of the at sign (@) is assumed to be a netgroup name, defined in the global netgroup database. The netgroup database is served by YP.

If none of the escape sequences is used, only the entries in /etc/hosts.equiv are used; YP is not used.

- /.rhosts also is never served by YP. As shown in Chapter 13, its format is identical to that of /etc/hosts.equiv. However, because this file controls remote root access to the local machine, unrestricted access to it is not recommended. Make the list of trusted hosts explicit, or use netgroup names for the same purpose. You can not use secondary hostnames in your .rhosts, hosts.equiv, or netgroup files. You can, however, use secondary hostnames in /etc/hosts. All of the above files are related in that they enable local machines to access remote machines in some fashion.

- /etc/hosts must contain entries for the local host's name, and the local loopback name. These are accessed at boot time when the YP service is not yet available. After the system is running, and after the ypbind process is up, the /etc/hosts file is not accessed at all. An example of the hosts file for YP client raks' is:

```
127.1        localhost
192.9.1.87   raks     # Stefania
```

- /etc/passwd should contain entries for the root user name and the primary users of the machine, and the + escape entry to force the use of the YP service. A few additional entries are recommended: daemon, to allow file-transfer utilities to work; and operator, to let a dump operator log in. A sample YP client's /etc/passwd file looks like:

```
root:9wxntql2tHT.k:0:1:Operator:/:/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
uucp:*:4:4::/var/spool/uucppublic:
news:*:6:6::/var/spool/news:/bin/csh
sync::1:1::/:/bin/sync
raks:7kjDXZD/Hug2s:624:20:Stefania:/home/dancer/raks:/bin/csh
+::0:0:::
```

The last line informs the library routines to use the YP service. If you remove the last line in the passwd file, you will disable YP password access.

A program that calls /etc/passwd first looks in the password file on your machine; it will then look in the YP password file only if your machine's password file contains + (plus sign) entries, as shown in the above example. Also, earlier entries in the file take precedence over, or mask later ones with the same user name, or the same user ID. Therefore, please note the order of the entries for daemon and for sync (which have the same user ID) and duplicate it in your own file.

- /etc/group may be reduced to a single line:

```
+:
```

which forces all translation of group names and group IDs to be made via the YP service. This is the recommended procedure.

## How To Set Up a Slave YP Server

The network must be working to set up a slave YP server — in particular, you must be able to rcp files from the master YP server to YP slaves.

To create a new slave server, change directory to /var/yp. From there run ypinit with the -s option. You must be superuser when you run ypinit. Name a host already set up as a YP server as the master. Ideally, the named host really is the master server, but it can be any host that has its YP database set up. The host must be reachable. The default domain name on the machine intended to be the YP slave server must be set up, and must be set to the same domain name as the default domain name on the machine named as the master. Also, an entry for daemon must exist in the /etc/passwd files of both slave and master, and that entry must precede any other entries which have the same user ID. suninstall creates /etc/passwd; make sure your password file has not been altered to change the order. Note the example shown in the section above. You won't be prompted for a list of other servers, but you will have the opportunity to choose whether or not the procedure gives up at the first non-fatal error.

After running ypinit, make copies of /etc/passwd, /etc/hosts, /etc/group, /etc/networks, /etc/protocols, /etc/netgroup, and /etc/services. For instance on a machine named ypslave:

```
ypslave% cp /etc/passwd /etc/passwd-
```

Edit the original files in accordance with the preceding section "Altering a YP Client's Files To Use YP Services“ to insure that processes on the slave YP server actually use the YP services, rather than the local ASCII files. (That is, make sure the YP slave server is also a YP client) Make backup copies of the edited files, as well. For instance:

```
ypslave% cp /etc/passwd /etc/passwd+
```

After the YP database gets set up by ypinit, type /usr/etc/ypserv to begin supplying YP services. On subsequent reboots, it will start automatically from /etc/rc.local.

## How To Set Up a YP Client

To set up a YP client, edit the local files as described above in "Altering a YP Client's Files to Use YP Services." If /usr/etc/ypbind is not running already, start it. With the ASCII databases of /etc abbreviated and /usr/etc/ypbind running, the processes on the machine will be clients of the YP services. At this point, there must be a YP server available; processes will hang if no YP server is available while ypbind is running. Note the possible alterations to the client's /etc database as discussed above in the section on altering the client. Because some files may not be there, or some may be specially altered, it is not always obvious how the ASCII databases are being used. The escape conventions used within those files to force data to be included or excluded from the YP databases are found in the following man pages: passwd, hosts, netgroup, hosts.equiv, group. In particular, notice that changing passwords in /etc/passwd (by editing the file, or by running passwd, only affects the local client's environment. Change the YP password database by running yppasswd.

To add a new client to a network already running YP, you use the setup_client program described in Chapter 13. You use the same syntax as described in this chapter. Be sure to specify **client** for *yp_type*.

## How To Modify Existing YP Maps After YP Installation

Always change databases served by YP *on the master server.* The databases expected to change most frequently, like /etc/passwd, may be changed by first editing the ASCII file, changing directory to /var/yp, and running make.

Non-standard maps (that is, maps that are specific to the applications of a particular vendor or site, but are not part of Sun's release), or maps that are expected to change rarely, or maps for which no ASCII form exists (for example, maps not around before you installed YP), may be modified manually. The general procedure is to use makedbm with the -u option to disassemble them into a form which can be modified using standard tools (such as awk, sed, or vi). Then build a new version again using makedbm. This may be done by hand in two ways:

- The output of makedbm can be redirected to a temporary file that can be modified, then fed back into makedbm, or
- The output of makedbm can be operated on within a pipeline that feeds into makedbm again directly. This is appropriate if the disassembled map can be updated by modifying it with awk, sed, or a cat append, for instance.

Suppose you want to create a non-standard YP map, called mymap. You want it to consist of key-value pairs in which the keys are strings like al, bl, cl, etc. (the l is for left), and the values are ar, br, cr (the "r" is for right). There are two possible procedures to follow when creating new maps. One uses an existing ASCII file as input; the other uses standard input.

For example, suppose there is an existing ASCII file named /var/yp/mymap.asc, created with an editor or a shell script on the machine ypmaster. home_domain is the subdirectory where the map is located. You can create the YP map for this file by typing:

```
ypmaster%  cd /var/yp
ypmaster%  makedbm mymap.asc home_domain/mymap
```

But at this point you notice the map really should have included another 2-tuple: (dl, dr). You can make the modification simply. In all situations like this, remember to modify the map by first modifying the ASCII file. Modifications made to the dbm files, not also in the ASCII file, will be lost. Make the modification like this:

```
ypmaster%  cd /var/yp
ypmaster%  <make editorial change to mymap.asc>
ypmaster%  makedbm mymap.asc home_domain/mymap
```

When there is no original ASCII file, create the YP map from the keyboard by typing input like this (assume the machine name is `ypmaster`, and the default domain is `home_domain`):

```
ypmaster% cd /var/yp
ypmaster% makedbm - home_domain/mymap
al ar
bl br
cl cr
<ctl D>
```

When you need to modify that map, you can use `makedbm` to create a temporary ASCII intermediate file which can be edited using standard tools. For instance:

```
ypmaster%  cd /var/yp
ypmaster%  makedbm -u home_domain/mymap > mymap.temp
```

At this point you can edit `mymap.temp` to contain the correct information. A new version of the database is created by the commands

```
ypmaster%  makedbm mymap.temp home_domain/mymap
ypmaster%  rm mymap.temp
```

The preceding paragraphs explained how to use some tools, but in reality almost everything you actually have to do can be done by `ypinit` and `/var/yp/Makefile`, unless you add non-standard maps to the database, or change the set of YP servers after the system is already up and running.

Whether you use the `Makefile` in `/var/yp` or some other procedure `Makefile`— is one of many possible— the goal is the same: a new pair of well-formed dbm files must end up in the domain directory on the master YP server.

## Propagation of a YP Map

To propagate a map means to move it from place to place — in general, to move it from the master YP server to all slave YP servers. Initially, `ypinit` moves it, as described above in the section "How To Set Up A Slave YP Server." After a slave YP server has been initialized, updated maps are transferred from the master server by `ypxfr`. You can run `ypxfr` in three different ways: periodically by `cron`; by `ypserv`; and interactively by a user. Following are examples of each.

Maps have differing rates of change; for instance "protocols.byname" may not change for months at a time, but "passwd.byname" may change several times a day in a large organization. You can set up entries in an `crontab` file to periodically run `ypxfr` at a rate appropriate for any map in your YP database. `ypxfr` contacts the master server and transfers the map only if the master's copy is more recent than the local copy.

To avoid having to have a crontab entry for each map, several maps with approximately the same change characteristics can be grouped in a shell script, and the shell script can be run from a /crontab file. In 4.0, note that now instead of directly editing the crontab file, you must edit a copy of the file and use the crontab command to give the new crontab file to the cron daemon.

Suggested groupings, mnemonically named, can be found in /var/yp: ypxfr_1perhour, ypxfr_1perday, and ypxfr_2perday. If the rates of change are inappropriate for your environment, you can easily modify or replace these shell scripts.

These same shell scripts should be run at each YP slave server in the domain. Alter the exact time of execution from one server to another, so as to prevent the checking from bogging down the master. If you want the map transferred from some particular server, not the master, you can specify that (using ypxfr's -h option) within the shell script. You can use the -s option to transfer maps from another domain. Finally, maps having unique change characteristics can be checked and transferred by explicit invocations of Lypxfr within the system crontab file, /var/spool/cron/crontabs/root.

ypxfr also gets invoked by ypserv, responding to a "Transfer Map" request. That request is made as an RPC call from yppush. yppush is run on the master YP server. It enumerates the YP map "ypserver" to generate a list of YP servers in your domain. To each of the named YP servers, it sends a "Transfer Map" request. ypserv forks off a copy of ypxfr, invoking it with the -C flag, and passing it the information it needs to identify the map, and to call back the initiating yppush process with a summary status.

Release 3.0 and later ypserv also kicks off ypxfr when it attempts to emulate the behavior of ypserv from an earlier release, for instance when a 2.x master ypserv communicates directly with the 3.0 ypserv.

In the cases mentioned above, ypxfr's transfer attempts and the results can be captured in a log file. If /var/yp/ypxfr.log exists, results are appended to it. No attempt to limit the log file is made. To turn off logging, remove the log file.

Finally, you can run ypxfr as a command. Typically, you do this only in exceptional situations — for example when setting up a temporary YP server to create a test environment, or when trying to quickly get a YP server that has been out of service consistent with the other servers.

## How to Make New YP Maps After YP Installation

Adding a new YP map entails getting copies of the map's dbm files into the domain directory on each of the YP servers in the domain. The actual mechanism has been described above in "Propagation of a YP Map". This section only describes how to get the proper files in place so the propagation works correctly. Things must be set up correctly on both the master and the slaves.

After deciding which YP server is the master of the map, modify /var/yp/Makefile on the master server so that the map can be conveniently rebuilt. Actual case-by-case modification is too varied to describe here, but typically a human-readable ASCII file is filtered through awk, sed, and/or grep to make it suitable for input to makedbm. Consult the existing Makefile as a source for programming examples. You should use the mechanisms already in place in /var/yp/Makefile when deciding how to create dependencies that make will recognize; specifically, the use of .time files allows you to see when the Makefile was last run for the map.

To get an initial copy of the map, you can run yppush on the YP master server. The map must be globally available before clients begin to access it. If the map is available from some YP servers, but not all, you will see unpredictable behavior from client programs.

## How To Add a New YP Server Not in the Original Set

To add a new YP slave server, you start by modifying some maps on the master YP server. If the new server is a host that has not been a YP server before, you must add the host's name to the map "ypservers" in the default domain. The sequence for adding a server named "ypslave" to *domain_name* is:

```
ypmaster# cd /var/ypdomain_name
ypmaster# ../makedbm -u ypservers > /tmp/temp_file
ypmaster# vi /tmp/temp_file
ypmaster# ../makedbm /tmp/temp_file ypservers
```

Running the makedbm command with the -u option undoes the *ypservers* dbm file; that is, it converts it from dbm format temporarily so you can add the new hostname to the temporary file *temp_file*. Then running the makedbm command with *temp_file* as the input file and ypservers as the output file converts ypservers back into dbm format.

The new slave YP server's databases can then be set up by copying the databases from YP master server ypmaster. To do this, remote log in to the new YP slave, and run the ypinit command:

```
ypslave# cd /var/yp
ypslave# ypinit -s ypmaster
```

To verify that the the ypservers file is correct (since there is no ASCII file for ypservers) do the following:

```
ypslave# cd /var/yp/domain_name
ypslave# ../makedbm -u ypservers
```

Then complete the steps described above in the section "How To Set Up A Slave YP Server".

**Note:** If a host name is not in *ypservers* it will not be warned of updates to the YP map files.

## How To Change the Master Server

To change a map's master, first build the map at the new master. Because the old YP master's name occurs as a key-value pair in the existing map, it is not sufficient to use an existing copy at the new master, or to send a copy there with ypxfr. The key must be reassociated with the new master's name. If the map has an ASCII source file, it should be present in its current version at the new master. Remake the YP map (called jokes.bypunchline below) locally with the sequence:

```
newmaster# cd /var/yp
newmaster# make jokes.bypunchline
```

/var/yp/Makefile must be set up correctly for this to work. If it isn't, do it now. Also, this is a good time to go back to the old master (if it will remain a YP server) and edit /var/yp/Makefile so that jokes.bypunchline is no longer made there — that is, comment out the section of oldmaster:/var/yp/Makefile which made jokes.bypunchline.

If the map only exists as a dbm database, you can remake it on the new master by disassembling an existing copy (one from any YP server will do) and running the disassembled version back through makedbm. For example:

```
newmaster# cd /var/yp
newmaster# ypcat -k jokes.bypunchline |\
makedbm - mydomain/jokes.bypunchline
```

After making the map on the new master, you must send a new copy of the map to the other (slave) YP servers. However, don't use yppush— the other slaves will try to get new copies from the old master, rather than the new one.

A typical method (you may find others) is to become superuser on the old master server and type:

```
oldmaster# /etc/yp/ypxfr -h newmaster jokes.bypunchline
```

Now you have a new copy on the old master, and now you may run yppush. The remaining slave servers still believe that the old master is the current master, and will attempt to get the current version of the map from the old master. When they do so, they will get the new map, which names the new master as the current master.

If the method above fails, there is a cumbersome but sure-fire option. On each YP server machine, while superuser, execute the command shown just above. This will certainly work, but should be considered the worst case solution.

## Debugging a YP Client

This debugging section has two parts — first those problems seen on a YP client, and then those problems seen on a YP server.

Before trying to debug a YP client, read the earlier section in this chapter on how the YP works.

### On Client: Commands Hang

The most common problem at a YP client node is for a command to hang and generate console messages which say:

```
yp: server not responding for domain <wigwam>.  Still trying
```

Sometimes many commands begin to hang, even though the system as a whole seems okay and you can run new commands.

The message above indicates that ypbind on the local machine is unable to communicate with ypserv in the domain wigwam. This often happens when machines that run ypserv have crashed. It may also occur if the network or the YP server machine is so overloaded that ypserv cannot get a response back to your ypbind within the timeout period. Under these circumstances, all the other YP client nodes on your net show the same or similar problems. The condition is temporary in most cases, and the messages will usually go away when the YP server machine reboots and ypserv gets back in business; or when the load on the YP server nodes and/or the Ethernet decreases.

However, in the circumstances described below, the situation will never get better.

- The YP client has not set, or has incorrectly set, domainname on the machine. Clients must use a domain name that the YP servers know. Use domainname to see the client domain name. Compare that with the domain name set on the YP servers. The domain name should be set in /etc/rc.local. When /etc/rc.local fails to set or incorrectly sets domainname, do the following: become superuser on the machine in question, edit /etc/rc.local to fix the domainname line with a proper domain name (this assures domain name will be correct every time the machine boots), and set domainname manually so it is fixed immediately. To do this, type:

```
# domainname good_domain_name
```

- If your domain name is correct, make sure your local network has at least one YP server machine. You can automatically bind only to a ypserv process on your local network, not on another accessible network. There must be at least one YP server for your machine's domain running on your local network. Two or more YP servers will improve availability and response characteristics for YP services.

- If your local network has a YP server, make sure it is up and running. Check other machines on your local net. If several client machines have problems simultaneously, suspect a server problem. Find a client machine behaving normally, and try the ypwhich command. If ypwhich never returns an answer, kill it and go to a terminal on the YP server machine. Type:

```
# ps ax | grep yp
```

  and look for ypserv and ypbind processes. If the server's ypbind daemon is not running, start it up by typing:

```
# /usr/etc/ypbind
```

  If there is a ypserv process running, do a ypwhich on the YP server machine. If ypwhich returns no answer, ypserv has probably hung and should be restarted. Kill the existing ypserv process (you must be logged on as root), and start /usr/etc/ypserv:

```
# kill -9 [some pid # from ps]
# /usr/etc/ypserv
```

If ps shows no ypserv process running, start one up.

### On Client: YP Service Unavailable

When other machines on the network appear to be okay, but YP service becomes unavailable on your machine, many different symptoms may show up. Among them: some commands appear to operate correctly while others terminate printing an error message about the unavailability of YP; some commands limp along in a backup-strategy mode particular to the program involved; and some commands or daemons crash with obscure messages or no message at all. For example, things like the following may show up:

```
my_machine% ypcat myfile
ypcat: can't bind to YP server for domain <wigwam>.
    Reason: can't communicate with ypbind.
```

```
my_machine% /usr/etc/yp/yppoll myfile
Sorry, I can't make use of the yellow pages.  I give up.
```

When symptoms like those above occur, try

```
my_machine% ls -l
```

on a directory containing files owned by many users, including users not in the local machine's /etc/passwd file, for example /usr. If the ls -l reports file owners not in the local machine's /etc/passwd file as numbers, rather than names, it is one more symptom that YP service is not working.

These symptoms usually indicate that your ypbind process is not running. You can do a ps ax to check for one. If it you do not find it, type:

```
my_machine# /usr/etc/ypbind
```

to start it. YP problems should disappear.

### On Client: ypbind Crashes

If ypbind crashes almost immediately each time it is started, you should look for a problem in some other part of the system. Check for the presence of the portmap daemon by typing:

```
my_machine%  ps ax | grep portmap
```

If you don't find it running, reboot.

If portmap itself will not stay up or behaves strangely, look for more fundamental problems. Check the network software in the ways suggested in the section on *Ethernet Debugging* in Chapter 12, "The SunOS Network Environment."

You may be able to talk to the portmap on your machine from a machine operating normally. From such a machine, type:

```
rigel% rpcinfo -p client
```

If your portmap is okay, the output should look like:

```
program vers proto   port
  100007    2   tcp   1024  ypbind
  100007    2   udp   1028  ypbind
  100007    1   tcp   1024  ypbind
  100007    1   udp   1028  ypbind
  100021    1   tcp   1026  nlockmgr
  100024    1   udp   1052  status
  100021    1   udp   1054  nlockmgr
  100024    1   tcp   1027  status
  100020    1   udp   1058  llockmgr
  100020    1   tcp   1028  llockmgr
  100021    2   tcp   1029  nlockmgr
  100012    1   udp   1083  sprayd
  100011    1   udp   1085  rquotad
  100005    1   udp   1087  mountd
  100008    1   udp   1089  walld
  100002    1   udp   1091  rusersd
  100002    2   udp   1091  rusersd
  100001    1   udp   1094  rstatd
  100001    2   udp   1094  rstatd
  100001    3   udp   1094  rstatd
  100015    6   udp   1365  selection_svc
```

On your machine the port numbers will be different. The four entries that represent the ypbind process are:

```
  100007    2   tcp   1024  ypbind
  100007    2   udp   1028  ypbind
  100007    1   tcp   1024  ypbind
  100007    1   udp   1028  ypbind
```

If they are not there, ypbind has been unable to register its services. Reboot the machine. If they are there and they change each time you try to restart /usr/etc/ypbind, reboot the system, even if the portmap is up. If the situation persists after reboot, call for help.

On Client: ypwhich Inconsistent

When you use ypwhich several times at the same client node, the answer you get back varies — the YP server changes. This is normal. The binding of YP client to YP server changes over time on a busy net, and when the YP servers are busy. Whenever possible, the system stabilizes at a point where all clients get acceptable response time from the YP servers. As long as your client machine gets YP service, it doesn't matter where the service comes from. Often a YP server machine gets its own YP services from another YP server on the net.

## Debugging a YP Server

Before trying to debug a YP server, read the earlier section in this chapter on how YP works.

### On Server: Different Versions of a YP Map

Because YP works by propagating maps among servers, you will sometimes find different versions of a map at servers on the network. This version skew is normal if transient, and abnormal otherwise.

Most commonly, normal update is prevented when some YP server or some router between YP servers is down during a map transfer attempt. (Normal update is described in the section above, *Propagation Of A YP Map*). When all the YP servers, and all the routers between them, are up and running, ypxfr should succeed.

If a particular slave server has problems updating, log in to that server and run ypxfr interactively. If ypxfr fails, it will tell you why it failed, and you can fix the problem. If ypxfr succeeds, but you think it has been failing sometimes, create a log file to enable logging of messages. Type:

```
ypslave# cd /var/yp
ypslave# touch ypxfr.log
```

This saves all output from ypxfr. The output looks much like what ypxfr creates when run interactively, but each line in the log file is timestamped. (You may see unusual orderings in the timestamps. That's okay— the timestamp tells you when ypxfr began its work. If copies of ypxfr ran simultaneously, but their work took differing amounts of time, they may actually write their summary status line to the log files in an order different from the order in which they were invoked.) Any pattern of intermittent failure shows up in the log. When you have fixed the problem, turn off logging by removing the log file. If you forget to remove it, it will grow without limit.

While still logged in to the problem YP slave server, inspect the system crontab file, /var/spool/cron/crontabs/root, and the ypxfr* shell scripts it invokes. Typos in these files cause propagation problems, as do failures to refer to a shell script within /var/spool/cron/crontabs/root, or failures to refer to a map within any shell script.

Also, make sure that the YP slave server is in the map ypservers within the domain. If it's not, it will still work fine as a server, but yppush won't tell it when a new copy of a map exists.

If the problem is not obvious, you can work around it while you debug using rcp(1) or tftp(1) to copy a recent version from any healthy YP server. You may not be able to do this as root, but you can probably do it as daemon. For instance, to transfer map 'busted':

```
ypslave# chmod go+w /var/yp/mydomain
ypslave# su daemon
$ rcp ypmaster:/var/yp/mydomain/busted.\* /var/yp/mydomain
$ ^D
ypslave# chown root /var/yp/mydomain/busted.*
ypslave# chmod go-w /var/yp/mydomain
```

Notice that the * character has been escaped in the command line, so that it will be expanded on ypmaster, instead of locally on ypslave. Also, notice that the map files should be owned by root, so you must change ownership of them after the transfer. Obviously, if you can do the rcp as root, it makes the whole thing easier.

### On Server: ypserv Crashes

When the ypserv process crashes almost immediately, and won't stay up even with repeated activations, the debug process is virtually identical to that described above in the section l“On Client: ypbind Crashes.”. Check for the portmap daemon:

```
ypserver% ps ax | grep portmap
```

Reboot the server if you do not find the daemon. If it is there, type:

```
rigel% rpcinfo -p speed
```

and look for output to the screen like:

```
 program vers proto   port
  100004    2   udp   1027  ypserv
  100004    2   tcp   1024  ypserv
  100004    1   udp   1027  ypserv
  100004    1   tcp   1024  ypserv
  100007    2   tcp   1025  ypbind
  100007    2   udp   1035  ypbind
  100007    1   tcp   1025  ypbind
  100007    1   udp   1035  ypbind
  100009    1   udp   1023  yppasswdd
  100003    2   udp   2049  nfs
  100024    1   udp   1074  status
  100024    1   tcp   1031  status
  100021    1   tcp   1032  nlockmgr
  100021    1   udp   1079  nlockmgr
  100020    1   udp   1082  llockmgr
  100020    1   tcp   1033  llockmgr
  100021    2   tcp   1034  nlockmgr
  100012    1   udp   1104  sprayd
  100011    1   udp   1106  rquotad
  100005    1   udp   1108  mountd
  100008    1   udp   1110  walld
  100002    1   udp   1112  rusersd
  100002    2   udp   1112  rusersd
  100001    1   udp   1115  rstatd
  100001    2   udp   1115  rstatd
  100001    3   udp   1115  rstatd
```

On your machine, the port numbers will be different. The four entries that represent the ypserv process are:

```
  100004    2   udp   1027  ypserv
  100004    2   tcp   1024  ypserv
  100004    1   udp   1027  ypserv
  100004    1   tcp   1024  ypserv
```

If they are not there, ypserv has been unable to register its services. Reboot the machine. If they are there, and they change each time you try to restart /usr/etc/ypserv, reboot the machine. If the situation persists after reboot, call for help.

## How Security Is Changed with YP

Security on a system running YP is dependent on how YP consults the administrative files on which its maps are based. Local files on the host are consulted first, then the system consults maps in the YP domain. For example, the system checks the /etc/aliases file for mail aliases, then checks the mail.aliases YP map.

The remaining files on which YP maps are based are global files: /etc/hosts, /etc/networks, /etc/ethers, /etc/services, /etc/netmasks, /etc/protocols, and /etc/netgroup. The information in these files is network wide data, and is accessed only from YP. However, when booting, each machine needs an entry in /etc/hosts for itself. In summary, if YP is running, global files are only checked in the YP maps; a file on your local machine is not consulted.

### Special YP Password Change

When you change your password with the passwd command, you change the entry explicitly given in your machine's local /etc/passwd file. If your password is not given explicitly, but rather is pulled in from YP with a + entry, then the passwd command will print the error message

```
Not in passwd file.
```

To change your password in the YP password file, you must use the yppasswd command. To enable this service, you must start up the daemon yppasswdd server on the machine serving as the master for the YP password file.

### /etc/publickey

To enable users to use the "secure" option to mount a directory the YP database publickey must exist. publickey is the public key database used for secure networking. Each entry in the database consists of a network user name (which may either refer to a user or a hostname), followed by the user's public key (in hexadecimal notation), a colon, and then the user's secret key encrypted with its login password (also in hexadecimal notation).

This file is altered either by the user through the chkey(1) command or by using the newkey(8) command. The file /etc/publickey should only contain data on the YP master machine, where it is converted into the YP database publickey.byname. Also see ypupdated(8c), newkey(8), chkey(1), and getpublickey(3r).

There are two new YP databases that enhance system security called publickey and netid. You use the newkey program to put new entries into this map. Give the following command:

```
# cd /var/yp
# make publickey
```

to create the YP database from the /etc/publickey file. Do not use a text editor to alter the /etc/publickey file because the file contains encryption keys. To alter the /etc/publickey file, you need to use the newkey command. There are two options to this command:

```
# newkey -u username
```

for a regular user on a host machine, and

```
# newkey -h hostname
```

for a root user on a host machine. The publickey database contains every user that has a public key. They are identified by a long string.

You should also be aware that there is a new YP database called netid; however, you do not need to administer it. The netid database is created from the passwd, host, and group files.

### Netgroups: Network Wide Groups of Machines and Users

YP uses the /etc/netgroup file on the master YP server for permission checking during remote mount, login, remote login, and remote shell. It uses /etc/netgroup to generate three YP maps in the /var/yp/*domainname* directory: Lnetgroup, netgroup.byuser and netgroup.byhost. The YP map netgroup contains the basic information in /etc/netgroup. The two other YP maps contain a more specific form of the information to speed up the lookup of netgroups given the host or user.

The programs that consult these YP maps are login, mountd, rlogin, and rsh. login consults them for user classifications if it encounters netgroup names in /etc/passwd. mountd consults them for machine classifications if it encounters netgroup names in /etc/exports. rlogin and rsh consult the netgroup map for both machine and user classifications if they encounter netgroup names in the /etc/hosts.equiv or /.rhosts file.

Refer to Chapter 12 for more information about etc/netgroup.

### What If You Do Not Use YP?

If ypserv on the master is disabled, you can no longer update any of the YP maps.

If you choose not to use YP, you can disable it by simply renaming the /usr/etc/ypbindfile to /usr/etc/ypbind.orig. (This is what suninstall does automatically if you tell it you do not want to run YP.)

```
% cd /etc
% mv /usr/etc/ypbind /usr/etc/ypbind.orig
```

To disable YP on a particular YP slave or master, do the following:

```
% mv /usr/etc/ypserv /usr/etc/ypserv.orig.
```

suninstall does this automatically if you do not select yp.

Refer back to Chapter 13, "The Sun Network File System," for information about the files you need to maintain for a server should you decide to disable YP.

## 14.2. Adding a New User to a YP Server

To add a new user, add an entry to the password file and create a home directory on the new user's machine as described in the steps below.

To add a new user to the YP servers, first update the yppasswd file. Log in as root on the master YP server machine. First edit the master YP server's /etc/passwd file. Later the password file entry for the user will be copied to the /etc/passwd in the new client's / directory; without an entry in the local /etc/passwd, the person administering the new client machine cannot log in should YP fail.

On the master YP server add a new line to the password file with the `vipw` command; `vipw` brings the password file into the `vi` editor, and prevents anyone else from editing it until you are done:

```
# /usr/etc/vipw
```

The following example shows an entry in the password file for new user Mr. Chimp with an account called bonzo:

```
bonzo:3u0mRdrJ4tEVs:1947:10:Mr. Chimp:/usr/myserver/bonzo:/bin/csh
```

Here is how the fields in the password file work when used in a network running YP:

Login name
Synonymous with user name.

Encrypted password
The user's password. Tell all new users how to add, or change, their password with the `yppasswd` command. Remember, you can make this field empty when a user has forgotten his or her password, thereby enabling them to log in without a password until such time as a new one is given. Note that an asterisk in this field matches no password. The user can only log in if the name of the machine logged in from is in the `/etc/hosts.equiv` file on the local machine or if this user has this machine name in the `/.rhosts` file.

User ID
A number unique to this user. A system knows the user by ID number associated with login name. Therefore a login name must have the same user ID number on all password files of machines that are networked in a local domain. Failure to keep ID's unique prevents files on different machines from being moved between directories because the system will respond as if the directories are owned by two different users. Also, file ownership may become confused when an NFS server exports a directory to an NFS client whose password file contains users with user IDs that match those of different users on the NFS server.

Group ID
Can be used to group users together who are working on similar projects.

User Information
Information about the user— usually real name, phone number, etc. An ampersand (&) here is shorthand for the user's login name.

Home Directory
The directory the user logs in to, usually `/home/`*server_name*`/`*user_name*.

Login Shell
Initial shell to use on login. If this field is blank the default `/bin/sh` is used. It is recommended that you

place /bin/csh here, as in the example above, to give the user the C shell for a login shell.

After you have updated the password file and created a password for the new user, be sure to update the YP maps by changing directories to /var/yp and running the make command.

```
# cd /var/yp
# make passwd
```

Note that the C2 secure configuration option will change the passwd file: it will be split into two files. In C2 secure environment, only processes running with superuser privileges will be able to access the file containing the encrypted password.

### Make a Home Directory

After adding a new entry to the password file, you should create a home directory for the new user to log in to. This will be the same as the directory given in the sixth field of the password file entry. In the /usr/*myserver* directory, make a directory for the new user and change ownership to the user's login name and change group to the user's group. For example:

```
# cd /usr/myserver
# mkdir bonzo
# chown bonzo bonzo
# chgrp 10 bonzo
```

Note that if the YP maps for the password file have not yet been updated on the machine's YP server, you will get the following error message when you attempt to do the chown:

```
unknown user id:  username
```

In that case, you can use the following set of commands:

```
# cd /usr/servername
# mkdir bonzo
# chown userid# bonzo
# chgrp 10 bonzo
```

You use Mr. Chimp's user ID number (from the password file entry) instead of login name to change the ownership of his home directory.

# 15

# Electronic Mail and Communications

# 15

# Electronic Mail and Communications

## 15.1. Setting Up the Mail Routing System

SunOS electronic mail is handled by `sendmail`, a general internetwork mail routing service. `sendmail` features aliasing and forwarding, automatic routing to network gateways, and flexible configuration.

This section explains how to install the mail system on your computer. For a more detailed explanation, refer to Chapter 18, "Sendmail Installation and Operation."

The mail system consists of the following commands and files:

| | |
|---|---|
| `/usr/ucb/mail` | UCB mail program, described in `mail`(1) |
| `/usr/lib/sendmail` | Mail routing program |
| `/etc/sendmail.cf` | Configuration file for mail routing |
| `/etc/sendmail.main.cf` | Configuration file for main machines (see below) |
| `/usr/bin/mailtool` | Window-based interface to `/usr/ucb/mail`. |
| `/usr/lib/etc/sendmail.subsidiary.cf` | Configuration file for subsidiary machines (see below) |
| `/var/spool/mail` | Mail spooling directory for delivered mail |
| `/var/spool/mqueue` | Spool directory for mail going out over the network |
| `/var/spool/secretmail` | Secure mail directory |
| `/usr/bin/xsend` | Secure mail sender |
| `/usr/bin/xget` | Secure mail receiver |
| `/usr/bin/enroll` | To receive secure mail messages |
| `/etc/aliases` | Mail forwarding information |
| `/usr/ucb/newaliases` | Symbolic link to `/usr/lib/sendmail`. |

| | |
|---|---|
| `/usr/ucb/biff` | Mail notification enabler |
| `/usr/etc/in.comsat` | Mail notification daemon |
| `/usr/etc/in.syslog` | Error message logger, used by `sendmail` |

Mail is normally sent and received using `mail(1)` or `mailtool`, which provides a user interface to edit the messages sent and received, and passes the messages to `sendmail(8)` for routing. `sendmail` uses knowledge of the network name syntax, aliasing and forwarding information, and network topology, as defined in the configuration file `/etc/sendmail.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 `/var/spool/mail`, using a locking protocol to avoid problems with simultaneous updates. After the mail is delivered, the local mail notification daemon `/usr/etc/in.comsat` notifies users who have issued a `biff y` command that mail has arrived.

Mail for *username* is normally accessible in the file `/var/spool/mail/`*username*. You are the only person who can read your mail file; however, anyone with the superuser password can read others' files, including their mail. To send mail that is secure against any possible perusal (except by a code-breaker), use the secret mail facility, which encrypts the mail so that no one can read it. See `xsend(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 Chapter 18.

## Picking a Name for your Domain

The network mail delivery program, `sendmail`, uses Internet standard mail addressing formats that 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.

In mail addressing, domains nest inside one another like directories, except that the names go from right to left. Thus "joe.sun.com" is host joe in subdomain "sun," which is in domain "uucp," just like `/etc/hosts.equiv` is file `hosts.equiv` in directory `etc`. To make life easier for the people who maintain mailers, there are only a small number of top-level domains like "uucp" and "arpa."

Ideally, domain names should be unique within the world and registered with the Network Information Center (NIC) at SRI International. For example, Sun's domain, "Sun.COM," is registered. Sun can divide this domain into subdomains as need arises. You can also contact the UUCP Project at AT&T in Columbus, Ohio for registering names on the UUCP network. Refer to Chapter 12, "The SunOS Network Environment," for more information about contacting NIC.

Some valid top-level domains are:

| | |
|---|---|
| .COM | Commercial companies |
| .EDU | Educational Institutions |

| | |
|---|---|
| .GOV | Government Agencies |

Upper and lowercase is not significant in domain names. The SunOS and UNIX convention is to use lowercase; many other systems use uppercase.

## 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 to phone lines, you can make it a main machine. Pick a machine of this type on the network. Treat the rest as subsidiary machines for configuration purposes. Note that if you have a non-networked standalone, treat it like the main machine in a one machine network.

If your machine is on an Ethernet with an ARPANET host, but the network does not have phone lines, you can pick any workstation as the main machine. If your ARPANET host runs `sendmail`, you can make it your main machine. All other Suns will be subsidiary machines.

Similarly, if you have several machines on an Ethernet and none have phones, pick one as the main machine and leave the others as subsidiaries.

A main machine should be aliased as `mailhost` in `/etc/hosts`.

### Configuring for Mail Forwarding

Next, you must define each machine's mailer configuration. Log in as superuser and type the following to configure a main machine:

```
# cp /usr/lib/sendmail.main.cf /etc/sendmail.cf
```

To configure a subsidiary, type:

```
# cp /usr/lib/sendmail.subsidiary.cf /etc/sendmail.cf
```

### Telling `sendmail` Your Domain Name

To tell `sendmail` your domain name, edit the file `/etc/sendmail.cf` on the main machine and all subsidiary machines. Near the top of the file is a block of lines that look like the following:

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

This defines the Sun domain name as "sun" within the "COM" domain: in other words, "sun.COM." Changes these lines to reflect your site. For example,

```
# local domain names
DDdance
CDdance
```

defines your domain name as "dance.COM." The hostname of your main machine (as set up by the `hostname` command) is dance, and subsidiary machines, if you have any, will be called, for example, "raks.dance.COM" for a machine with hostname raks.

If your top-level domain is not COM, find the lines that look like:

```
# domain-spec for local domain within universe
DDcuCOM
```

They should be the next lines in the file. Change COM to your top-level domain name, for example, DUedu.

If you are editing `sendmail.cf` for a subsidiary machine with phone lines, you can have `sendmail` route `uucp` mail to certain hosts via the local phone lines, rather than having all `uucp` traffic go through the main machine. To do this, look for the following lines in `sendmail.cf`:

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

Put the names of the local `uucp` sites on the end of the CV line, or on additional CV lines. For example, you could do the following:

```
CV rome prussia georgia
```

This completes `sendmail.cf` editing for the subsidiary machine. Note that if you have more than one subsidiary with no local `uucp` connections, you can edit the file just once, then copy it to all the subsidiary machines with `rcp`. If your server and all clients can share the subsidiary `sendmail.cf`, then put the file in `/usr/lib` or use symbolic links.

## ARPANET Forwarding

On your main machine, you can make several optional changes. If a machine with which you have a `uucp` or Ethernet connection is on the ARPANET and will relay mail to you, look for the following block of lines in `sendmail.cf`:

```
# major relay mailer
DMether

# major relay host
DRmailhost
CRmailhost
```

Edit the mailer that you connect to this host with (`uucp` or `ether`) and the name of the relay host. For example, if you share an Ethernet with a machine called "cmu-cs-vlsi" that is on the ARPANET, your entry might look like the following:

```
# 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, which means your entry may resemble the following:

```
# major relay mailer
DMether

#major relay host
DRucbvax
CRucbvax
```

This change enables you to mail to address such as “charlie@MIT.EDU.” Even though you are not on the ARPANET, the message will arrive. Do not worry about this if you do not have an ARPANET relay point.

### Setting Up the Postmaster Alias

Edit the /etc/aliases file and look for an entry for postmaster. This is the machine where people will send mail to the person at your site who handles mailer problems. (You can send mail to postmaster at other network sites if you have problems with mail originating at those sites.) The line will look like this:

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

If you administer the mail system, change the name root to your user name, so that messages directed to the postmaster of your machine arrive with your other mail. If you also manage the entire domain, add the alias to the domain wide aliases.

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 “amina” on machine raks, the line in /etc/aliases should read

```
Postmaster: amina@raks
```

You can also create aliases for people or groups of people called *mailing lists* at this time. See the examples in the /etc/aliases file. You can edit this file at any time. Each time you edit /etc/aliases, run the newaliases program to rebuild the local alias database. Run make in /etc/yp on the master YP server after updating domain wide aliases.

By default, any time a message is returned as undeliverable by sendmail, a copy of the message header is sent to the postmaster. Should you prefer to, you can disable the feature by editing the sendmail.cf file. Look for the following lines:

```
# CC my postmaster on error replies I generate
OPPostmaster
```

and change them as shown:

```
    # CC my postmaster on error replies I generate
    OPnobody
```

### Yellow Pages Requirements

Finally, you need to edit /etc/hosts on the YP master server, so that the main machine (raks in this example) will also be known by the alias mailhost to enable mail relay. For example, the /etc/hosts file for domain dance might look like the following:

```
192.9.1.1     raks
192.9.1.93    samba
192.9.1.23    ballet
192.9.1.30    jazz
```

with three subsidiary machines. On the YP server in raks' YP domain, change the /etc/hosts file to look like:

```
192.9.1.1     raks mailhost
192.9.1.93    samba
192.9.1.23    ballet
192.9.1.30    jazz
```

Store domain wide aliases in /etc/aliases on the YP master server. You can now use the map mail.aliases for domain-wide mail aliases.

Since YP map mail.aliases can also be used for domain-wide mail aliases, you should not have to remember hostnames when sending mail. mail.aliases will usually contain a copy of all the aliases known to a central mail machine.

To complete the mailer configuration process, type the following on the master YP server:

```
    # cd /usr/etc/yp
    # make
```

After you have completed this process, remember to configure the individual machines.

### Testing Your Mailer Configuration

First, reboot all systems whose configuration files you have changed. Then, send test messages from various machines on the network. To do so, go to a particular machine and type the following:

```
    samba% /usr/lib/sendmail -v </dev/null adddresses
```

This command sends a null message to the specified address, and displays messages while it runs. Try the following tests:

- □ Send mail to yourself or other people on the local machine by addressing the message in the command above to a regular user name (root, for example).

- □ If you are on an Ethernet, send mail to someone on another machine (`root@jazz,` 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 more than two are on the network.

- □ If you have set up a `uucp` connection on your phone line to another host, you can send mail to someone at that host and they can send mail back, or call you on the phone if they receive it.

  Try having them send mail to you. For example, you could send to ucbvax!azhar if you have a connection to ucbvax. `sendmail` will not be able to tell you whether the message really got through, since it hands the message to `uucp` for delivery. You have to ask the human at the other end if mail has been received. To get an idea of the message's progress, look in the file `/etc/spool/uucp/LOGFILE`. For more information, refer to Chapter 21.

- □ Send a message to "postmaster" on various machines and make sure that it comes to your usual mailbox, so that when other sites send you mail as postmaster, you will see it.

## Diagnosing Problems with Mail Delivery

Here are some tools you can use for diagnosing mail problems.

- □ `Received` lines in the header of the message.

  These trace which systems the message was relayed through. Note that in the `uucp` network, many sites do not update these lines, and that in the 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 account for different time zones.

- □ Messages from " MAILER-DAEMON."

  These typically report delivery problems.

- □ The system log, for delivery problems in your group of workstations.

  `sendmail` always records what it is doing in the system log, as explained in the next subsection. You might want to modify the `crontab` file to run a shell script nightly that searches the log for SYSERR messages and mails any that it finds to postmaster. In this way, problems are often fixed before anyone notices them, and the mail system runs more smoothly.

- □ You can use the `mconnect` program to open connections to other `sendmail` systems over the network.

## The System Log

The system log is supported by the `syslog` program, which is fully described in the man page `syslog`(8).

### Format

Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the Ethernet), and a message.

### Levels

`syslog` can log a large amount of information. The log is arranged as succession of levels. At the lowest level, only unusual occurrences are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered "useful;" log levels above ten are usually for debugging purposes.

## 15.2. Dial-up Networks

### An Overview Of `uucp`

Before trying to set up `uucp`, install your modem using the instructions in Chapter 11, "Adding Hardware to Your System."

`uucp` (UNIX to UNIX copy) is a series of programs designed for communication, via dial-up or hardwired lines, between two systems running UNIX. You can use `uucp` to transfer files between UNIX systems, and also to run commands on remote machines. For more detailed background, see Chapter 21, "UUCP Implementation Description."

Support for `uucp` is located in four major directories: `/usr/bin` (which contains user commands), `/usr/lib/uucp` (operational commands), `/etc/uucp` (per-machine configuration files), and `/usr/spool/uucp` (spooling area).

The `uucp` commands are:

| | | |
|---|---|---|
| `/bin/rmail` | receive remote mail | `rmail`(8) |
| `/usr/bin/rnews` | receive remote news | |
| `/usr/bin/uucp` | file-copy command | `uucp`(1C) |
| `/usr/bin/uux` | remote execution command | `uux`(1C) |
| `/usr/bin/uusend` | binary file transfer using mail | `uusend`(1C) |
| `/usr/bin/uuencode` | `uusend` binary file encoder | `uuencode`(1C) |
| `/usr/bin/uudecode` | `uusend` binary file decoder | `uudecode`(1C) |
| `/usr/bin/uulog` | scans session log files | `uucp`(1C) |
| `/usr/bin/uuname` | who do you talk to? | `uuname`(1C) |
| `/usr/bin/uustat` | uucp status — gives error messages | `uustat` |

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

| | |
|---|---|
| /etc/uucp/L-devices | list of dialers and hardwired lines |
| /etc/uucp/L-dialcodes | dialcode abbreviations |
| /etc/uucp/L.cmds | commands remote sites may execute |
| /etc/uucp/L.sys | systems to communicate with and how to connect |
| /etc/uucp/SEQF | sequence numbering control file |
| /etc/uucp/USERFILE | remote site pathname access specifications |
| /usr/lib/uucp/uucico | uucp protocol daemon |
| /etc/uucp/uucp.day | script for daily polling/cleanup |
| /etc/uucp/uucp.hour | script for hourly polling |
| /etc/uucp/uucp.night | script for nightly polling |
| /etc/uucp/uucp.noon | script for midday polling |
| /etc/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, /etc/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 would have /usr/spool/uucp/C.res45n0031; but in the current version the file is /usr/spool/uucp/C./C.res45n0031.

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 uuclean 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; maintain these files with uulog. 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'; people at other sites send to it when sending files to users at your site.

uucp occasionally sends mail about minor problems to "uucp" or "root." You can redirect these mail messages to your mailbox with an entry for 'uucp' in /etc/aliases. Look at the examples there.

Under normal conditions, uucp calls your designated sites at specified times and, in addition, checks to see if any files are queued on that site's machine for you. If you ever need to invoke uucp 'on command', the line:

```
# /usr/lib/uucp/uupoll
```

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 have trouble with the connection, run uucp with the debugging option, as described in installation step 7, below.

## Installing uucp

A uucp network link using modems or dedicated lines may be established between two machines running UNIX. 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 have modems which both call and answer, like Ven-Tel'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 Chapter 21; it describes in detail the file formats and conventions, and will give you necessary context.

- The name of the host machine in /etc/rc.boot will be the name of your uucp connection to the outside world (make sure it is fewer than 8 letters). We will call this machine your 'uucp host'; its name is your 'uucp hostname'.

  If this machine is your 'main machine' (for a definition see the next section, *Setting Up the Mail System*), then your uucp hostname should be the same as your domain name.

- Change the /usr/spool/uucp/D.noname directory to your own site's /usr/spool/uucp/D.*hostname* directory with the following:

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

  Use your uucp hostname for *hostname.*

- Create a uucp account for each remote host in the password file on your uucp host machine. Use /etc/vipw to enter a line of the following form in /etc/passwd:

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

  Note, make this change only on the machine handling uucp traffic, not on the YP master server. Now use the passwd command to establish a password for each remote host:

  ```
  # passwd Uhostname
  ```

  Tell this password to the administrator of the remote host, who must then

[1] * 'suid' stands for 'set user id;' if you set this bit on an executable file, the system 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 writable no matter who invokes the uucp program.

incorporate it into the `L.sys` file on that machine.

- The `L.sys` file contains the phone numbers and login sequences required to establish a connection with a `uucp` daemon on another machine. Edit `/etc/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:***password*

  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 to connect through an auto-call unit, and finally a login sequence ending with a password. 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 Sun supports three currently manufactured modems: The Ven-Tel 1200-31, the Ven-Tel 1200-32, and the Hayes Smartmodem 1200. Sun will continue to support the Ven-Tel MD 212-4. Indicating only `ACU` causes the `uucp` daemon, `uucico`, to search for any available auto-call unit in `L-devices`. For example, Sun's `L.sys` file looks something like:

```
adiron Any ACUHAYES 1200 7620183 login:-EOT-login: uucp ssword: secret
ucbvax Any,20 ACUVENTEL 1200 6728212% login:-EOT-login: uucp ssword: almama
ucbarpa Any,20 ACUVENTEL 1200 4539351 login:-EOT-login: uucp ssword: netnut
decvax Any ACUVENTEL 1200 6139949241 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
```

- Connect your auto-dial/auto-answer modem to ttya (the port labeled 'SIO-A' on the backpanel of your Sun Workstation) on your host machine. There is a complete discussion in *Adding A Modem To Your System* in Chapter 11.
- To set up your modem for both dial-in and dial-out on the same serial port, the flags bit corresponding to the serial port has to be set to zero in the kernel. To find out how to do this, read the *Kernel Modification* section of *Adding A Modem To Your System.*
- Create the appropriate device for your modem with the following series of commands:

```
# cd /dev
# mknod cua0 c 12 128
# chmod 600 cua0
# chown uucp cua0
# mv ttya ttyd0
```

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

```
# kill -1 1
```

□ In some older releases, uuxqt may fail because it cannot open files due to permission problems. Check permissions on the following files (note that *systemname* below should be fewer than 7 characters):

```
/usr/spool/uucp/C.
/usr/spool/uucp/D.
/usr/spool/uucp/D.systemname
```

If they are not rwx--x--x (711), type the following:

```
# chmod 711 /usr/spool/uucp/{C.,D.,D.systemname}
```

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

```
# tip 5551234
```

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

□ As a final test, run uucp with the debug option (–x), as follows:

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

With –x, the higher the number, the more debugging output you get; 1, 4, and 7 are reasonable choices. If you get substantial quantities of output from this command, everything is fine; you can go on to edit the files below.

□ Add the uucp.day, uucp.noon, uucp.night, uucp.hour, and uucp.week, files to /etc/crontab (see cron(8)). These arrange for the appropriate sites to be polled at the appropriate times. For example, the entries in crontab might look like:

```
5 6 * * * su uucp < /etc/uucp/uucp.day
15 12 * * * su uucp < /etc/uucp/uucp.noon
30 23 * * * su uucp < /etc/uucp/uucp.night
10,30,50 * * * * su uucp < /etc/uucp/uucp.hour
0 7 * * 2 su uucp < /etc/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 it poll you regularly (once a day is a reasonable minimum). When you send mail to another site, you must wait for it to call you. To handle installation for a passive node, just complete steps one through five and step seven in the procedures above. When you come to the fourth step, editing /etc/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.

# Part Four: Administrator's Reference

This part contains reference material that supplements the information given in Parts One, Two, and Three. Much of the text is taken from tutorials published by the University of California at Berkeley. Where applicable, references are cited.

### Who Should Read This Part

Since this material is very technical in nature, it is recommended for experienced system administrators who want to tailor their systems to suit special needs, and those with programming experience who want to know more about the programs used to administer a Sun workstation.

### What Is In This Part

Part Four contains reference material about the following subjects:

- Adding your own device drivers to the kernel configuration file
- The `fsck` utility.
- The `sendmail` routing program
- `printcap` and the print spooling programs.
- Customizing a terminal using `termcap`.
- The `uucp` wide area network
- Creating a name server

# 16

# Advanced Kernel Reconfiguration

# 16

# Advanced Kernel Reconfiguration

This chapter explains how to add customized features to your kernel configuration file. Topics covered are:

- Adding device drivers that you have written into the kernel configuration file.
- Configuring multiple kernel images
- Configuring systems without source code
- Sharing object modules
- Building profiled kernels.
- Tuning IPC System Parameters

The information in this chapter is supplemental to that provided in Chapter 12, "Reconfiguring the System Kernel." The text assumes that you are an advanced system administrator or programmer.

## 16.1. Files Used in Kernel Configuration

The configuration-build utility /usr/etc/config. uses information from the following files to build the kernel:

Makefile.*architecture.name*
Generic makefile, where *architecture.name* is sun2, sun3, or sun4, depending on the machine's model type.

files.cmn — List files required to build the basic kernel.

files — Machine-specific files

devices — Name to major device mapping

SYSTEM_NAME — Describes characteristics of a specific system named *SYSTEM_NAME.* This is kernel configuration file for the specific system, as created during reconfiguration process described in Chapter 12.

From these files, /usr/etc/config generates the files needed to compile and link your kernel. One of these files is a *Makefile,* which you then use to actually build the new system and install the kernel.

## Adding Your Own Device Drivers

NOTE: *In addition to entries in the configuration file, new device drivers require entries in* `/sys/sun/conf.c`, `/sys/conf/files`, *and possibly* `/sys/sun/swapgeneric.c` *and* `sys/conf/devices`. *New devices also require you to add one or more new special files to* `/dev`. *Refer to the manual "Writing Device Drivers" for more information on these files.*

To add a new device driver, you have to add lines in two of the three major sections of the GENERIC configuration file. These lines are options lines in the system identification section and device lines in the devices section. which are completely defined in Chapter 12.

### The `options` *optlist* Line

Device lines of this type should be located in the system identification section along with `options` lines provided by Sun. You use the `options` *optlist* argument when you want to add your own device drivers to the configuration file. This argument tells the kernel to compile the listed options (*optlist*) into the system. The items in *optlist* are separated by commas. A line of the form `"options FUNNY, HAHA"` yields `-DFUNNY -DHAHA` to the C compiler. An option may be given a value by following its name with "=", then the value enclosed in double quotes. None of the standard options use such a value.

In addition, options can be used to bring in additional files if the option is listed in the *files* files. All options should be listed in uppercase. In this case, no corresponding *option*`.h` will be created, as it would be using the corresponding *pseudo-device* method.

### Adding Device Description Lines

Chapter 12 provides a general overview of device description lines. The sample kernel configuration files in that chapter illustrate how Sun has used these lines to configure standard equipment.

The syntax of the device description line is:

```
dev_type dev_name  at  connect_dev more info
```

The arguments above are explained in detail, except the *more_info* argument, which is especially important when you add your own drivers. *more_info* is a sequence of the following:

- `csr` *addr*. This specifies the address of the csr (command and status registers) for a device. The csr addresses specified for the device are always the addresses within the bus type specified. For example, on the following line in a Sun-2 configuration file:

```
controller      sc0 at mbmem ? csr 0x80000
```

the address is 0x80000. The line indicates that the sc0 controller exists in the Multibus memory (mbmem) at address 0x80000.

When configuring your own drivers, the csr address must be specified for all controllers, and for all devices connected to a main system bus.

- `drive` *number*. This specifies which drive the line applies to. For example, the argument `sd0` refers to the first SCSI disk drive on a system's first SCSI controller, and the argument `xd4` refers to the first SMD disk drive on a system's second Xylogics 7053 controller.
- `flags` *number*. These are flags made available to the device driver, which are usually read during system initialization.
- `priority` *level*. This argument specifies the interrupt level at which a device that interrupts must operate.
- `vector` *intr number* [*intr number...*]. This argument applies to devices that use vectored interrupts on VMEbus systems. *intr* specifies the vectored interrupt routine, and `number` specifies the corresponding vector to be used (64-255).

If you are adding non-standard devices to the configuration file, you can substitute a question mark (?) for a number in two places. You can put questions marks on a *connect_dev* (for example, `xyc?`), or on a drive number (for example, drive ?). The system will automatically replace the ? with the appropriate value during boot up. This allows redundancy, since a single system can be built that will boot on different hardware configurations.

## 16.2. Configuring Systems Without Source

Object-only releases have binaries for standard system modules in the directory `/usr/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:

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 `/usr/sys/OBJ`. Attempting to do so will not be detected and may cause the kernel to appear to work but have only occasional failures. Double check the `.h` files in `/usr/sys/OBJ` if you change the number of devices configured for any standard drivers.

## 16.3. Sharing Object Modules

If you have many kernels that are all built on a single machine there are at least two approaches to save time in building kernel images. The best way is to have a single kernel image being run on all machines. This is attractive because it minimizes disk space used and time required to rebuild kernels after making changes. However, often one or more systems requires a separately configured kernel image. This may be due to limited memory (building a kernel with many unused device drivers wastes core), or to configuration requirements (one machine may be a development machine where accounting is not needed, while another is a production machine where it is). In these cases kernels could share relocatable object modules which are not configuration dependent. Most of the modules in the directory `/usr/sys/os` are of this type.

To share object modules, first build a GENERIC kernel. Then, for each kernel, configure as before, but before recompiling and linking, type `make links` (specifically, after the `make depend` command and before the `make` that makes your kernel, as explained in Chapter 12.) This causes the kernel source to be searched for source modules that can be shared 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 and executed with this request and may be checked for correctness.

The file `/sys/conf.common/defines` contains a list of symbols that are relatively safe to ignore when checking the source code for modules 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 that are not directly visible in the source code are not caught. Thus if you place per-system dependencies in an include file, they will not be recognized.

### Note 4: Building Profiled Kernels

It is simple to configure a kernel that will automatically collect profiling information as it operates. The profiling data can be collected with `kgmon` (8) and processed with `gprof` (1) to obtain information about how the kernel operates. Profiled kernels maintain histograms of the program counter as well as the number of invocations of each routine. The `gprof` (1) command generates a dynamic call graph of the executing kernel and propagates time spent in each routine along the arcs of the call graph.

To configure a profiled kernel, use the `-p` option with the `config` program. A profiled kernel is about 5-10 percent larger in its text space due to the calls to count the subroutine invocations. When the kernel executes, the profiling data is stored in a buffer that is 1.2 times the size of the text space. The overhead for running a profiled kernel varies; under normal load this may be anywhere from 5-25 percent of the kernel time spent in the profiling code.

Note that kernels configured for profiling should not be shared as described above unless all the other shared kernels are also to be profiled.

## 16.4. Rules for Defaulting System Devices

This section covers the rules the `/usr/etc/config` program uses to define underspecified locations of system devices.

When the `/usr/etc/config` program processes a `config` line that does not fully specify the location of the root file system, swap or paging area, and device for system dumps it applies a set of rules to define those values left unspecified. The following list of rules is used 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 chosen for system dumps, the swap partition is selected (see below to find out where dumps are placed within the partition).

The following table summarizes the default partitions selected when a device specification is incomplete, e.g. xy0.

| Type | Partition |
|---|---|
| root | a |
| /export/swap | b |
| /export/dump | d |

You can use swapon(8) additional swapping devices/files while your system is in use. Only the primary swap device needs to be known by the kernel at boot time. Add secondary devices as needed by using swapon.

Finally, system dumps are automatically taken after a system crash, provided the device driver for the dumps device supports this. The dump contains the contents of memory, but not the swap areas. Normally the dump device is a disk, in which case the information is copied to a location near the back of the partition. The dump is placed in the back of the partition because the primary swap and dump device are commonly the same device and this allows the system to be rebooted without immediately overwriting the saved information. When a dump has occurred, the system variable dumpsize is set to a non-zero value indicating the size (in bytes) of the dump. The savecore (8) program then copies the information from the dump partition to a file in a "crash" directory and also makes a copy of the system that was running at the time of the crash (usually /vmunix).

## Configuration File Grammar

This section covers the grammar used by the /usr/etc/config program to parse the input kernel configuration file.

The following grammar is a compressed form of the actual yacc (1) grammar used by the /usr/etc/config program to parse configuration files. Terminal symbols are shown all in uppercase, literals are emboldened; optional clauses are enclosed in brackets, [ and ]; zero or more instances are denoted with *.

```
Configuration ::=  [ Spec ; ]*

Spec ::= Config_spec
     | Device_spec
     | trace
     | /* lambda */

/* configuration specifications */

Config_spec ::=  machine ID
     | cpu ID
     | options Opt_list
     | ident ID
     | System_spec
     | maxusers NUMBER

/* system configuration specifications */

System_spec ::= config ID System_parameter [ System_parameter ]*

System_parameter ::=  swap_spec | root_spec | dump_spec

swap_spec ::=  swap [ on ] swap_dev

swap_dev ::=  dev_spec [ size NUMBER ]

root_spec ::=  root [ on ] dev_spec

dump_spec ::=  dumps [ on ] dev_spec

dev_spec ::=  dev_name | major_minor

major_minor ::=  major NUMBER minor NUMBER

dev_name ::=  ID [ NUMBER [ ID ] ]

/* option specifications */

Opt_list ::=  Option [ , Option ]*

Option ::=  ID [ = Opt_value ]

Opt_value ::=  ID | NUMBER

/* device specifications */

Device_spec ::= device Dev_name Dev_info Int_spec
     | disk Dev_name Dev_info
     | tape Dev_name Dev_info
     | controller Dev_name Dev_info [ Int_spec ]
     | pseudo-device Dev [ NUMBER ]
```

```
Dev_name ::=  Dev NUMBER

Dev_info ::=  Con_info [ Info ]*

Con_info ::=  at Dev NUMBER
    | at nexus NUMBER

Info ::=  csr NUMBER
    | drive NUMBER
    | slave NUMBER
    | flags NUMBER

Int_spec ::=  priority NUMBER
    | priority NUMBER vector ID NUMBER [ ID NUMBER ] *
    | /* epsilon */
```

### Lexical Conventions

The terminal symbols are loosely defined as:

`ID`
: One or more 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.

In special instances a question mark, `?`, can be substituted for a `NUMBER` token. This is used to effect wildcarding in device interconnection specifications.

Comments in configuration files are indicated by a # character at the beginning of the line; the remainder of the line is discarded.

A specification is interpreted as a continuation of the previous line if the first character of the line is tab.

## Data Structure Sizing Rules

This section explains the rules for sizing kernel data structures, both those calculated at compile time and those calculated at boot time.

Certain kernel data structures are sized at compile time according to the maximum number of simultaneous users expected, while others are calculated at boot time based on the physical resources present, such as memory. This section lists both sets of rules and also includes some hints on changing built-in limitations on certain data structures.

### Compile Time Rules

The file `/usr/sys/conf.common/param.c` contains the definitions of almost all data structures sized at compile time. This file is copied into the directory of each configured kernel to allow configuration-dependent rules and values to be maintained. The rules implied by its contents are summarized below (Here `MAXUSERS` refers to the value defined in the configuration file in the `maxusers` rule.)

**nproc**

The maximum number of processes that may be running at any time. It is defined to be 10 + 16 * MAXUSERS and referred to in other calculations as NPROC.

**ninode**

The maximum number of UFS files in the file system that may be active at any time. This includes files being used by users, as well as directory files being read or written by the system and files associated with bound sockets in the IPC domain. This is defined as (NPROC + 16 + MAXUSERS) + 64.

**nfile**

The number of file table structures. One file table structure is used for each open, unshared, file descriptor. Multiple file descriptors may refer to a single file table entry when they are created through a dup call, or as the result of a fork. This is defined to be

```
16 * (NPROC + 16 + MAXUSERS) / 10 + 64 NWIN = 0

16 * (NPROC + 16 + MAXUSERS)/5 + 64 NWIN > 0
```

**ncallout**

The number of callout structures. One callout structure is used per internal system event handled with a timeout. Timeouts are used for terminal delays, watchdog routines in device drivers, protocol timeout processing, etc. This is defined as 16 + NPROC.

**nclist**

The number of c-list structures. C-list structures are used in terminal I/O. This is defined as 100 + 16 * MAXUSERS.

**nmbclusters**

The maximum number of pages that may be allocated by the network. This is defined as 256 pages in /usr/sys/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.

**ndquot**

The number of dquot structures allocated. dquot structures are present only when disk 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 (MAXUSERS * NMOUNT) / 4 + NPROC, where NMOUNT is the maximum number of mountable file systems.

## 16.5. Tuning IPC System Parameters

The System V Inter-Process Communications (IPC) extensions to UNIX are now implemented in the SunOS operating system. These IPC extensions provide mechanisms for message passing, semaphores, and shared memory.

Data structures that describe and control various IPC functions are allocated in the SunOS kernel at system initialization and remain resident in memory as long as the operating system is running. The size of these structures is determined by a variety of tunable parameters in the system configuration file. Some IPC parameters also exist to control and limit the resources dynamically allocated by IPC subsystems.

This section describes these parameters and gives some guidelines for tuning them. Before attempting to modify any of these parameters, you should be thoroughly familiar with the capabilities provided by the IPC system, and with your particular application's needs.

Refer to the *AT&T System V Interface Definition* (SVID) and Sun's *SunOS Reference Manual.* for a detailed description of the IPC system interface. But note that there are now two separate IPC subsystems – the one from 4.x BSD involving sockets, and the one from System V discussed here.

### Why Reconfigure IPC Parameters?

The Sun Operating System, as shipped, is configured to support a modest level of IPC activity. Applications that require more IPC resources than are provided in the distributed system must be run with reconfigured kernels. In general, Sun recommends that all system kernels be reconfigured in order to reduce the base-level memory requirements, as described earlier in Chapter 9. The following paragraphs assume a working knowledge of the system configuration procedure.

IPC parameters may be tuned by editing the system configuration file `/usr/sys/conf/SYSTEM_NAME` and rebuilding the kernel. In order to adjust the default setting of a particular parameter, insert a line of the following form into the configuration file:

```
options  OPTION_NAME = OPTION_VALUE
```

Default values are assumed for all unspecified parameters.

**Note:** Do not change the `options` lines that refer to the IPC subsystems unless you wish to disable the entire subsystem.

Users of System V may note that there is not an exact correspondence between Sun's tunable parameters and those found in the AT&T System V release. This is because several implementation algorithms have been changed, rendering some of the parameters meaningless. In particular, certain static structures in System V are allocated dynamically by the Sun kernel, obviating the need for configurable limits. In future Sun releases, there are likely to be even fewer tunable parameters. It should be noted that although some parameters have been omitted, there are no semantic differences in the IPC system call operation.

For the remaining tunable parameters, reasonable values may be estimated using information about the application programs' IPC usage. The following sections specify the default values and attempt to give tuning guidelines. It is generally a good idea to write application programs that recover gracefully from resource allocation failures, so that system configuration requirements surface early in the development cycle. Applications that require large amounts of IPC resources are often poorly designed and should be carefully reviewed before making drastic increases in the IPC parameters.[2]

### IPC Message Parameters

These parameters control system characteristics associated with inter-process message passing.

#### `MSGPOOL`

`MSGPOOL` defines the size (in kilobytes) of the kernel message memory pool. All queued IPC messages are stored in this pool. The behavior of the *msgsnd*(2) system call when the message pool is full depends on the value of the `msgflg` argument; see *msgop*(2). Attempts to queue messages larger than the message pool return EINVAL.

[2] The total amount of memory available for user processes may be estimated using the `vmstat`(8) program.

The IPC message pool is allocated at system initialization and is never made available for other uses. Thus, tuning the MSGPOOL parameter involves a trade-off between the performance of processes that depend upon a high level of message activity, and the degradation of overall system performance due to wasted memory. This parameter may not exceed 255.

```
options     MSGPOOL=8
```

MSGMNB

MSGMNB defines the maximum number of message bytes that may be queued on any particular message queue. Attempts to queue messages that would exceed this limit either sleep or return EAGAIN; see *msgop*(2). MSGMNB is used as a default value for msg_qbytes when message queues are created; this limit may be lowered by any process but only the super-user may raise the limits on a particular message queue; see the *msgctl*(2) option named IPC_SET.

```
options     MSGMNB=2048
```

MSGMNI

MSGMNI defines the maximum number of uniquely identifiable message queues that may exist simultaneously. Attempts to create more than MSGMNI message queues return ENOSPC; see *msgget*(2). Each increment of MSGMNI reserves 49 bytes of kernel memory.

```
options     MSGMNI=50
```

MSGTQL

MSGTQL defines the total number of undelivered messages that may exist at any instant. Attempts to queue more than MSGTQL messages either sleep or return EAGAIN; see *msgop*(2). Since zero-length messages are allowed, this limit could theoretically be set arbitrarily high. Each increment of MSGTQL reserves 12 bytes.

```
options     MSGTQL=50
```

**IPC Semaphore Parameters**

These parameters control system characteristics associated with inter-process semaphores.

SEMMNI

SEMMNI defines the maximum number of uniquely identifiable semaphore clusters that may exist simultaneously. Attempts to create more than SEMMNI semaphore clusters return ENOSPC; see *semget*(2). Although SEMMNI may be set arbitrarily high, there is no reason to set it to be larger than SEMMNS. Each increment of SEMMNI reserves 32 bytes of kernel memory.

```
options     SEMMNI=10
```

SEMMNS

SEMMNS defines the maximum number of semaphores in the system. Attempts to create semaphore clusters when there are not enough semaphores available result in an ENOSPC error; see *semget*(2). Attempts to create semaphore clusters with more than SEMMNS semaphores return EINVAL. Each increment of SEMMNS reserves 8 bytes.

```
options     SEMMNS=60
```

SEMUME

SEMUME defines the maximum number of semaphores (per process) that may simultaneously have non-zero adjust-on-exit values. The adjust-on-exit values are manipulated when semaphore operations are requested in conjunction with the SEM_UNDO flag. Attempts to exceed this limit return EINVAL; see *semop*(2). The value of SEMUME affects the number of bytes allocated for semaphore undo structures (see SEMMNU below). The value of SEMUME must be less than 32768.

```
options     SEMUME=10
```

SEMMNU

SEMMNU defines the maximum number of processes that may simultaneously be using the IPC SEM_UNDO feature. Attempts to exceed this limit result in an ENOSPC error; see *semop*(2). There is no reason to set SEMMNU larger than the maximum number of processes that can run on the system (approximately 16*maxusers, where maxusers is configurable, defaulting to 4). Therefore, SEMMNU need not exceed 64, under normal circumstances. Each increment of SEMMNU allocates 14+(8*SEMUME) bytes.

```
options     SEMMNU=30
```

### IPC Shared Memory Parameters

These parameters control system characteristics associated with inter-process shared memory.

SHMMNI

SHMMNI defines the maximum number of shared memory segments that may simultaneously exist in the system. Attempts to exceed this limit return ENOSPC; see *shmget*(2). Each increment of SHMMNI reserves 42 bytes.

```
options     SHMMNI=100
```

SHMSIZE

SHMSIZE defines the size, in kilobytes, of the largest single shared memory segment. Attempts to exceed this limit return EINVAL; see shmget(2). Although indiscriminate allocation of shared memory can exhaust system resources (swap space for instance), there is no enforced limit on the total amount of shared memory that may be allocated. However, in order to protect against simple program errors, this parameter limits the size of individual shared segments. The default value, equivalent to 1 MByte, should be sufficient for most applications.

```
options     SHMSIZE=1024
```

### Named Pipe Parameters

These parameters control system characteristics associated with FIFO special files (named pipes).

FIFOCNT

FIFOCNT determines the maximum number of FIFOs that may be in use in the system at any given time. Attempts to write to more than FIFOCNT simultaneous FIFOs will block until some FIFOs are closed, releasing system resources. In the current implementation, FIFO buffers are allocated in non-paging memory, and can lead to system lockup if indiscriminately allocated. This restriction may be removed in a future release.

```
options     FIFOCNT=10
```

# 17

# File System Check Program

# 17

# File System Check Program

When the SunOS operating system is brought up, a consistency check of the file systems should always be performed. This precautionary measure helps to insure a reliable environment for file storage on disk. If an inconsistency is discovered, corrective action must be taken. `fsck` runs in two modes. Normally it is run non-interactively by the system after a normal boot. When running in this mode, it will only make changes to the file system that are known to always be correct. If an unexpected inconsistency is found, `fsck` will exit with a non-zero exit status, leaving the system running single user. Typically you then run `fsck` interactively. When running in this mode, each problem is listed followed by a suggested corrective action. You must decide whether or not the suggested correction should be made.

The purpose of this appendix is to dispel the mystique surrounding file system inconsistencies. It first describes the updating of the file system (the calm before the storm) and then describes file system corruption (the storm). Finally, the set of deterministic corrective actions used by `fsck` is presented.

## 17.1. Overview of the File System

The file system is discussed in detail in [McKusick84]; this section gives a brief overview.

### Superblock

A file system is described by its *superblock*. The superblock is built when the file system is created (see `newfs` (8)) and never changes. The superblock contains the basic parameters of the file system, such as the number of data blocks it contains and a count of the maximum number of files. Because the superblock contains critical data, `newfs` replicates it to protect against catastrophic loss. The *default superblock* always resides at a fixed offset from the beginning of the file system's disk partition. The *redundant superblocks* are not referenced unless a head crash or other hard disk error causes the default superblock to be unusable. The redundant blocks are sprinkled throughout the disk partition.

Within the file system are files. Certain files are distinguished as directories and contain collections of pointers to files that may themselves be directories. Every file has a descriptor associated with it called an *inode*. The inode contains information describing ownership of the file, time stamps indicating modification and

[2] This document reflects the use of `fsck` with the file system organization implemented in release 4.0 of the SunOS operating system. This is a revision of the original paper written by T. J. Kowalski and modified by Kirk McKusick.

access times for the file, and an array of indices pointing to the data blocks for the file. This section assumes that the first 12 blocks of the file are directly referenced by values stored in the inode structure itself†. The inode structure may also contain references to indirect blocks containing further data block indices. In a file system with a 8192 byte block size, a singly indirect block contains 1024 further block addresses, a doubly indirect block contains 1024 addresses of further single indirect blocks, and a triply indirect block contains 1024 addresses of further doubly indirect blocks (the triply indirect block is never needed in practice).

The standard SunOS block size is 8K; fragment size is 1K.

In order that files with up to $2^{32}$ bytes require only two levels of indirection, the minimum size of a file system block is 4096 bytes. The size of file system blocks can be any power of two greater than or equal to 4096. The block size of the file system is maintained in the superblock, so it is possible for file systems of different block sizes to be accessible simultaneously on the same system. The block size must be decided when newfs creates the file system; the block size cannot be subsequently changed without rebuilding the file system.

### Summary Information

Associated with the superblock is non-replicated *summary information*. The summary information changes as the file system is modified. The summary information contains the number of blocks, fragments, inodes, and directories in the file system.

### Cylinder Groups

The file system partitions the disk into one or more areas called *cylinder groups*. A cylinder group is comprised of one or more consecutive cylinders on a disk. Each cylinder group includes inode slots for files, a *block map* describing available blocks in the cylinder group, and summary information describing the usage of data blocks within the cylinder group. A fixed number of inodes is allocated for each cylinder group when the file system is created. The current policy is to allocate one inode for each 2048 bytes of disk space; this is expected to be far more inodes than will ever be needed.

All the cylinder group bookkeeping information could be placed at the beginning of each cylinder group. However if this approach were used, all the redundant information would be on the top platter. A single hardware failure that destroyed the top platter could cause the loss of all copies of the redundant superblocks. Thus the cylinder group bookkeeping information begins at a floating offset from the beginning of the cylinder group. The offset for the *i+1*st cylinder group is about one track further from the beginning of the cylinder group than it was for the *i*th cylinder group. In this way, the redundant information spirals down into the pack; any single track, cylinder, or platter can be lost without losing all copies of the superblocks. Except for the first cylinder group, the space between the beginning of the cylinder group and the beginning of the cylinder group information stores data.

[2] †The actual number may vary from system to system, but is usually in the range 5-13.

## Fragments

To avoid waste in storing small files, the file system space allocator divides a single file system block into one or more *fragments*. The fragmentation of the file system is specified when the file system is created; each file system block can be optionally broken into 2, 4, or 8 addressable fragments. The lower bound on the size of these fragments is constrained by the disk sector size; typically 512 bytes is the lower bound on fragment size. The block map associated with each cylinder group records the space availability at the fragment level. Aligned fragments are examined to determine block availability.

On a file system with a block size of 8192 bytes and a fragment size of 1024 bytes, a file is represented by zero or more 8192 byte blocks of data, and possibly a single fragmented block. If a file system block must be fragmented to obtain space for a small amount of data, the remainder of the block is made available for allocation to other files. For example, consider an 11000 byte file stored on a 8192/1024 byte file system. This file uses one full size block and a 3072 byte fragment. If no fragments with at least 3072 bytes are available when the file is created, a full size block is split yielding the necessary 3072 byte fragment and an unused 1024 byte fragment. This remaining fragment can be allocated to another file, as needed.

## Updates to the File System

Every working day hundreds of files are created, modified, and removed. Every time a file is modified, the operating system performs a series of file system updates. These updates, when written on disk, yield a consistent file system. The file system stages all modifications of critical information; modification can either be completed or cleanly backed out after a crash. Knowing the information that is first written to the file system, deterministic procedures can be developed to repair a corrupted file system. To understand this process, you must first know the order in which the update requests were being honored.

When a user program does an operation to change the file system, such as a *write*, the data to be written is copied into an internal *in-core* buffer in the kernel. Normally, the disk update is handled asynchronously; the user process is allowed to proceed even though the data has not yet been written to the disk. The data, along with the inode information reflecting the change, is eventually written out to disk. The real disk write may not happen until long after the *write* system call has returned. Thus at any given time, the file system, as it resides on the disk, lags behind the state of the file system represented by the in-core information.

The disk information is updated to reflect the in-core information when the buffer is required for another use, when a `sync` (2) is done (at 30 second intervals) by `/usr/etc/update` (8), or by manual operator intervention with the `sync` (8) command. If the system is halted without writing out the in-core information, the file system on the disk will be in an inconsistent state.

If all updates are done asynchronously, several serious inconsistencies can arise. One inconsistency is that a block can be claimed by two inodes. Such an inconsistency can occur when the system is halted before the pointer to the block in the old inode has been cleared in the copy of the old inode on the disk, and after the pointer to the block in the new inode has been written out to the copy of the new inode on the disk. Here, there is no deterministic method for deciding which inode should really claim the block. A similar problem can arise with a multiply claimed inode.

The problem with asynchronous inode updates can be avoided by doing all inode deallocations synchronously. Consequently, inodes and indirect blocks are written to the disk synchronously (that is, the process blocks until the information is really written to disk) when they are being deallocated. Similarly, inodes are kept consistent by synchronously deleting, adding, or changing directory entries.

## 17.2. Fixing Corrupted File Systems

A file system can become corrupted in several ways. The most common of these ways are improper shutdown procedures and hardware failures.

File systems may become corrupted during an *unclean halt*. This happens when proper shutdown procedures are not observed, physically write-protecting a mounted file system, or a mounted file system is taken off-line. The most common operator procedural failure is forgetting to `sync` the system before halting the CPU.

File systems can become further corrupted if proper startup procedures are not observed, for example, not checking a file system for inconsistencies, and not repairing those inconsistencies. Allowing a corrupted file system to be used (and to be modified further) can be disastrous.

Any piece of hardware can fail at any time. Failures can be as subtle as a bad block on a disk pack, or as blatant as a non-functional disk-controller.

### Detecting and Correcting Corruption

Normally you run `fsck` non-interactively. In this mode it only fixes corruptions that are expected to occur from an unclean halt. These actions are a proper subset of the actions that `fsck` takes when it is running interactively. This appendix assumes that you are running `fsck` interactively, and all possible errors can be encountered. When an inconsistency is discovered in this mode, `fsck` reports the inconsistency for you to choose a corrective action.

You can check a quiescent file system (that is, one that is unmounted and is not being written on) for structural integrity by performing consistency checks on the redundant data present on a file system. The redundant data is either read from the file system, or computed from other known values. The file system **must** be in a quiescent state when you run `fsck` because `fsck` is a multi-pass program.

The following sections discuss methods to discover inconsistencies and possible corrective actions for the cylinder group blocks, the inodes, the indirect blocks, and the data blocks containing directory entries.

### Superblock Checking

The most commonly corrupted item in a file system is the summary information associated with the superblock. The summary information is prone to corruption because it is modified with every change to the file system's blocks or inodes, and is usually corrupted after an unclean halt.

The superblock is checked for inconsistencies involving file system size, number of inodes, free block count, and the free inode count. The file system size must be larger than the number of blocks used by the superblock and the number of blocks used by the list of inodes. The file system size and layout information are the most critical pieces of information for `fsck`. While there is no way to actually check these sizes, since they are statically determined by newfs, fsck can check that these sizes are within reasonable bounds. All other file system checks require that these sizes be correct. If fsck detects corruption in the static parameters of the default superblock, fsck requests the operator to specify the location of an alternate superblock.

### Free Block Checking

fsck checks that all the blocks marked as free in the cylinder group block maps are not claimed by any files. When all the blocks have been initially accounted for, fsck checks that the number of free blocks plus the number of blocks claimed by the inodes equals the total number of blocks in the file system.

If anything is wrong with the block allocation maps, fsck will rebuild them, based on the list it has computed of allocated blocks.

The summary information associated with the superblock counts the total number of free blocks within the file system. fsck compares this count to the number of free blocks it found within the file system. If the two counts do not agree, then fsck replaces the incorrect count in the summary information by the actual free block count.

The summary information counts the total number of free inodes within the file system. fsck compares this count to the number of free inodes it found within the file system. If the two counts do not agree, then fsck replaces the incorrect count in the summary information by the actual free inode count.

### Checking the Inode State

An individual inode is not as likely to be corrupted as the allocation information. However, because of the great number of active inodes, a few of the inodes are usually corrupted.

The list of inodes in the file system is checked sequentially starting with inode 2 (inode 0 marks unused inodes; inode 1 is saved for future generations) and progressing through the last inode in the file system. The state of each inode is checked for inconsistencies involving format and type, link count, duplicate blocks, bad blocks, and inode size.

Each inode contains a mode word. This mode word describes the type and state of the inode. Inodes must be one of six types: regular inode, directory inode, symbolic link inode, special block inode, special character inode, or socket inode. Inodes may be found in one of three allocation states: unallocated, allocated, and neither unallocated nor allocated. This last state suggests an incorrectly formated inode. An inode can get in this state if bad data is written into the inode list. The only possible corrective action is for fsck is to clear the inode.

### Inode Links

Each inode counts the total number of directory entries linked to the inode. fsck verifies the link count of each inode by starting at the root of the file system, and descending through the directory structure. The actual link count for each inode is calculated during the descent.

If the stored link count is non-zero and the actual link count is zero, then no directory entry appears for the inode. If this happens, fsck will place the disconnected file in the lost+found directory. If the stored and actual link counts are non-zero and unequal, a directory entry may have been added or removed without the inode being updated. If this happens, `fsck` replaces the incorrect stored link count by the actual link count.

Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by the inode. Since indirect blocks are owned by an inode, inconsistencies in indirect blocks directly affect the inode that owns it.

`fsck` compares each block number claimed by an inode against a list of already allocated blocks. If another inode already claims a block number, then the block number is added to a list of *duplicate blocks*. Otherwise, the list of allocated blocks is updated to include the block number.

If there are any duplicate blocks, `fsck` performs a partial second pass over the inode list to find the inode of the duplicated block. The second pass is needed, since without examining the files associated with these inodes for correct content, not enough information is available to determine which inode is corrupted and should be cleared. If this condition does arise (only hardware failure will cause it), then the inode with the earliest modify time is usually incorrect, and should be cleared. If this happens, `fsck` prompts the operator to clear both inodes. The operator must decide which one should be kept and which one should be cleared.

`fsck` checks the range of each block number claimed by an inode. If the block number is lower than the first data block in the file system, or greater than the last data block, then the block number is a *bad block number*. Many bad blocks in an inode are usually caused by an indirect block that was not written to the file system, a condition which can only occur if there has been a hardware failure. If an inode contains bad block numbers, `fsck` prompts the operator to clear it.

### Inode Data Size

Each inode contains a count of the number of data blocks that it contains. The number of actual data blocks is the sum of the allocated data blocks and the indirect blocks. `fsck` computes the actual number of data blocks and compares that block count against the actual number of blocks the inode claims. If an inode contains an incorrect count `fsck` prompts the operator to fix it.

Each inode contains a thirty-two bit size field. The size is the number of data bytes in the file associated with the inode. The consistency of the byte size field is roughly checked by computing from the size field the maximum number of blocks that should be associated with the inode, and comparing that expected block count against the actual number of blocks the inode claims.

### Checking the Data Associated with an Inode

An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be the same kind. The three types of data blocks are: plain data blocks, symbolic link data blocks, and directory data blocks. Plain data blocks contain the information stored in a file; symbolic link data blocks contain the path name stored in a link. Directory data blocks contain directory entries. `fsck` can only check the validity of directory data blocks.

Each directory data block is checked for several types of inconsistencies. These inconsistencies include directory inode numbers pointing to unallocated inodes, directory inode numbers that are greater than the number of inodes in the file system, incorrect directory inode numbers for "." and "..", and directories that are not attached to the file system. If the inode number in a directory data block references an unallocated inode, then fsck will remove that directory entry. Again, this condition can only arise when there has been a hardware failure.

If a directory entry inode number references outside the inode list, then fsck will remove that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for "." must be the first entry in the directory data block. The inode number for "." must reference itself; e.g., it must equal the inode number for the directory data block. The directory inode number entry for ".." must be the second entry in the directory data block. Its value must equal the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory). If the directory inode numbers are incorrect, fsck will replace them with the correct values. If there are multiple hard links to a directory, the first one encountered is considered the real parent to which ".." should point; fsck recommends deletion for the subsequently discovered names.

### File System Connectivity

fsck checks the general connectivity of the file system. If directories are not linked into the file system, then fsck links the directory back into the file system in the lost+found directory. This condition only occurs when there has been a hardware failure.

## 17.3. fsck Error Conditions

### Conventions

fsck is a multi-pass file system check program. Each file system pass invokes a different phase of the fsck program. After the initial setup, fsck performs successive phases over each file system, checking blocks and sizes, path names, connectivity, reference counts, and the map of free blocks, (possibly rebuilding it), and performs some cleanup.

Normally fsck is run non-interactively to *preen* the file systems after an unclean halt. While preening a file system, it will only fix corruptions that are expected to occur from an unclean halt. These actions are a proper subset of the actions that fsck will take when it is running interactively. Throughout this appendix many errors have several options that the operator can take. When an inconsistency is detected, fsck reports the error condition. If a response is required, fsck prints a prompt message and waits for a response. When preening most errors are fatal. For those that are expected, the response taken is noted. This appendix explains the meaning of each error condition, the possible responses, and the related error conditions.

The error conditions are organized by the *phase* of the fsck program in which they can occur. The error conditions that may occur in more than one phase will be discussed in initialization.

## Initialization

Before a file system check can be performed, certain tables have to be set up and certain files opened. This section concerns itself with the opening of files and the initialization of tables. This section lists error conditions resulting from command line options, memory requests, opening of files, status of files, file system size checks, and creation of the scratch file. All the initialization errors are fatal when the file system is being preened.

**Message**

```
C option?
```

*C* is not a legal option to `fsck`; legal options are –b, –y, –n, and –p. `fsck` terminates on this error condition. See the `fsck` (8) manual entry for further detail.

**Message**

```
cannot alloc NNN bytes for blockmap
cannot alloc NNN bytes for freemap
cannot alloc NNN bytes for statemap
cannot alloc NNN bytes for lncntp
```

`fsck`'s request for memory for its virtual memory tables failed. This should never happen. `fsck` terminates on this error condition. See a guru.

**Message**

```
Can't open checklist file: F
```

The file system checklist file *F* (usually `/etc/mtab`) cannot be opened for reading. `fsck` terminates on this error condition. Check access modes of *F*.

**Message**

```
Can't stat root
```

`fsck`'s request for statistics about the root directory ''/'' failed. This should never happen. `fsck` terminates on this error condition. See a guru.

**Message**

```
Can't stat F
Can't make sense out of name F
```

`fsck`'s request for statistics about the file system *F* failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of *F*.

**Message**

```
Can't open F
```

`fsck`'s request attempt to open the file system *F* failed. When running manually, it ignores this file system and continues checking the next file system given. Check access modes of *F*.

**Message**

```
F: (NO WRITE)
```

Either the `-n` flag was specified or `fsck`'s attempt to open the file system *F* for writing failed. When running manually, all the diagnostics are printed out, but no modifications are attempted to fix them.

**Message**

```
file is not a block or character device; OK
```

You have given `fsck` a regular file name by mistake. Check the type of the file specified.

Possible responses to the `OK` prompt are:

YES
: Ignore this error condition.

NO
: Ignore this file system and continue checking the next file system given.

**Message**

```
UNDEFINED OPTIMIZATION IN SUPERBLOCK (SET TO DEFAULT)
```

The superblock optimization parameter is neither `OPT_TIME` nor `OPT_SPACE`.

Possible responses to the `SET TO DEFAULT` prompt are:

YES
: Set the superblock to request optimization to minimize running time of the system. (If optimization to minimize disk space utilization is desired, it can be set using `tunefs` (8).)

NO
: Ignore this error condition.

**Message**

```
IMPOSSIBLE MINFREE=D IN SUPERBLOCK (SET TO DEFAULT)
```

The superblock minimum space percentage is greater than 99% or less then 0%.

Possible responses to the `SET TO DEFAULT` prompt are:

YES
: Set the minfree parameter to 10%. (If some other percentage is desired, it can be set using `tunefs` (8).)

NO

Ignore this error condition.

**Message**

```
MAGIC NUMBER WRONG
NCG OUT OF RANGE
CPG OUT OF RANGE
NCYL DOES NOT JIVE WITH NCG*CPG
SIZE PREPOSTEROUSLY LARGE
TRASHED VALUES IN SUPER BLOCK
```

followed by the message:

```
F: BAD SUPER BLOCK: B
USE -b OPTION TO FSCK TO SPECIFY LOCATION OF AN ALTERNATE
SUPER-BLOCK TO SUPPLY NEEDED INFORMATION; SEE fsck(8).
```

The superblock has been corrupted. An alternative superblock must be selected from among those listed by `newfs` (8) when the file system was created. For file systems with a block size less than 32K, specifying `–b32` is a good first choice.

**Message**

```
INTERNAL INCONSISTENCY: M
```

`fsck`'s has had an internal panic, whose message is specified as *M*. This should never happen. See a guru.

**Message**

```
CAN NOT SEEK: BLK B (CONTINUE)
```

`fsck`'s request for moving to a specified block number *B* in the file system failed. This should never happen. 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 recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` will terminate with the message

```
Fatal I/O error
```

NO

Terminate the program.

**Message**

```
CAN NOT READ: BLK B (CONTINUE)
```

fsck's request for reading a specified block number *B* in the file system failed. This should never happen. See a guru.

Possible responses to the CONTINUE prompt are:

YES

Attempt to continue to run the file system check. It will retry the read and print out the message:

```
THE FOLLOWING SECTORS COULD NOT BE READ: N
```

where *N* indicates the sectors that could not be read. If fsck ever tries to write back one of the blocks on which the read failed it will print the message:

```
WRITING ZERO'ED BLOCK N TO DISK
```

where *N* indicates the sector that was written with zero's. If the disk is experiencing hardware problems, the problem will persist. This error condition will not allow a complete check of the file system. A second run of fsck should be made to recheck this file system. If the block was part of the virtual memory buffer cache, fsck will terminate with the message

```
Fatal I/O error
```

NO

Terminate the program.

**Message**

```
CAN NOT WRITE: BLK B (CONTINUE)
```

fsck's request for writing a specified block number *B* in the file system failed. The disk is write-protected; check the write protect lock on the drive. If that is not the problem, see a guru.

Possible responses to the CONTINUE prompt are:

YES

Attempt to continue to run the file system check. The write operation will be retried with the failed blocks indicated by the message:

```
THE FOLLOWING SECTORS COULD NOT BE WRITTEN: N
```

where *N* indicates the sectors that could not be written. If the disk is experiencing hardware problems, the problem will persist. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` will terminate with the message

```
Fatal I/O error
```

NO

Terminate the program.

**Message**

```
bad inode number DDD to ginode
```

An internal error has attempted to read non-existent inode *DDD*. This error causes `fsck` to exit. See a guru.

### Phase 1 – Check Blocks and Sizes

This phase concerns itself with the inode list. This section lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format. All errors in this phase except `INCORRECT BLOCK COUNT` and `PARTIALLY TRUNCATED INODE` are fatal if the file system is being preened.

**Message**

```
UNKNOWN FILE TYPE I=I (CLEAR)
```

The mode word of the inode *I* indicates that the inode is not a special block inode, special character inode, socket inode, regular inode, symbolic link, FIFO file, or directory inode.

Possible responses to the `CLEAR` prompt are:

YES

De-allocate inode *I* by zeroing its contents. This will always invoke the `UNALLOCATED` error condition in Phase 2 for each directory entry pointing to this inode.

NO

Ignore this error condition.

**Message**

```
PARTIALLY TRUNCATED INODE I=I (SALVAGE)
```

`fsck` has found inode *I* whose size is shorter than the number of blocks allocated to it. This condition should only occur if the system crashes while in the midst of truncating a file. When preening the file system, `fsck` completes the truncation to the specified size.

Possible responses to the SALVAGE prompt are:

YES

Complete the truncation to the size specified in the inode.

NO

Ignore this error condition.

**Message**

```
LINK COUNT TABLE OVERFLOW (CONTINUE)
```

An internal table for `fsck` containing allocated inodes with a link count of zero cannot allocate more memory. Increase the virtual memory for `fsck`.

Possible responses to the CONTINUE prompt are:

YES

Continue with the program. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If another allocated inode with a zero link count is found, this error condition is repeated.

NO

Terminate the program.

**Message**

```
B BAD I=I
```

Inode *I* contains block number *B* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the EXCESSIVE BAD BLKS error condition in Phase 1 (see next message) if inode *I* has too many block numbers outside the file system range. This error condition will always invoke the BAD/DUP error condition in Phase 2 and Phase 4.

**Message**

```
EXCESSIVE BAD BLKS I=I (CONTINUE)
```

There is more than a tolerable number (usually 10) of blocks with a number lower than the number of the first data block in the file system or greater than the number of last block in the file system associated with inode *I*.

Possible responses to the CONTINUE prompt are:

YES

Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system.

NO

Terminate the program.

**Message**

```
BAD STATE DDD TO BLKERR
```

An internal error has scrambled `fsck`'s state map to have the impossible value *DDD*. `fsck` exits immediately. See a guru.

**Message**

```
B DUP I=I
```

Inode *I* contains block number *B* that is already claimed by another inode. This error condition may invoke the `EXCESSIVE DUP BLKS` error condition in Phase 1 if inode *I* has too many block numbers claimed by other inodes. This error condition will always invoke Phase 1b and the `BAD/DUP` error condition in Phase 2 and Phase 4.

**Message**

```
EXCESSIVE DUP BLKS I=I (CONTINUE)
```

There is more than a tolerable number (usually 10) of blocks claimed by other inodes.

Possible responses to the `CONTINUE` prompt are:

YES

> Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system.

NO

> Terminate the program.

**Message**

```
DUP TABLE OVERFLOW (CONTINUE)
```

An internal table in `fsck` containing duplicate block numbers cannot allocate any more space. Increase the amount of virtual memory available to `fsck`.

Possible responses to the `CONTINUE` prompt are:

YES

> Continue with the program. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If another duplicate block is found, this error condition will repeat.

NO

> Terminate the program.

**Message**

```
PARTIALLY ALLOCATED INODE I=I (CLEAR)
```

Inode *I* is neither allocated nor unallocated.

Possible responses to the `CLEAR` prompt are:

YES
: De-allocate inode *I* by zeroing its contents.

NO
: Ignore this error condition.

**Message**

```
INCORRECT BLOCK COUNT I=I (X should be Y) (CORRECT)
```

The block count for inode *I* is *X* blocks, but should be *Y* blocks. When preening the count is corrected.

Possible responses to the `CORRECT` prompt are:

YES
: Replace the block count of inode *I* with *Y*.

NO
: Ignore this error condition.

## Phase 1B: Rescan for More Dup's

When a duplicate block is found in the file system, the file system is re-scanned to find the inode that previously claimed that block. This section lists the error condition when the duplicate block is found.

**Message**

```
B DUP I=I
```

Inode *I* contains block number *B* that is already claimed by another inode. This error condition will always invoke the `BAD/DUP` error condition in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the DUP error condition in Phase 1.

## Phase 2 – Check Pathnames

This phase concerns itself with removing directory entries pointing to error conditioned inodes from Phase 1 and Phase 1b. This section lists error conditions resulting from root inode mode and status, directory inode pointers in range, directory entries pointing to bad inodes, and directory integrity checks. All errors in this phase are fatal if the file system is being preened, except for directories not being a multiple of the blocks size and extraneous hard links.

**Message**

```
ROOT INODE UNALLOCATED (ALLOCATE)
```

The root inode (usually inode number 2) has no allocate mode bits. This should never happen.

Possible responses to the ALLOCATE prompt are:

YES

Allocate inode 2 as the root inode. The files and directories usually found in the root will be recovered in Phase 3 and put into lost+found. If the attempt to allocate the root fails, fsck will exit with the message

```
CANNOT ALLOCATE ROOT INODE
```

NO

Terminate the program.

**Message**

```
ROOT INODE NOT DIRECTORY (REALLOCATE)
```

The root inode (usually inode number 2) is not directory inode type.

Possible responses to the REALLOCATE prompt are:

YES

Clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root will be recovered in Phase 3 and put into lost+found. If the attempt to allocate the root fails, fsck will exit with the message:

```
CANNOT ALLOCATE ROOT INODE
```

NO

fsck will then prompt with FIX

Possible responses to the FIX prompt are:

YES

Replace the root inode's type to be a directory. If the root inode's data blocks are not directory blocks, many error conditions will be produced.

NO

Terminate the program.

**Message**

```
DUPS/BAD IN ROOT INODE (REALLOCATE)
```

Phase 1 or Phase 1b have found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system.

Possible responses to the REALLOCATE prompt are:

YES

Clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root will be recovered in Phase 3 and put into `lost+found`. If the attempt to allocate the root fails, `fsck` will exit with the message:

```
CANNOT ALLOCATE ROOT INODE
```

NO

`fsck` will then prompt with `CONTINUE`.

Possible responses to the `CONTINUE` prompt are:

YES

Ignore the `DUPS/BAD` error condition in the root inode and attempt to continue to run the file system check. If the root inode is not correct, then this may result in many other error conditions.

NO

Terminate the program.

**Message**

```
NAME TOO LONG F
```

An excessively long path name has been found. This usually indicates loops in the file system name space. This can occur if the super user has made circular links to directories. The offending links must be removed (by a guru).

**Message**

```
I OUT OF RANGE I=I NAME=F (REMOVE)
```

A directory entry *F* has an inode number *I* that is greater than the end of the inode list.

Possible responses to the `REMOVE` prompt are:

YES

Remove the directory entry *F*.

NO

Ignore this error condition.

**Message**

```
UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T type=F (REMOVE)
```

A directory or file entry *F* points to an unallocated inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and name *F* are printed.

Possible responses to the `REMOVE` prompt are:

YES

Remove the directory entry *F*.

NO

Ignore this error condition.

**Message**

```
DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T type=F (REMOVE)
```

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory or file entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the `REMOVE` prompt are:

YES

Remove the directory entry *F*.

NO

Ignore this error condition.

**Message**

```
ZERO LENGTH DIRECTORY I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
(REMOVE)
```

A directory entry *F* has a size *S* that is zero. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the `REMOVE` prompt are:

YES

Remove the directory entry *F*; this will always invoke the `BAD/DUP` error condition in Phase 4.

NO

Ignore this error condition.

**Message**

```
DIRECTORY TOO SHORT I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *F* has been found whose size *S* is less than the minimum size directory. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the `FIX` prompt are:

YES

Increase the size of the directory to the minimum directory size.

NO

Ignore this directory.

**Message**

```
DIRECTORY F LENGTH S NOT MULTIPLE OF B (ADJUST)
```

A directory *F* has been found with size *S* that is not a multiple of the directory block size *B*.

Possible responses to the `ADJUST` prompt are:

YES

Round up the length to the appropriate block size. This error can occur on SunOS file systems prior to release 4.0. Thus when preening the file system only a warning is printed and the directory is adjusted.

NO

Ignore the error condition.

**Message**

```
DIRECTORY CORRUPTED I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
(SALVAGE)
```

A directory with an inconsistent internal state has been found.

Possible responses to the `FIX` prompt are:

YES

Throw away all entries up to the next directory boundary (usually a 512-byte boundary). This drastic action can throw away up to 42 entries, and should be taken only after other recovery efforts have failed.

NO

Skip up to the next directory boundary and resume reading, but do not modify the directory.

**Message**

```
BAD INODE NUMBER FOR '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
(FIX)
```

A directory *I* has been found whose inode number for '.' does does not equal *I*.

Possible responses to the `FIX` prompt are:

YES

Change the inode number for '.' to be equal to *I*.

NO

Leave the inode number for '.' unchanged.

**Message**

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found whose first entry is unallocated.

Possible responses to the `FIX` prompt are:

YES

Build an entry for '.' with inode number equal to *I*.

NO

Leave the directory unchanged.

**Message**

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, FIRST ENTRY IN DIRECTORY CONTAINS F
```

A directory *I* has been found whose first entry is *F*. `fsck` cannot resolve this problem. The file system should be mounted and the offending entry *F* moved elsewhere. The file system should then be unmounted and `fsck` should be run again.

**Message**

```
MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'
```

A directory *I* has been found whose first entry is not '.'. `fsck` cannot resolve this problem as it should never happen. See a guru.

**Message**

```
EXTRA '.' ENTRY I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found that has more than one entry for '.'.

Possible responses to the `FIX` prompt are:

YES

Remove the extra entry for '.'.

NO

Leave the directory unchanged.

**Message**

```
BAD INODE NUMBER FOR '..' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
(FIX)
```

A directory *I* has been found whose inode number for '..' does does not equal the parent of *I*.

Possible responses to the `FIX` prompt are:

YES

Change the inode number for '..' to be equal to the parent of *I* ("..'' in the root inode points to itself).

NO

Leave the inode number for '..' unchanged.

**Message**

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found whose second entry is unallocated.

Possible responses to the `FIX` prompt are:

YES
: Build an entry for '..' with inode number equal to the parent of *I* ("..'' in the root inode points to itself).

NO
: Leave the directory unchanged.

**Message**

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, SECOND ENTRY IN DIRECTORY CONTAINS F
```

A directory *I* has been found whose second entry is *F*. `fsck` cannot resolve this problem. The file system should be mounted and the offending entry *F* moved elsewhere. The file system should then be unmounted and `fsck` should be run again.

**Message**

```
MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F
CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'
```

A directory *I* has been found whose second entry is not '..'. `fsck` cannot resolve this problem. The file system should be mounted and the second entry in the directory moved elsewhere. The file system should then be unmounted and `fsck` should be run again.

**Message**

```
EXTRA '..' ENTRY I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (FIX)
```

A directory *I* has been found that has more than one entry for '..'.

Possible responses to the `FIX` prompt are:

YES
: Remove the extra entry for '..'.

NO
: Leave the directory unchanged.

**Message**

```
N IS AN EXTRANEOUS HARD LINK TO A DIRECTORY D (REMOVE)
```

`fsck` has found a hard link, *N*, to a directory, *D*. When preening the extraneous links are ignored.

Possible responses to the REMOVE prompt are:

YES

Delete the extraneous entry, *N*.

NO

Ignore the error condition.

**Message**

```
BAD INODE S TO DESCEND
```

An internal error has caused an impossible state *S* to be passed to the routine that descends the file system directory structure. fsck exits. See a guru.

**Message**

```
BAD RETURN STATE S FROM DESCEND
```

An internal error has caused an impossible state *S* to be returned from the routine that descends the file system directory structure. fsck exits. See a guru.

**Message**

```
BAD STATE S FOR ROOT INODE
```

An internal error has caused an impossible state *S* to be assigned to the root inode. fsck exits. See a guru.

### Phase 3 – Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. This section lists error conditions resulting from unreferenced directories, and missing or full lost+found directories.

**Message**

```
UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)
```

The directory inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. When preening, the directory is reconnected if its size is non-zero; otherwise it is cleared.

Possible responses to the RECONNECT prompt are:

YES

Reconnect directory inode *I* to the file system in the directory for lost files (usually lost+found). This may invoke the lost+found error condition in Phase 3 if there are problems connecting directory inode *I* to lost+found. This may also invoke the CONNECTED error condition in Phase 3 if the link was successful.

NO

Ignore this error condition. This will always invoke the UNREF error condition in Phase 4.

**Message**

```
NO lost+found DIRECTORY (CREATE)
```

There is no lost+found directory in the root directory of the file system; When preening fsck tries to create a lost+found directory.

Possible responses to the CREATE prompt are:

YES

Create a lost+found directory in the root of the file system. This may raise the message:

```
NO SPACE LEFT IN / (EXPAND)
```

See below for the possible responses. Inability to create a lost+found directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

**Message**

```
lost+found IS NOT A DIRECTORY (REALLOCATE)
```

The entry for lost+found is not a directory.

Possible responses to the REALLOCATE prompt are:

YES

Allocate a directory inode, and change lost+found to reference it. The previous inode reference by the lost+found name is not cleared. Thus it will either be reclaimed as an UNREF'ed inode or have its link count ADJUST'ed later in this phase. Inability to create a lost+found directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

**Message**

```
NO SPACE LEFT IN /lost+found (EXPAND)
```

There is no space to add another entry to the `lost+found` directory in the root directory of the file system. When preening the `lost+found` directory is expanded.

Possible responses to the `EXPAND` prompt are:

YES

> Expand the `lost+found` directory to make room for the new entry. If the attempted expansion fails `fsck` prints the message:
>
> ```
> SORRY. NO SPACE IN lost+found DIRECTORY
> ```
>
> and aborts the attempt to linkup the lost inode. This will always invoke the `UNREF` error condition in Phase 4. Clean out unnecessary entries in `lost+found`. This error is fatal if the file system is being preened.

NO

> Abort the attempt to linkup the lost inode. This will always invoke the `UNREF` error condition in Phase 4.

**Message**

```
DIR I=I1 CONNECTED. PARENT WAS I=I2
```

This is an advisory message indicating a directory inode *I1* was successfully connected to the `lost+found` directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the `lost+found` directory.

**Message**

```
DIRECTORY F LENGTH S NOT MULTIPLE OF B (ADJUST)
```

A directory *F* has been found with size *S* that is not a multiple of the directory block size *B* (this can reoccur in Phase 3 if it is not adjusted in Phase 2).

Possible responses to the `ADJUST` prompt are:

YES

> Round up the length to the appropriate block size. This error can occur on SunOS file systems prior to release 4.0. Thus when preening the file system only a warning is printed and the directory is adjusted.

NO

> Ignore the error condition.

**Message**

```
BAD INODE S TO DESCEND
```

An internal error has caused an impossible state *S* to be passed to the routine that descends the file system directory structure. `fsck` exits. See a guru.

## Phase 4 – Check Reference Counts

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This section lists error conditions resulting from unreferenced files, missing or full `lost+found` directory, incorrect link counts for files, directories, symbolic links, or special files, unreferenced files, symbolic links, and directories, and bad or duplicate blocks in files, symbolic links, and directories. All errors in this phase are correctable if the file system is being preened except running out of space in the `lost+found` directory.

**Message**

```
UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)
```

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When preening the file is cleared if either its size or its link count is zero, otherwise it is reconnected.

Possible responses to the `RECONNECT` prompt are:

YES

Reconnect inode *I* to the file system in the directory for lost files (usually `lost+found`). This may invoke the `lost+found` error condition in Phase 4 if there are problems connecting inode *I* to `lost+found`.

NO

Ignore this error condition. This will always invoke the `CLEAR` error condition in Phase 4.

**Message**

```
  (CLEAR)
```

The inode mentioned in the immediately previous error condition cannot be reconnected. This cannot occur if the file system is being preened, since lack of space to reconnect files is a fatal error.

Possible responses to the `CLEAR` prompt are:

YES

De-allocate the inode mentioned in the immediately previous error condition by zeroing its contents.

NO

Ignore this error condition.

**Message**

```
  NO lost+found DIRECTORY (CREATE)
```

There is no `lost+found` directory in the root directory of the file system; When preening `fsck` tries to create a `lost+found` directory.

Possible responses to the CREATE prompt are:

YES

Create a lost+found directory in the root of the file system. This may raise the message:

```
NO SPACE LEFT IN / (EXPAND)
```

See below for the possible responses. Inability to create a lost+found directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

**Message**

```
lost+found IS NOT A DIRECTORY (REALLOCATE)
```

The entry for lost+found is not a directory.

Possible responses to the REALLOCATE prompt are:

YES

Allocate a directory inode, and change lost+found to reference it. The previous inode reference by the lost+found name is not cleared. Thus it will either be reclaimed as an UNREF'ed inode or have its link count ADJUST'ed later in this phase. Inability to create a lost+found directory generates the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

NO

Abort the attempt to linkup the lost inode. This will always invoke the UNREF error condition in Phase 4.

**Message**

```
NO SPACE LEFT IN /lost+found (EXPAND)
```

There is no space to add another entry to the lost+found directory in the root directory of the file system. When preening the lost+found directory is expanded.

Possible responses to the `EXPAND` prompt are:

YES

Expand the `lost+found` directory to make room for the new entry. If the attempted expansion fails `fsck` prints the message:

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

and aborts the attempt to linkup the lost inode. This will always invoke the `UNREF` error condition in Phase 4. Clean out unnecessary entries in `lost+found`. This error is fatal if the file system is being preened.

NO

Abort the attempt to linkup the lost inode. This will always invoke the `UNREF` error condition in Phase 4.

**Message**

```
LINK COUNT type I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X
SHOULD BE Y (ADJUST)
```

The link count for inode *I* is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed. When preening the link count is adjusted unless the number of references is increasing, a condition that should never occur unless precipitated by a hardware failure. When the number of references is increasing under preen mode, `fsck` exits with the message:

```
LINK COUNT INCREASING
```

Possible responses to the `ADJUST` prompt are:

YES

Replace the link count of file inode *I* with *Y*.

NO

Ignore this error condition.

**Message**

```
UNREF type I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)
```

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When preening, this is a file that was not connected because its size or link count was zero, hence it is cleared.

Possible responses to the `CLEAR` prompt are:

YES

De-allocate inode *I* by zeroing its contents.

NO

Ignore this error condition.

**Message**

```
BAD/DUP type I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)
```

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. This error cannot arise when the file system is being preened, as it would have caused a fatal error earlier.

Possible responses to the `CLEAR` prompt are:

YES

De-allocate inode *I* by zeroing its contents.

NO

Ignore this error condition.

## Phase 5 - Check Cyl Groups

This phase concerns itself with the free block and used inode maps. This section lists error conditions resulting from allocated blocks in the free block maps, free blocks missing from free block maps, and the total free block count incorrect. It also lists error conditions resulting from free inodes in the used inode maps, allocated inodes missing from used inode maps, and the total used inode count incorrect.

**Message**

```
CG C: BAD MAGIC NUMBER
```

The magic number of cylinder group *C* is wrong. This usually indicates that the cylinder group maps have been destroyed. When running manually, the cylinder group is marked as needing to be reconstructed. This error is fatal if the file system is being preened.

**Message**

```
BLK(S) MISSING IN BIT MAPS (SALVAGE)
```

A cylinder group block map is missing some free blocks. During preening the maps are reconstructed.

Possible responses to the `SALVAGE` prompt are:

YES

Reconstruct the free block map.

NO

Ignore this error condition.

**Message**

```
SUMMARY INFORMATION BAD (SALVAGE)
```

The summary information was found to be incorrect. When preening, the summary information is recomputed.

Possible responses to the `SALVAGE` prompt are:

YES
: Reconstruct the summary information.

NO
: Ignore this error condition.

**Message**

```
FREE BLK COUNT(S) WRONG IN SUPERBLOCK (SALVAGE)
```

The superblock free block information was found to be incorrect. When preening, the superblock free block information is recomputed.

Possible responses to the `SALVAGE` prompt are:

YES
: Reconstruct the superblock free block information.

NO
: Ignore this error condition.

## Cleanup

Once a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system and modify status of the file system.

**Message**

```
V files, W used, X free (Y frags, Z blocks)
```

This is an advisory message indicating that the file system checked contained *V* files using *W* fragment sized blocks, leaving *X* fragment sized blocks free in the file system. The numbers in parenthesis break the free count down into *Y* free fragments and *Z* free full sized blocks.

**Message**

```
***** REBOOT THE SYSTEM *****
```

This is an advisory message indicating that the root file system has been modified by `fsck`. If SunOS is not rebooted immediately, the work done by `fsck` may be undone by the in-core copies of tables SunOS keeps. When preening, `fsck` will exit with a code of 4. The auto-reboot script interprets an exit code of 4 by issuing a reboot system call.

**Message**

```
***** FILE SYSTEM WAS MODIFIED *****
```

This is an advisory message indicating that the current file system was modified by `fsck`. If this file system is mounted or is the current root file system, `fsck` should be halted and SunOS rebooted. If SunOS is not rebooted immediately, the work done by `fsck` may be undone by the in-core copies of tables SunOS keeps.

## 17.4. References

[Dolotta78] Dolotta, T. A., and Olsson, S. B. eds., *UNIX User's Manual, Edition 1.1*, January 1978.

[Joy83] Joy, W., Cooper, E., Fabry, R., Leffler, S., McKusick, M., and Mosher, D. *System Interface Overview*, California*University*of *Computer Systems Research Group Technical Report* #4, 1982.

[McKusick84] McKusick, M., Joy, W., Leffler, S., and Fabry, R. A Fast File System for UNIX, University of California at Berkeley, *ACM Transactions on Computer Systems* 2, 3. pp. 181-197, August 1984.

[Ritchie78] Ritchie, D. M., and Thompson, K., The UNIX Time-Sharing System, *The Bell System Technical Journal* 57, 6 (July-August 1978, Part 2), pp. 1905-29.

[Thompson78] Thompson, K., UNIX Implementation, *The Bell System Technical Journal* 57, 6 (July-August 1978, Part 2), pp. 1931-46.

# 18

# Sendmail Installation and Operation

# 18

# Sendmail Installation and Operation

sendmail implements a general internetwork mail transport facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration. To do this, it uses a configuration file. Sun supplies standard configuration files that most sites can use. Other sites might have to adjust the file to fit any unusual circumstances.

sendmail can use different types of communications protocols, such as TCP/RFC822, and UUCP. It also implements an SMTP server, message queueing, and mailing lists. Name interpretation is controlled by a pattern-matching system that can handle both domain-based naming and *ad hoc* conventions.

Sections in this chapter discuss the following subjects:

- How to do a basic sendmail installation.
- Day-to-day information you need to know to maintain your mail system. If you have a relatively typical site, these two topics contain sufficient information for you to install sendmail and keep it running smoothly.
- Command line arguments to sendmail.
- sendmail parameters that you can alter.
- In-depth information on the configuration file. This section provides information for those sites that need to write their own configuration file. Sections
- Brief but detailed explanation of several lesser-used features of sendmail.

For information on gateways and installing an internetwork, refer to Chapter 12, "The SunOS Networking Environment."

The domain technique separates the issue of physical versus logical naming. The top-level domains used today on the internetwork include EDU, COM, MIL, and GOV. For example, a name of the form "nowicki@sun.COM" describes only the logical organization, not the physical connection. sendmail can accept old arbitrary name syntaxes, resolving ambiguities using heuristics specified by the system administrator, as well as domain-based naming. It helps to convert messages between disparate naming schemes.

Certain special cases can be handled by *ad hoc* techniques, such as providing network names that appear local to hosts on other networks. For example, "user@host" is left to right syntax, and "host!host" is right to left syntax.

## 18.1. Design Goals

Design goals for `sendmail` include:

1. Including System V mail, Bell version 7 mail [UNIX83], Internet mail [Crocker77a, Postel77].

2. Reliability: every message should be correctly delivered so that it can be disposed of. This way no message should ever be completely lost. This goal was considered essential because of the emphasis on mail in our environment. For example, sometimes sites generate improperly formatted names, causing error-message loops.

3. Existing software to do actual delivery should be used whenever possible.

4. Easy expansion to fairly complex environments, including multiple connections to a single network type (such as with UUCP or Ethernets). This goal requires consideration of the contents of a name as well as its syntax to determine which mailer to use.

5. Configuration should not be compiled into the code. A single compiled program should work at most sites. Besides the simple problems that occur when any program gets recompiled in a different environment, many sites would probably alter anything that they will be recompiling anyway.

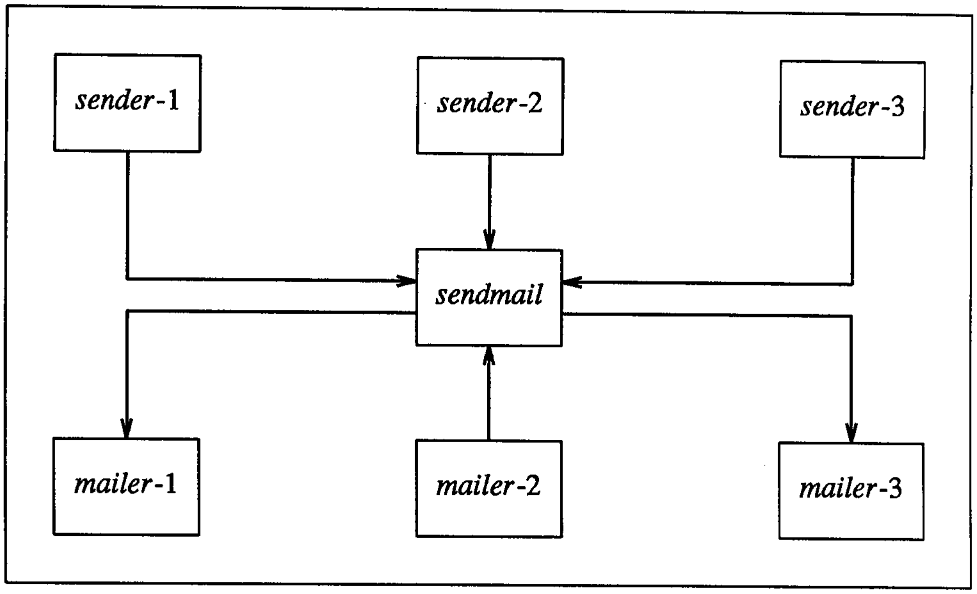
6. `sendmail` must be able to let various groups maintain their own mailing lists, and let individuals specify their own forwarding without modifying the domain-wide alias file (which is typically located in the domain-wide aliases in the yp database).

7. Each user should be able to specify a custom mailer to process mail being delivered for him. This feature allows functions such as returning an "I am on vacation" message. Refer to `vacation(1)` in the *SunOS Reference Manual*

8. Network traffic should be minimized by batching recipients to a single host where possible, without assistance from the user.

These goals motivated the architecture illustrated in Figure 1.

Figure 18-1 *Sendmail System Structure*



The user interacts with a mail generating and sending program. When the mail is submitted, the generator calls `sendmail`, which routes the message to the correct mailer(s). Since some of the senders may be network servers and some of the mailers may be network clients, `sendmail` may be used as an internet mail gateway.

## 18.2. `sendmail` Overview

`sendmail` neither interfaces with the user nor does actual mail delivery. Rather, it collects a message from a program such as *Mail*, MS [Crocker77b], or MH [Borden79], edits the message as required by the destination mailer, and calls appropriate mailers to do delivery or queueing for network transmission[3]. This discipline allows the insertion of new mailers at minimum cost. In this sense `sendmail` resembles the Message Processing Module (MPM) of [Postel79b].

### Interfaces to the Outside World

There are three ways `sendmail` can communicate with the outside world, both in receiving and in sending mail. These are using the conventional UNIX argument vector/return status, speaking over a pair of UNIX pipes, and speaking SMTP over a TCP connection.

#### Argument vector/exit status

The argument vector (command name and arguments) is the standard UNIX method of communicating with a process. A list of recipients is sent in the argument vector, and the message body is sent on the standard input. Anything that the mailer prints is simply collected and sent back to the sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

[3] except when mailing to a file, when `sendmail` does the delivery directly.

### SMTP over Pipes

The SMTP protocol can be used to run an interactive lock-step interface with the mailer. A subprocess is still created, but no recipient names are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the processes' standard input. Anything appearing on the standard output must be a standard SMTP reply code.

### SMTP over a TCP Connection

This technique is similar to the previous technique, except that it uses a TCP connection. Refer to Section 4 of the *SunOS Reference Manual.* This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a `sendmail` process on another machine.

### How `sendmail` Works

When a sender wants to send a message, it issues a request to `sendmail` using one of the three methods described above. `sendmail` operates as described in the steps below.

### Argument Processing and Address Parsing

If `sendmail` is called using one of the first two techniques above, the arguments are first scanned and option specifications are processed. Recipient names are then collected, either from the command line or from the SMTP command, and a list of recipients is created. Aliases are expanded at this step, including mailing lists. As much validation as possible of the remote recipient is done at this step: syntax is checked, and local recipients are verified, but detailed checking of host names is deferred until delivery. Forwarding is also performed as the local recipients are verified.

`sendmail` appends each name to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list, and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates, preventing alias loops and duplicate messages delivered to the same recipient, as might occur if a person is in two groups.

### Message Collection

`sendmail` then collects the message. The message should have a header at the beginning. No formatting requirements are imposed on the message body except that they must be lines of text (in other words, binary data is not allowed). The header is stored in memory, and the body of the message is saved in a temporary file.

To simplify the program interface, the message is collected even if no names were valid. The message will be returned with an error.

### Message Delivery

For each unique mailer and host in the recipient list, `sendmail` calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that only accept one recipient at a time are handled properly.

The message is sent to the mailer using one of the same three interfaces used to submit a message to sendmail. Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and a suitable error message given as appropriate. The exit code must conform to a system standard or a generic message ("Service unavailable") is given.

Queueing for Retransmission

If the mailer returned a status that indicated that it might be able to handle the mail later, `sendmail` will queue the mail and try again later.

Return to Sender

If errors occur during processing, `sendmail` returns the message to the sender for retransmission. The letter can be mailed back or written in the `dead.letter` file in the sender's home directory[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.

**Configuration File**

Almost all configuration information is read at runtime from a text file, encoding macro definitions (defining the value of macros used internally), header declarations (the format of header lines that it will process specially, and lines that it will add or reformat), mailer definitions (giving information such as the location and characteristics of each mailer), and name rewriting rules (a limited pattern-matching system used to rewrite names).

## 18.3. How to Implement and Use `sendmail`

Following flag arguments, recipient name arguments may be given, unless you run in SMTP mode. In brief, the format of recipient names is:

1. Anything in parentheses is thrown away (as a comment).
2. Anything in angle brackets ("<>") is preferred over anything else. This rule implements the ARPANET standard that names of the form

   **user name <machine-name>**

   will send to the electronic "machine-name" rather than the human "user name."
3. Double quotes ( Backslashes are more powerful in that they will cause otherwise equivalent phrases to compare differently — for example, `user` and are equivalent, but `\user` is different from either of them.

Parentheses, angle brackets, and double quotes must be properly balanced and nested. The rewriting rules control the rest of processing that needs to be done.

[4] Obviously, if the site giving the error is not the originating site, the only reasonable option is to mail back to the sender. Also, there are many more error disposition options, but they only effect the error message — the "return to sender" function is always handled in one of these two ways.

### Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to maintain a public repository of systems messages such as the Usenet news system.[5]

Any name passing through the initial parsing algorithm as a local name is scanned for two special cases. If prefixed by a vertical bar (“|”) the rest of the name is processed as a shell command. If the user name begins with a slash mark (“/”) the name is used as a filename, instead of a login name.

### Message Collection

Once all recipient names are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body, separated by a blank line.

The header is formatted as a series of lines of the form

```
field-name: field-value
```

For example, a sample header might be:

```
From:  John Smith <Smith@Podunk.edu>
```

Field-value can be split across lines by starting the following lines with a space or a tab. Some header fields have special internal meaning, and have appropriate special processing. Other headers are simply passed through. Some header fields may be added automatically, such as time stamps.

The body is a series of text lines. It is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

### Message Delivery

The send queue is grouped by receiving host before transmission to implement message batching. An argument list is built as the scan proceeds. Mail to files is detected during the scan of the send list. The interface to the mailer is performed using one of the techniques described in section 18.2.

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.

### Queued Messages

If the mailer returns a “temporary failure” exit status, the message is queued. A control file is used to describe the recipients to be sent to and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other parameter of the message. The header of the message is stored in the control file so that the associated data file in the queue is just the temporary file that was originally collected.

[5] Not provided by Sun.

## Configuration Overview

Configuration is controlled primarily by a configuration file read at startup. Adding mailers or changing the rewriting or routing information does not require recompilation. The configuration file encodes macro definitions, header definitions, mailer definitions, rewriting rules, and options.

### Macros

Macros can be used in three ways. Certain macros transmit unstructured textual information into the mail system, such as the name `sendmail` will use to identify itself in error messages. Other macros are unused internally, and can be used as shorthand in the configuration file.

### Header Declarations

Header declarations inform `sendmail` of the format of known header lines. Knowledge of a few header lines is built into `sendmail`, such as the ''From:'' and ''Date:'' lines.

Most configured headers will be automatically inserted in the outgoing message if they don't exist in the incoming message. Certain headers are suppressed by some mailers.

### Mailer Declarations

Mailer declarations specify the internal name of the mailer; some flags associated with the mailer, and an argument vector to be used on the call; this vector is macro-expanded before use.

### Address Rewriting Rules

The heart of name parsing in `sendmail`. These are an ordered list of pattern-replacement rules, which are applied to each name. In particular, rule set zero determines which mailer to use. The name is rewritten until it is either rewritten into a special canonical form — for example, a (mailer, host, user) triple, such as {ddn, isi.edu, postel} representing the name "postel@isi.edu" — or it falls off the end. When a pattern matches, the rule is reapplied until it fails.

The configuration file also supports the editing of names into different formats. For example, a name of the form:

```
ucsfcgl!tef
```

might be mapped into:

```
tef@ucsfcgl.UUCP
```

to conform to the internal syntax. Translations can also be done in the other direction for particular mailers.

### Option Setting

There are several options that can be set from the configuration file. These include the pathnames of various support files, timeouts, default modes, etc.

## 18.4. Basic Installation

There are two basic steps to installing `sendmail`. The first, and hardest, part is to build the configuration file. The configuration file describes the mailers it knows about, how to parse names, how to rewrite the message header, and the settings of various options. Each time `sendmail` starts up, it reads the configuration file. Although the configuration file is complex, you can usually build a configuration by adjusting an existing off-the-shelf configuration. The second step is to create and install the necessary files.

**sendmail Files**

You can produce your own sendmail.cf file from one of the generic configuration files provided with this release: /usr/lib/sendmail.main.cf or /usr/lib/sendmail.subsidiary.cf. If your machine is the primary mail handling machine for your site, copy /usr/lib/sendmail.main.cf into a file named /etc/sendmail.cf. For all other machines in your organization you would copy /usr/lib/sendmail.subsidiary.cf to /etc/sendmail.cf. Refer to the procedures in Chapter 15 of of this manual. Then the suninstall program should perform all other installation for you.

/usr/lib/sendmail

/usr/lib/sendmail is the actual routing program for mail. /usr/lib/sendmail receives mail from user interface programs and uses the information found in the configuration files to direct the mail messages. This file contains the binary of sendmail.

/etc/sendmail.cf

This is the configuration file used by /usr/lib/sendmail to set up the appropriate environment and conditions. The configuration file can be tailored for your particular machine. Here are three subsets of the configuration file:

/usr/lib/sendmail.main.cf
The default configuration file for main machines, in text form.

/usr/lib/sendmail.subsidiary.cf
The default configuration file for subsidiary machines, in text form.

/usr/lib/sendmail.fc
The frozen configuration file.

/var/spool/mqueue

The directory /var/spool/mqueue holds the mail queue. The default is for /usr/lib/sendmail to have the setuid bit set and /var/spool/mqueue to be owned by root with mode 755. If this is not the case, /var/spool/mqueue needs to be mode 777. Mail that has not been delivered yet is stored here. The mail messages are broken into two files: a qf file, which contains the control information, and the df file, which contains the body of the message. Delivered mail is stored in /var/spool/mail/login_name.

/var/spool/mqueue
The directory in which the mail queue and temporary files reside.

/var/spool/mqueue/qf*
Control (queue) files for messages.

/var/spool/mqueue/df*
Data files.

/var/spool/mqueue/lf*
Lock files

/var/spool/mqueue/tf*
Temporary versions of the qf files, used during queue file rebuild.

`/var/spool/mqueue/nf*`
A file used when creating a unique id.

`/var/spool/mqueue/xf*`
A transcript of the current session.

`/etc/aliases*`

`/etc/aliases` is the optional aliases file. You can use the aliases in this file to redirect mail to another machine, send mail to a group of people, and to redirect mail from nicknames or commonly misspelled user names to the correct user. Typically, you might also use the domain-wide or "global aliases" found in the yellow pages database. (The global aliases are usually located in `/etc/aliases` on the master `yp` server.)

`/etc/aliases`
The textual version of the alias file. (Also referred to as "local aliases.")

`/etc/aliases.{pag,dir}`
The alias file in dbm(3) format.

`/usr/ucb/newaliases`
The `newaliases` command updates the alias database, from the `/etc/aliases` file, which then can be found in the `/etc/aliases.dir` and `/etc/aliases.pag` files. You can adjust the configuration file to have this done automatically. This is really just a link to `/usr/lib/sendmail`. Running this program is equivalent to giving `sendmail` the `-bi` flag.

`/etc/rc.local`

The `sendmail` daemon may need to be started when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically The following lines should be in `/etc/rc.local` to start up the daemons:

```
if [ -f /etc/sendmail -a -f /usr/lib/sendmail.cf ]; then
    (cd /var/spool/mqueue; rm -f nf* lf*)
    /usr/lib/sendmail -bd -q1h & echo -n ' sendmail'    >/dev/console
fi
```

The `cd` and `rm` commands ensure that all lock files have been removed; expendable lock files may be left around if the system goes down in the middle of processing a message. The line that actually invokes `sendmail` has two flags: `-bd` causes it to listen on the SMTP port for incoming mail from other machines, `-q1h` causes it to run the queue every hour. The queue on a *mailhost* is run at shorter intervals, typically every 15 minutes (the flag in this case would be -q15m).

`/usr/lib/sendmail.hf` This help file is not intended to be used by the system administrator, but by the SMTP **HELP** command. The file is already installed in the distribution.

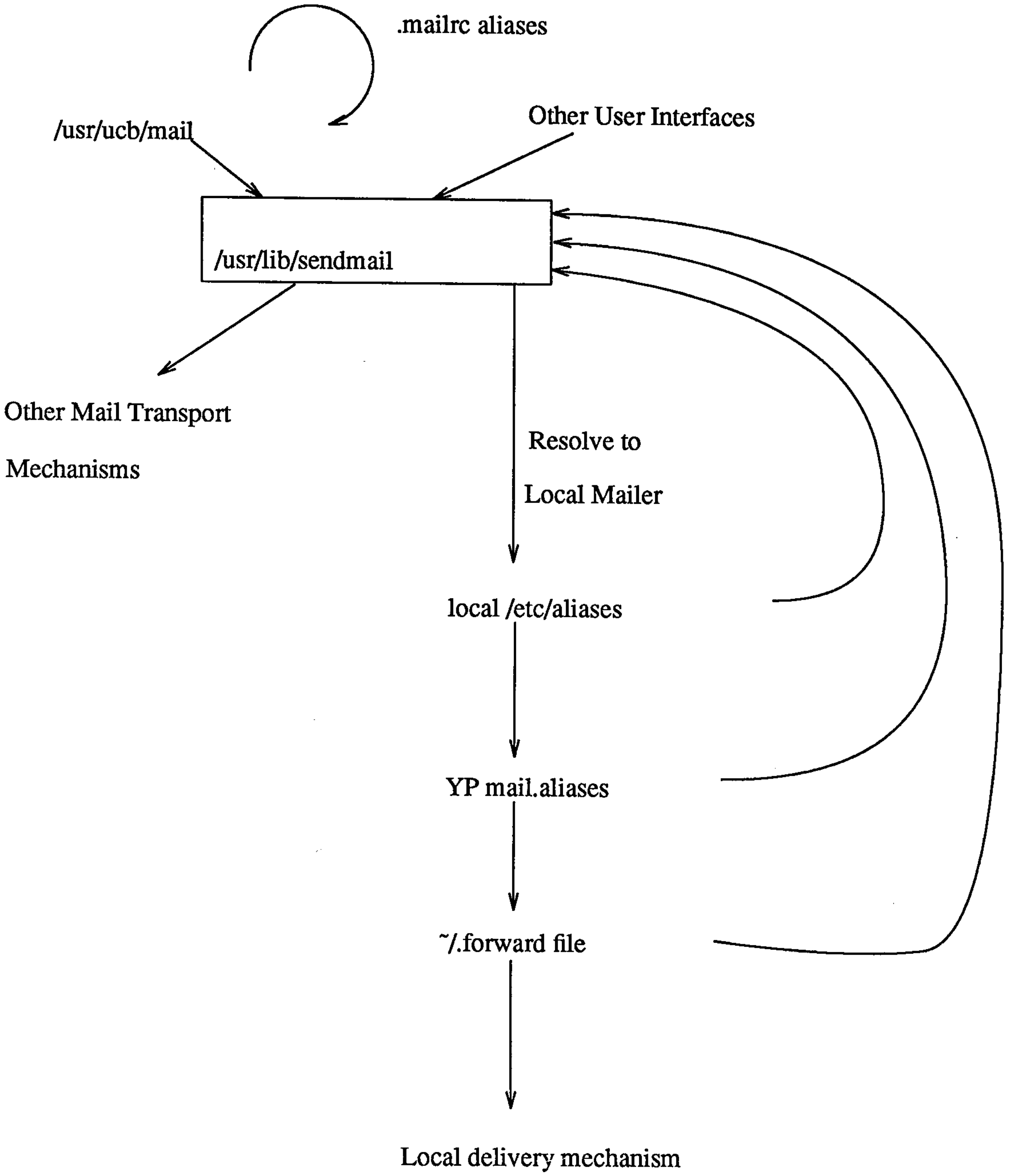
`/var/spool/mail` Mailbox directory that can be mounted from a mailbox server using NFS.

## 18.5. Aliases, Forwarding, and Mailing Lists

Aliases and forwarding can be used to map one recipient name to one or more other recipients. Figure 18-2 shows how `sendmail` handles aliases. Programs that read mail, such as `/usr/ucb/mail`, can have aliases of their own, expanded before sendmail gets invoked.

The local alias database exists in two forms. (See the section below for information about domain-wide aliases with the yellow pages.) One is a text form, maintained in the file `/etc/aliases,` which is a text file.

Figure 18-2 *How* `sendmail Uses Aliases`



The aliases are of the form

```
name: name1, name2, ...
```

You can alias only local names. Local names are those that resolve to a particular mailer called "local," which usually implies a name with the current host name, or no host name.

For example,

```
eric@mit-xx: eric@berkeley
```

produces an error message if eric@mit-xx is not a local name. Blank lines and lines beginning with a sharp sign (''#'') are comments.

The second form is processed by the *dbm*(3) library. You can run the /usr/ucb/newaliases command to rebuild the binary forwarding database from the mail aliases file /etc/aliases, and create the dbm database files /etc/aliases.dir and /etc/aliases.pag. This is the form that sendmail actually uses to resolve aliases. This is equivalent to giving sendmail the -bi flag:

```
% /usr/lib/sendmail -bi
```

If the **D** option is specified in the configuration, sendmail rebuilds the alias database automatically if possible when it is out of date. The conditions under which it will do this are:

1. The dbm version of the database is mode 666.
2. sendmail is running setuid to root.

Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

### Potential Problems

Several problems can occur with the alias database when a sendmail process accesses the dbm version while it is only partially built. This can happen under two circumstances. The first is when one process accesses the database while another process is rebuilding it, The second circumstance occurs when the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

sendmail has two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, at the end of the rebuild it adds an alias of the form

```
@: @
```

which is not normally legal. Before sendmail accesses the database, it checks to insure that this entry exists. You need to specify the a option in the configuration file for this to occur. It will wait for the time specified in the a option for this entry to appear, at which point it will force a rebuild itself.[6]

### Mailing Lists

Aliases that expand to more than one name are called mailing lists. In typical systems, most mail traffic is from lists, so take extra caution when administering them. If an error occurs when sending to a certain name, say *x*, sendmail will look for an alias of the form ''owner-*x*'' to receive the errors. This is typically useful for a mailing list where the submitter to the list has no control over the

[6] Note: the D option must be specified in the configuration file for this rebuild to occur.

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.

### Inclusion

Inclusion is specified with the syntax:

```
:Include: pathname
```

A name of this form reads the file specified by *pathname* and sends to all users listed in that file. The intent is *not* to directly use this feature, but rather to use this as an alias. For example, an alias of the form:

```
project: :include:/usr/project/userlist
```

is a method of letting a project maintain a mailing list without interaction with the system administration, even if the alias file is protected.

It is not necessary to rebuild the alias database when an :include: list is changed.

### Yellow Pages Aliases

Mail aliases can also be obtained from a domain-wide database using YP. The YP database is usually the /etc/aliases file from some central mail machine for the group or department. This database is searched after local aliases but before forward files. The yellow pages map name can be specified with the Y configuration option; the default is mail.aliases. This feature can be disabled by turning off YP, by not having the given map in the default domain, or by including a Y option with a null string as its argument.

### Update Policy

As system administrator, you need to decide on a policy for updating aliases. The proper way to change forwarding might be to mail a message to "aliases@mailhost" that tells them the machine on which you wish to get your mail. Machines should run YP to get automatic access to the latest alias database, instead of forcing users to manage their own forwarding, which can create problems. For example, in previous releases of sendmail, some users tried to forward their mail from user@hosta to user@hostb by making hosta an alias for hostb in the host file. This caused mail to go into a black hole—a problem which has been fixed in Release 4.0. Another common mistake is to put a .forward file in the home directory of hosta that says "user@hostb." The problem is that when the mail gets to hostb, it looks up "user" in the YP aliases and sends it back to user@hosta, resulting in a loop and more bounced mail.

Naming Conventions

It is a good idea to follow a standard convention for naming users. For example, give every user an alias consisting of their first initial followed by last name. These aliases exist on the machine that relays mail to the outside world, and are used as the domain-wide `mail.aliases` map. This way, on any machine that uses domain-wide YP aliases, you can just mail to the first initial and last name instead of having to remember the specific user and host name.

Potential Problems

Extra care must be taken to avoid loops and inconsistent databases when both local and domain-wide aliases are used. For example, consider a user named "smith" who moves from a machine called "a" to a machine named "b." You might be tempted to do this by adding a local alias on machine a that forwards smith@a to smith@b. But if smith maps to smith@a in the domain-wide database, a loop will result.

CAUTION

**`:include:` files and program aliases in domain aliases may not have the expected result, unless the appropriate files are available everywhere in the domain. The domain can include aliases to a central machine which can have the files.**

Forwarding

After aliasing, recipients that are local and valid are checked for the existence of a `.forward` file in their home directory. If it exists, the message is *not* sent to that user, but rather to the list of users in that file. Often this list will contain only one name, and the feature will be used for network mail forwarding.

Forwarding also permits you to specify a private program for incoming mail. For example, forwarding to:

```
" | /usr/local/newmail login_name"
```

will pipe the message into the given program instead of delivering to the default file `/var/spool/mail/login_name`. This feature is used by the `vacation` program.

## The Mail Queue

Under conditions of high load or temporary failures, `sendmail` places a message into a job queue instead of delivering it immediately. The mail queue should be processed automatically. However, sometimes you may have to intervene manually. For example, if a major host is down for a period of time the queue may become clogged. Although `sendmail` ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

Printing the Queue

The contents of the queue can be printed by specifying the `-bp` flag to `sendmail`:

```
% /usr/lib/sendmail -bp | more
```

This produces a listing of the queue IDs, the size of the message, the date the message entered the queue, and the sender and recipients.

## Format of Queue Files

All queue files located in `/var/spool/mqueue` have the form *xfAA99999* where *AA99999* is the *id* for this file and the *x* is a type. The types are:

`d` The data file. The message body (excluding the header) is kept in this file.

`l` The lock file. If this file exists, the job is currently being processed, and a queue run will not process the file. For that reason, an extraneous `lf` file can cause a job to apparently disappear.

`n` This file is created when an ID is created. It is a separate file that insures that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time.

`q` The queue control file. This file contains the information necessary to process the job.

`t` A temporary file. These are an image of the `qf` file when it is being rebuilt. It should be renamed to a `qf` file almost immediately.

`x` A transcript file, existing during the life of a session showing everything that happens during that session.

The `qf` file is structured as a series of lines each beginning with a code letter. The lines are as follows:

`P` The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.

`T` The job creation time. This is used to compute when to time out the job.

`D` The name of the data file. There may only be one of these lines.

`M` A message. This line is printed by using `sendmail` with the **–bp** flag, and is generally used to store status information. It can contain any text.

`S` The sender name. There may only be one of these lines.

`E` Error recipient name. Error messages will be sent to this user instead of the sender. It is optional.

`H` A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.

`R` A recipient name. There may any number of these lines. This will normally be completely aliased, but is actually realiased when the job is processed. There will be one line for each recipient. The recipient name must be at the end of the `qf` file.

As an example, the following is a queue file sent to ‘‘jg@granite’’.

```
P4579
T546644595
DdfAA00393
MDeferred: Host granite.dec.com is down
S<dshr@devnull>
Hreceived: from snail.sun.com by Sun.COM (4.0/SMI-3.2)
        id AA00393; Tue, 28 Apr 87 14:43:15 PDT
Hreceived: from devnull.sun.uucp by snail.sun.com (3.2/SMI-3.2)
        id AA12023; Tue, 28 Apr 87 14:42:54 PDT
Hreceived: by devnull.sun.uucp (3.2/SMI-3.2)
        id AA01475; Tue, 28 Apr 87 14:43:37 PDT
Hmessage-id: <8704282143.AA01475@devnull.sun.uucp>
Hdate: Tue 28 Apr 1987 14:42:49 EST
HFrom: David Rosenthal <dshr@devnull>
Hsubject: X11 symlink
HTo: jg@granite.DEC.COM (Jim Gettys)
Hin-reply-to: jg's message of Tue, 28 Apr 87 10:28:17 PDT
R<jg@granite.DEC.COM>
```

This shows the name of the data file, the person who sent the message, the submission time (in seconds since January 1, 1987), the message priority, headers for the message, and the recipients.

### Forcing the Queue

sendmail should run the queue automatically at intervals. The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, sendmail first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to insure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process. Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no way to resolve this without setting a time limit.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in sendmail spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory as follows:

```
    # cd /var/spool
    # mv mqueue omqueue; mkdir mqueue; chmod 777 mqueue
    # ps ax | grep sendmail
```

Kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon. To do this, do the following:

```
    # kill process_id
    # /usr/lib/sendmail -bd -q1h
```

To run the old mail queue, run the following command:

```
# /usr/lib/sendmail -oQ/var/spool/omqueue -q
```

The -oQ flag specifies an alternate queue directory and the -q flag says to just run every job in the queue. Use the -v flag if you want to see the verbose output displayed on the screen.

When the queue is finally emptied, you can remove the directory:

```
# rmdir /var/spool/omqueue
```

You can also run a subset of the queue at any time with the -R*string* (run queue where any recipient name matches *string*) or with -M*nnnnn* (run just one message, with queue id *nnnnn*). For example, you could give the command

```
% /usr/lib/sendmail -Rwnj
```

to run the subset of the queue that contains the recipient "wnj".

## 18.6. Arguments

The complete list of arguments to sendmail is described in detail in Section 18.9. Some important arguments are described here.

### Queue Interval

The amount of time between forking a process to run through the queue is defined by the -q flag. If you run in mode **i** or **b** (the default) this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in **q** mode it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue.

### Daemon Mode

If you allow incoming mail over an TCP connection, you should have a daemon running. This should be set by your /etc/rc.local file using the -bd flag. You can combine the -bd flag and the -q flag in one call:

```
# /usr/lib/sendmail -bd -q30m
```

### Debugging

There are a fairly large number of debug flags built into sendmail. Each debug flag has a number and a level, where higher levels indicate "print more information." The convention is that levels greater than nine are unnecessary unless you are debugging that particular piece of code. Debug flags are set using the -d option; the syntax is:

```
debug-flag:      –d debug-list
debug-list:      debug-option [ , debug-option ]
debug-option:    debug-range [ . debug-level ]
debug-range:     integer | integer — integer
debug-level:     integer
```

where spaces are for reading ease only. For example:

```
-d12        Set flag 12 to level 1
-d12.3      Set flag 12 to level 3
-d3-17      Set flags 3 through 17 to level 1
-d3-17.4        Set flags 3 through 17 to level 4
```

**Note** There are numerous debugging flags available for `sendmail`. If you have source code, you can refer to the list of debug flags in the code.

### Trying a Different Configuration File

You can specify an alternative configuration file by using the `-C` flag; for example,

```
% /usr/lib/sendmail -Ctest.cf
```

uses the configuration file `test.cf` instead of the default `/etc/sendmail.cf.` If the `-C` flag has no value it defaults to `sendmail.cf` in the current directory.

### Quick Configuration Startup

If you are using a `sendmail` command-line interface heavily for mail you can set up a fast version of the configuration file by using the `-bz` flag:

```
# /usr/lib/sendmail -bz
```

This creates the file `/etc/sendmail.fc` ("frozen configuration"). This file is an image of `sendmail`'s data space after reading in the configuration file. If this file exists, it is used instead of `/etc/sendmail.cf. sendmail.fc` must be rebuilt manually every time you change `sendmail.cf.`

The frozen configuration file will be ignored if a `-C` flag is specified. The frozen configuration file depends on the specific version of the `/usr/lib/sendmail` binary file. If the binary file does not match the frozen configuration file, then the frozen file will be removed. There is no checking to make sure that the frozen file has been updated after the text version of the file; it is important for you to remove the frozen file or run `/etc/sendmail -bz` after every change to the configuration file.

CAUTION **Note that once you create the frozen configuration file, the default configuration file is ignored. You need to remove `sendmail.fc` for the default configuration file to be read instead of the frozen configuration file.**

## 18.7. Tuning

There are several configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line `OT3d` sets option T to the value "3d" (three days).

### Time Values

All time intervals are given using a syntax of numbers and letters. For example, "10m" represents ten minutes, whereas "2h30m" represents two and a half hours. The full set of letters is:

| | |
|---|---|
| s | seconds |
| m | minutes |
| h | hours |
| d | days |
| w | weeks |

#### Queue Interval

The argument to the `-q` flag specifies how often `sendmail` runs the queue. This is typically set to between fifteen minutes (-q15m) and one hour (-q1h).

#### Read Timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. Technically, this is not acceptable within the published protocols. However, it might be appropriate to set it to something large in certain environments (such as an hour). This will reduce the chance of several idle daemons piling up on your system. This timeout is set using the **r** option in the configuration file.

#### Message Timeouts

After sitting in the queue for a few days, a message should be returned. This is to insure that at least the sender is aware of the inability to send a message. The timeout is typically set to three days. This timeout is set using the T option in the configuration file.

You can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example,

```
% /usr/lib/sendmail -oT1d -q
```

will run the queue and flush anything that is one day old or older.

### Delivery Mode

There are several delivery modes that `sendmail` can operate in, set by the **d** configuration option. These modes specify how quickly mail will be delivered. Legal modes are:

```
i   deliver interactively (synchronously)
b   deliver in background (asynchronously)
q   queue only (don't deliver)
```

There are tradeoffs. Mode `i` passes the maximum amount of information to the sender, but is hardly ever necessary. Mode `q` puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. Mode `b` is probably a good compromise.

### Load Limiting

Often central mail machines can be overloaded. Of course the best solution is to dedicate a more powerful machine to handling mail, but load almost always expands to consume whatever resources are allocated.

The `x` and `X` options can be used to limit the load caused by `sendmail`. Both of these configuration options take an argument that is an integer load average. For example, if you specify **x4** and **X8** then the **x** load limiting will be used when the load is above four, and the **X** load limiting will be used when the load is above eight. When the load is above the value specified in the **X** option, the SMTP server will not accept connections from the network (locally originated mail and other mail such as UUCP are not affected). The **x** option has a more subtle effect, controlling whether messages are queued for later delivery or delivered immediately. The general idea is to always deliver 'small' messages immediately, and defer 'large' messages for delivery during off-peak periods.

The **q** option specifies the maximum size of message that will be delivered immediately. The "size" of the message includes not only the number of bytes in the message, but also penalties for large number of recipients, and for delivery attempts that were unsuccessful. The penalty per recipient is option value `y`, by default set to 1000. The penalty per delivery attempt is the option value `z`, by default set to 9000. The size limit is also dependent on current load, so that more and more messages are queued as load goes higher. If the load is one above the **x** threshold, then the limit is halved; if the load is two above the threshold, the limit is divided by three, etc. Note that this limit also applies to messages being delivered when running the queue, in contrast to earlier versions of `sendmail`.

The goal of load limiting is to prevent wasting time during loaded periods by attempting to deliver large messages, messages to many recipients, and messages to sites that have been down for a long time.

### Log Level

The level of logging can be set for `sendmail`. The default using a standard configuration table is level 9. The levels are as follows:

0 No logging.

1 Major problems only.

2 Message collections and failed deliveries.

3 Successful deliveries.

4 Messages being deferred (due to a host being down, etc.).

5 Normal message queueups.

6 Unusual but benign incidents, for example, trying to process a locked queue file.

9 Log internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts.

12 Several messages that are basically only of interest when debugging.

16 Verbose information regarding the queue.

Refer to the `syslog` section in Chapter 8, "Periodic Maintenance," for more information.

### File Modes

Certain files may have a number of modes. The modes depend on what functionality you want and the level of security you require.

#### To Suid or not to Suid?

By default, `sendmail` is setuid to root. At the point where it is about to `exec`(2) a mailer, it checks to see if the user ID is zero; if so, it resets the user ID and group ID to a default (set by the **u** and **g** options). (You can override this by setting the S flag to the mailer for mailers that are trusted and must be called as root.) However, this will cause mail processing to be accounted (using *sa*(8)) to root rather than to the user sending the mail.

#### Temporary File Modes

The mode of all temporary files that `sendmail` creates is determined by the **F** option. Reasonable values for this option are 0600 and 0644. If the more permissive mode is selected, it will not be necessary to run `sendmail` as root at all (even when running the queue), but will allow users to read mail in the queue.

#### Should my Alias Database be Writable?

One approach is to provide the alias database (`/etc/aliases*`) with mode 666. There are some dangers inherent in this approach: any user can add himself or herself to any list, or can cause any user's mail to be forwarded to another location.

## 18.8. The Configuration File

This section describes the configuration file in detail, including hints for writing your own, if required.

The syntax of the configuration file is designed to be reasonably easy to parse, since parsing can be done every time `sendmail` starts up. Unfortunately, this can sacrifice readability.

`sendmail` uses single letters for several different functions:

- Command-line flags
- Configuration options
- Queue file line types
- Configuration file line types
- Mailer field names
- Mailer flags
- Macro names
- Class names

This section provides an overview of the configuration file is provided, plus details of its semantics.

### Building a Configuration File from Scratch

Building a configuration file from scratch is a complex task. Fortunately, it is almost never necessary to do so; nearly every situation that may come up may be resolved by changing an existing file. In any case, it is critical that you understand what it is that you are trying to do and come up with a policy statement for the delivery of mail. This section is intended to explain what the real purpose of a configuration file is and to give you some ideas for what your policy might be.

## Purpose of the Configuration File

The configuration file has three major purposes. The first and simplest is to set up the environment for `sendmail`. This invólves setting the options and defining a few critical macros.

The second purpose is to map names into the actual set of commands necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

The third purpose is to rewrite names in the message. This should typically be done in two phases. The first phase maps names in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving mailer. `sendmail` does this in three subphases. Rulesets one and two are applied to all sender and recipient names respectively. After this, you may specify per-mailer rulesets for both sender and recipient names; this allows mailer-specific customization. Finally, ruleset four is applied to do any conversion to external form.

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 cannot be changed without changing the code. In particular, the following characters have special interpretations:

```
< > ( ) " \
```

Any attempt to use these characters for other than their RFC822 purpose in names is probably doomed to disaster.

## Domains and Policies

RFC819 describes domain-based naming. This is touched on in RFC822 as well. Essentially each host is given a name that is a right-to-left dot qualified pseudo-path from a distinguished root. The elements of the path are organizational entities, not physical networks.

## How to Proceed

Once you have decided a policy, it is worth examining the available configuration files to decide if any of them are close enough to steal major parts of. Even under the worst of conditions, there is a fair amount of boilerplate that can be collected safely.

The next step is to build ruleset three. This will be the hardest part of the job. Beware of doing too much to the name in this ruleset, since anything you do will reflect through to the message. In particular, stripping of local domains is best deferred, since this can leave you with names with no domain spec at all. Since `sendmail` likes to append the sending domain to names with no domain, this can change the semantics of names. Also try to avoid fully qualifying domains in this ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long names reflected into messages. The SunOS configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get all names into a cleaner form.

Once you have ruleset three finished, the other rulesets should be relatively simple. If you need hints, examine the supplied configuration files.

### Testing the Rewriting Rules — the –bt Flag

When you build a configuration file, you can do a certain amount of testing using the "test mode" of `sendmail`. For example, you could invoke `sendmail` as:

```
% sendmail -bt -Ctest.cf
```

which would read the configuration file `test.cf` and enter test mode. For example:

```
ADDRESS TEST MODE
Enter <ruleset> <name>
>
```

In this mode, you enter lines of the form:

```
rwset name
```

where *rwset* is the rewriting set you want to use and *name* is a name to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the name it ends up with. You may use a comma separated list of *rwsets* for sequential application of rules to an input; ruleset three is always applied first. For example:

```
> 1,21,4 monet:bollard
```

first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the `-d21` flag to turn on more debugging. For example,

```
% sendmail -bt -d21.99
```

turns on an incredible amount of information; a single word name may result in several pages worth of information.

## The Syntax

```
############################################################
#
#   Sendmail configuration file for "MAIN MACHINES"
#
#   You should install this file as /etc/sendmail.cf
#   if your machine is the main (or only) mail-relaying
#   machine in your domain.  Then edit the file to
#   customize it for your network configuration.
#
#   See the manual "System and Network Administration for the Sun
#   Workstation". Look at "Setting Up The Mail Routing System" in
#   the chapter on Communications.  The Sendmail reference in the
#   back of the manual is also useful.
#
#   @(#)main.mc 1.14 88/01/22 SMI
#
```

```
### local info

# my official hostname
# You have two choices here.  If you want the gateway machine to identify
# itself as the DOMAIN, use this line:
Dj$m
# If you want the gateway machine to appear to be INSIDE the domain, use:
#Dj$w.$m
# Unless you are using sendmail.mx (or have a fully-qualified hostname), use:
#Dj$w

# major relay mailer - typical choice is "ddn" if you are on the
# Defense Data Network (e.g. Arpanet or Milnet)
DMsmartuucp

# major relay host: use the $M mailer to send mail to other domains
DR ddn-gateway
CR ddn-gateway

# If you want to pre-load the "mailhosts" then use a line like
# FS /usr/lib/mailhosts
# and then change all the occurences of $%y to be $=S instead.
# Otherwise, the default is to use the hosts.byname map if YP
# is running (or else the /etc/hosts file if no YP).

# valid top-level domains (default passes ALL unknown domains up)
CT arpa com edu gov mil net org
CT us de fr jp kr nz il uk no au fi nl se ca ch my dk ar

# options that you probably want on a mailhost:

# checkpoint the queue after this many receipients
OC10

# refuse to send tiny messages to more than these recipients
Ob10

#################################################
#
#   General configuration information

# local domain names
#
# These can now be set from the domainname system call.
# If your YP domain is different from the domain name you would like to have
# appear in your mail headers, edit them to be your mail domain name.
# Note that the first component of the YP domain name is stripped off unless
# it begins with a dot or a plus sign.
# DmPodunk.EDU

# known hosts in this domain are obtained from gethostbyname() call

# Version number of configuration file
```

```
DVSMI-4.0


###   Standard macros

# name used for error messages
DnMailer-Daemon
# UNIX header format
DlFrom $g  $d
# delimiter (operator) characters
Do.:%@!^=/[]
# format of a total name
Dq$g$?x ($x)$.
# SMTP login message
De$j Sendmail $v/$V ready at $b

###   Options

# Remote mode - send through server if mailbox directory is mounted
OR
# location of alias file
OA/etc/aliases
# default delivery mode (deliver in background)
Odbackground
# rebuild the alias file automagically
OD
# temporary file mode -- 0600 for secure mail, 0644 for permissive
OF0600
# default GID
Og1
# location of help file
OH/usr/lib/sendmail.hf
# log level
OL9
# default messages to old style
Oo
# Cc my postmaster on error replies I generate
OPPostmaster
# queue directory
OQ/usr/spool/mqueue
# read timeout for SMTP protocols
Or15m
# status file -- none
OS/etc/sendmail.st
# queue up everything before starting transmission, for safety
Os
# return queued mail after this long
OT3d
# default UID
Ou1

###   Message precedences
Pfirst-class=0
```

```
Pspecial-delivery=100
Pjunk=-100

###   Trusted users
T root daemon uucp

###   Format of headers
H?P?Return-Path: <$g>
HReceived: $?sfrom $s $.by $j ($v/$V)
    id $i; $b
H?D?Resent-Date: $a
H?D?Date: $a
H?F?Resent-From: $q
H?F?From: $q
H?x?Full-Name: $x
HSubject:
H?M?Resent-Message-Id: <$t.$i@$j>
H?M?Message-Id: <$t.$i@$j>
HErrors-To:

###########################
###   Rewriting rules   ###
###########################


#  Sender Field Pre-rewriting
S1
# None needed.

#  Recipient Field Pre-rewriting
S2
# None needed.

# Name Canonicalization

# Internal format of names within the rewriting rules is:
#   anything<@host.domain.domain...>anything
# We try to get every kind of name into this format, except for local
# names, which have no host part.  The reason for the "<>" stuff is
# that the relevant host name could be on the front of the name (for
# source routing), or on the back (normal form).  We enclose the one that
# we want to route on in the <>'s to make it easy to find.
#
S3

# handle "from:<>" special case
R<>         $@@             turn into magic token

# basic textual canonicalization
R$*<$+>$*       $2              basic RFC822 parsing

# make sure <@a,@b,@c:user@d> syntax is easy to parse -- undone later
R@$+,$+:$+      @$1:$2:$3           change all "," to ":"
```

```
R@$+:$+         $@$>6<@$1>:$2         src route canonical

R$+:$*;@$+      $@$1:$2;@$3       list syntax
R$+@$+          $:$1<@$2>         focus on domain
R$+<$+@$+>      $1$2<@$3>         move gaze right
R$+<@$+>        $@$>6$1<@$2>          already canonical

# convert old-style names to domain-based names
# All old-style names parse from left to right, without precedence.
R$-!$+          $@$>6$2<@$1.uucp>     uucphost!user
R$-.$+!$+       $@$>6$3<@$1.$2>       host.domain!user
R$+%$+          $@$>3$1@$2        user%host

#  Final Output Post-rewriting
S4
R$+<@$+.uucp>       $2!$1             u@h.uucp => h!u
R$+         $: $>9 $1         Clean up addr
R$*<$+>$*       $1$2$3            defocus


#  Clean up an name for passing to a mailer
#  (but leave it focused)
S9
R@          $@$n              handle <> error addr
R$*<$*LOCAL>$*      $1<$2$m>$3            change local info
R<@$+>$*:$+:$+      <@$1>$2,$3:$4             <route-addr> canonical


#######################
#   Rewriting rules

# special local conversions
S6
R$*<@$*$=m>$*       $1<@$2LOCAL>$4            convert local domain

# Local and Program Mailer specification

Mlocal, P=/bin/mail, F=rlsDFMmnP, S=10, R=20, A=mail -d $u
Mprog,  P=/bin/sh,   F=lsDFMeuP,  S=10, R=20, A=sh -c $u

S10
# None needed.

S20
# None needed.

###########################################################
#####
#####       Ethernet Mailer specification
#####
#####   Messages processed by this configuration are assumed to remain
#####   in the same domain.  This really has nothing particular to do
#####   with Ethernet - the name is historical.
```

```
Mether, P=[TCP], F=msDFMuCX, S=11, R=21, A=TCP $h
S11
R$*<@$+>$*      $@$1<@$2>$3         already ok
R$+         $@$1<@$w>           tack on our hostname

S21
# None needed.



##########################################################
#  General code to convert back to old style UUCP names
S5
R$+<@LOCAL>     $@ $D!$1            name@LOCAL => sun!name
R$+<@$-.LOCAL>      $@ $2!$1            u@h.LOCAL => h!u
R$+<@$+.uucp>       $@ $2!$1            u@h.uucp => h!u
R$+<@$*>        $@ $2!$1            u@h => h!u
# Route-addrs do not work here.  Punt til uucp-mail comes up with something.
R<@$+>$*        $@ @$1$2            just defocus and punt
R$*<$*>$*       $@ $1$2$3           Defocus strange stuff

#   UUCP Mailer specification

Muucp,  P=/usr/bin/uux, F=msDFMhuU, S=13, R=23,
    A=uux - -r $h!rmail ($u)

# Convert uucp sender (From) field
S13
R$+         $:$>5$1             convert to old style
R$=w!$+         $2              strip local name
R$+         $:$w!$1             stick on real host name

# Convert uucp recipient (To, Cc) fields
S23
R$+         $:$>5$1             convert to old style


############################################################
#
#       DDN Mailer specification
#
#   Send mail on the Defense Data Network
#      (such as Arpanet or Milnet)

Mddn,   P=[TCP], F=msDFMuCX, S=22, R=22, A=TCP $h, E=

# map containing the inverse of mail.aliases
DZmail.byaddr

S22
R$*<@LOCAL>$*       $:$1
R$-<@$->        $:$>3${Z$1@$2$}         invert aliases
R$*<@$+.$*>$*       $@$1<@$2.$3>$4          already ok
```

```
R$+<@$+>$*        $@$1<@$2.$m>$3          tack on our domain
R$+          $@$1<@$m>            tack on our domain


# "Smart" UUCP mailer: Uses UUCP transport but domain-style naming
Msmartuucp, P=/usr/bin/uux, F=CmsDFMhuU, S=22, R=22,
    A=uux - -r $h!rmail ($u)


##############################################################
#
#       RULESET ZERO
#
#   This is the ruleset that determines which mailer a name goes to.

# Ruleset 30 just calls rulesets 3 then 0.
S30
R$*          $: $>3 $1            First canonicalize
R$*          $@ $>0 $1            Then rerun ruleset 0

S0
# On entry, the address has been canonicalized and focused by ruleset 3.
# Handle special cases.....
R@           $#local $:$n             handle <> form
# For numeric spec, you can't pass spec on to receiver, since rcvr's
# are not smart enough to know that [x.y.z.a] is their own name.
R<@[$+]>:$*      $:$>9 <@[$1]>:$2       Clean it up, then...
R<@[$+]>:$*      $#ether $@[$1] $:$2    numeric internet spec
R<@[$+]>,$*      $#ether $@[$1] $:$2    numeric internet spec
R$*<@[$+]>       $#ether $@[$2] $:$1    numeric internet spec

# resolve the local hostname to "LOCAL".
R$*<$*$=w.LOCAL>$*  $1<$2LOCAL>$4           thishost.LOCAL
R$*<$*$=w.uucp>$*   $1<$2LOCAL>$4           thishost.uucp
R$*<$*$=w>$*        $1<$2LOCAL>$4           thishost

# Mail addressed explicitly to the domain gateway (us)
R$*<@LOCAL>      $@$>30$1             strip our name, retry
R<@LOCAL>:$+        $@$>30$1             retry after route strip

# deliver to known ethernet hosts explicitly specified in our domain
R$*<@$%y.LOCAL>$*   $#ether $@$2 $:$1<@$2>$3    user@host.sun.com

# etherhost.uucp is treated as etherhost.$m for now.
# This allows them to be addressed from uucp as foo!sun!etherhost!user.
R$*<@$%y.uucp>$*    $#ether $@$2 $:$1<@$2>$3    user@etherhost.uucp

# Explicitly specified names in our domain -- that we've never heard of
R$*<@$*.LOCAL>$*    $#error $:Never heard of host $2 in domain $m

# Clean up addresses for external use -- kills LOCAL, route-addr ,=>:
R$*          $:$>9 $1             Then continue...
```

```
# resolve UUCP-style names
R<@$-.uucp>:$+      $#uucp  $@$1 $:$2       @host.uucp:...
R$+<@$-.uucp>       $#uucp  $@$2 $:$1       user@host.uucp

# Pass other valid names up the ladder to our forwarder
#R$*<@$*.$=T>$*     $#$M    $@$R $:$1<@$2.$3>$4 user@domain.known

# Replace following with above to only forward "known" top-level domains
R$*<@$*.$+>$*       $#$M    $@$R $:$1<@$2.$3>$4 user@any.domain

# if you are on the DDN, then comment-out both of the the lines above
# and use the following instead:
#R$*<@$*.$+>$*      $#ddn $@ $2.$3 $:$1<@$2.$3>$4   user@any.domain

# All addresses in the rules ABOVE are absolute (fully qualified domains).
# Addresses BELOW can be partially qualified.

# deliver to known ethernet hosts
R$*<@$%y>$*     $#ether $@$2 $:$1<@$2>$3    user@etherhost

# other non-local names have nowhere to go; return them to sender.
R$*<@$+.$->$*       $#error $:Unknown domain $3
R$*<@$+>$*      $#error $:Never heard of $2 in domain $m
R$*@$*          $#error $:I don't understand $1@$2

# Local names with % are really not local!
R$+%$+          $@$>30$1@$2         turn % => @, retry

# everything else is a local name
R$+         $#local $:$1            local names
```

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a pound sign (#) are comments.

### D — Define Macro

Macros are named with a single character. You can select these from the entire ASCII set, but you should select user-defined macros from the set of uppercase letters only. Lowercase letters and special symbols are used internally.

The syntax for macro definitions is:

**D***xval*

where *x* is the name of the macro and *val* is the value it should have. Macros can be inserted in most places using the escape sequence $*x*.

Following is an example of a macro definition from the configuration file:

```
DJ$w,$M
```

### C and F — Define Classes

You can define classes of words to match on the left hand side of rewriting rules. For example, you might create a class of all local names for this site so that you can eliminate attempts to send to yourself.

These can either be defined directly in the configuration file or read in from another file or from another command. You can give classes names from the set of uppercase letters. Lowercase letters and special characters are reserved for system use.

The syntax is:

```
Cc word1 word2
Fc file
Fc | command
```

The first form defines the class *c* to match any of the named words. The second form reads words from the file into the class *c*. The format, if given, is used with `scanf` to read from the file; otherwise, the first word from each line is used. The third form executes the given command and reads the elements of the class from standard output of the command. You could split all these class lines among multiple lines; for example, the two forms:

```
CHmonet ucbmonet
```

and

```
CHmonet
CHucbmonet
```

are equivalent.

Macro and class names can be any letter, with lowercase reserved for internal use, uppercase for users.

### O — Set Option

There are several options (not to be confused with mailer flags or command line arguments) that can be set from a configuration file. Options are also represented by single characters. The syntax of this line is:

**O***ovalue*

This sets option *o* to *value*. Depending on the option, *value* may be a string, an integer, a boolean (with legal values "t," "T," "f," or "F" — the default is TRUE), or a time interval. See Section 18.10 for the list of options.

### P — Precedence Definitions

You can define values for the "Precedence:" field using the **P** control line. The syntax of this field is:

**P***name=num*

when the *name* is found in a "Precedence:" field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than zero have the special property that error messages will not be returned. The default precedence is zero. For example, our list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

### T — Define Trusted Users

Trusted users are those users who are permitted to override the sender name using the `-f` flag. These typically are ''root,'' ''uucp,'' and ''network,'' but for some users it may be convenient to extend this list to include other users, perhaps to support a separate UUCP login for each host. The syntax of this line is:

**T** *user1 user2 . . .*

There may be more than one of these lines.

### H — Define Header

The format of the header lines is defined by the **H** line. The syntax of this line is:

**H** [?*mflags*?] *hname*:*htemplate*

Continuation lines in this specification are inserted directly into the outgoing message. The *htemplate* is macro expanded before it is inserted into the message. If the expansion is empty, the header line is not included. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input it is directed to the output regardless of these flags.

## Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into `sendmail` that cannot be changed without changing the code. These builtins are described here.

### Return-Receipt-To:

If this header is sent, a message will be sent to any specified names when the final delivery is complete. The mailer must have the **l** flag (local delivery) set in the mailer descriptor

### Errors-To:

If errors occur anywhere during processing, this header will cause error messages to go to the listed names rather than to the sender. This is intended for mailing lists.

### To:

If a message comes in with no recipients listed in the message (in a To:, Cc:, or Bcc: line) then `sendmail` will add an ''Apparently To:'' header line for each recipient specified on the `sendmail` command line.

### R and S — Rewriting Rules

Address parsing is done according to the rewriting rules. These are a simple pattern-matching system. `sendmail` scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the name is replaced by the right hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and may be referenced by the mailer definitions or by other rewriting sets.

The syntax of these two lines are:

**S***n*

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*

Following is an example of how a ruleset definition might look:

```
# handle "from:<>" special case
R<>          $@@               turn into magic token
```

The fields must be separated by at least one tab character; there may be embedded spaces in the fields. The *lhs* is a pattern that is applied to the input. If it matches, the input is rewritten to the *rhs*. The *comments* are ignored.

### M — Define Mailer

Programs and interfaces to mailers are defined on this line. The format is:

**M***name*, {*field*=*value*} *

where *name* is the name of the mailer (used in error messages) and the ''field=value'' pairs define attributes of the mailer. Fields are:

| | |
|---|---|
| `Path` | The pathname of the mailer |
| `Flags` | Special flags for this mailer |
| `Sender` | A rewriting set for sender names |
| `Recipient` | A rewriting ruleset for recipient names |
| `Argv` | An argument vector to pass to this mailer |
| `Eol` | The end-of-line string for this mailer |
| `Maxsize` | The maximum message length to this mailer |
| `Length` | The maximum length of the `argv` for this mailer |

Only the first character of the field name is checked.

## The Semantics

This section describes the details of rewriting rules and mailer descriptions.

### Special Macros, Conditionals

Macros are referenced using the construct $*x*, where *x* is the name of the macro to be matched (LHS) or inserted (RHS). Lower case letters are reserved to have special semantics, and some special characters are reserved to provide conditionals.

The following macros *must* be defined to transmit information into `sendmail`:

| | |
|---|---|
| **e** | The SMTP entry message |
| **j** | The official domain name for this site |
| **l** | The format of the UNIX "From" line |
| **n** | The name of the daemon (for error messages) |
| **o** | The set of "separators" in names |
| **q** | default format of sender names |

The **$e** macro is printed out when SMTP starts up. The first word of **$e** should be the **$j** macro. The **$j** macro should be in domain name format. The **$o** macro consists of a list of characters which will be considered tokens and which will separate tokens when scanning. For example, if “y” were in the **$o** macro, then the input “xyzzy” would be scanned as four tokens: “x,” “y,” and “zz” and ”y”. Finally, the **$q** macro specifies how a sender name should appear in a message when it is created. For example, on Sun's mail routing system these definitions are:

```
De$j Sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g  $d
Do.:%@!^=/
Dq$g$?x ($x)$.
Dj$H.$D
```

All of these macros need not be changed except under unusual circumstances.

An acceptable alternative for the **$q** macro is “$?x$x $.<$g>”. These correspond to the following two formats:

```
nowicki@sun.COM (Bill Nowicki)
Bill Nowicki <nowicki@sun.COM>
```

Some macros are defined by `sendmail` for use in mailer arguments or for other contexts. These macros are:

| | |
|---|---|
| **a** | The origination date in ARPANET format |
| **b** | The current date in ARPANET format |
| **c** | The hop count |
| **d** | The date in UNIX (ctime) format |
| **f** | The sender (from) name |
| **g** | The sender name relative to the recipient |
| **h** | The recipient host |
| **i** | The queue ID |
| **m** | The domain name |
| **p** | Sendmail's process ID |
| **r** | Protocol used |
| **s** | Sender's host name |
| **t** | A numeric representation of the current time |
| **u** | The recipient user |
| **v** | The version number of `sendmail` |
| **w** | The hostname of this site |
| **x** | The full name of the sender |
| **z** | The home directory of the recipient |

There are three types of dates that can be used. The **$a** and **$b** macros are in ARPANET format; **$a** is the time as extracted from the “Date:” line of the message (if there was one), and **$b** is the current date and time (used for postmarks). If no “Date:” line is found in the incoming message, **$a** is set to the current time also. The **$d** macro is equivalent to the **$a** macro in UNIX (ctime) format.

The **$f** macro is the ID of the sender as originally determined; when mailing to a specific host the **$g** macro is set to the name of the sender *relative to the recipient.* For example, if you send to “bollard@matisse” from the machine “ucbarpa” the **$f** macro will be “eric” and the **$g** macro will be "eric@ucbarpa."

The **$x** macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to `sendmail`. The second choice is the value of the "Full-name:" line in the header if it exists, and the third choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the `/etc/passwd` file.

When sending, the **$h**, **$u**, and **$z** macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the **$@** and **$:** part of the rewriting rules, respectively.

The **$p** and **$t** macros are used to create unique strings (for example, for the "Message-Id:" field). The **$i** macro is set to the queue ID on this host; if put into the timestamp line it can be useful for tracking messages. The **$v** macro is set to be the version number of `sendmail`; this is normally put in timestamps and has been proven extremely useful for debugging. The **$w** macro is set to the primary name of this host as given by `gethostname(1)` and `gethostbyname(3)`. The **$c** field is set to the "hop count," that is, the number of times this message has been processed. This can be determined by the `-h` flag on the command line or by counting the timestamps in the message.

The **$r** and **$s** fields are set to the protocol used to communicate with `sendmail` and the sending hostname;

You can specify conditionals by using the syntax:

```
$?x text1 $| text2 $.
```

This inserts *text1* if the macro **$x** is set, and *text2* otherwise. The "else" ($|) clause may be omitted.

### Special Classes

The class **$=w** is the set of all names this host is known by. This can be used to delete local hostnames. The class **$=m** is set to the domain name.

### The Left Hand Side

The left hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Dollars signs introduce "metasymbols", which match things other than simple words, such as macros or classes.

The metasymbols are:

| | |
|---|---|
| `$*` | Match zero or more tokens |
| $+ | Match one or more tokens |
| $– | Match exactly one token |
| $=*x* | Match any string in class *x* |
| $~*x* | Match any token not in class *x* |
| $%*x* | Match any token in yp map *$x* |
| $!*x* | Match any token not in yp map *$x* |
| $x | Match macro *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

```
$-:$+
```

is applied to the input

```
UCBARPA:eric
```

the rule will match, and the values passed to the RHS will be:

```
$1  UCBARPA
$2  eric
```

The $%*x* uses the macro *x* to specify the name of a Yellow Pages map. The special form $%y matches any hostname in the ‘ `hosts.byname` map, or in `/etc/hosts` if not running YP.

## The Right Hand Side

When the left hand side of a rewriting rule matches, the input is replaced by the right hand side. Tokens are copied directly from the right-hand side unless they begin with a dollar sign. Metasymbols for more complicated substitutions are:

| | |
|---|---|
| $x | Expand macro *x* |
| $*n* | Substitute indefinite token *n* from LHS |
| $>*n* | Call ruleset *n* |
| $#*mailer* | Resolve to *mailer* |
| $@*host* | Specify *host* (+ prefix? ruleset return) |
| $:*user* | Specify *user* (+ prefix rule limit) |
| $[*host$*] | Map to primary hostname |
| ${*x name$*} | Map name through yp map $x. |

The $*n* (*n* being a digit) syntax substitutes the corresponding value from a $+, $–, $*, $=, or $~ match on the LHS. It may be used anywhere.

The $>*n* syntax causes the remainder of the line to be substituted as usual and then passed to ruleset *n*. The final value of ruleset *n* then becomes the substitution for this rule. (This can be compared to a procedure or function call.)

The $# syntax should *only* be used in ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to `sendmail` that the name has completely resolved. The complete syntax is:

$#*mailer*$@*host*$:*user*

This specifies the {mailer, host, user} triple necessary to direct the mailer. More processing may then take place depending on the mailer. For example, local names are aliased.

A right-hand side may also be preceded by a $@ or a $: to control evaluation. A $@ prefix causes the ruleset to return with the remainder of the right-hand side as the value. A $: prefix causes the rule to terminate immediately, but the ruleset to continue; thus this can be used to limit a rule to one application. Neither prefix affects the result of the right-hand side expansion.

The $@ and $: prefixes can precede a $> spec; for example:

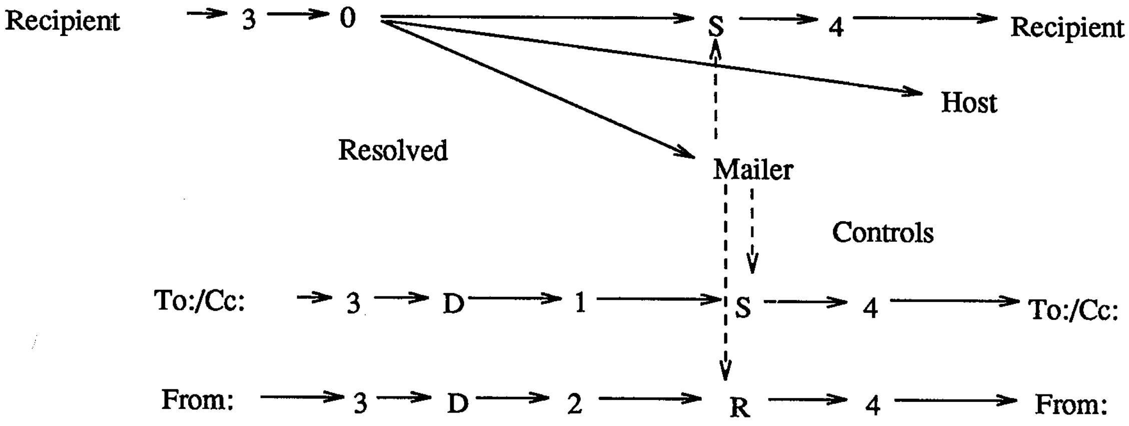
```
R$+     $:$>7$1
```

matches anything, passes that to ruleset seven, and continues; the $: is necessary to avoid an infinite loop. The $[host $] syntax replaces the host name with the "official" or primary host name, the one listed first in the *hosts.byname* yp map, or /etc/hosts if not running yp. This is used to eliminate "nicknames" for hosts. The ${x name $} syntax replaces the string by the result of the yp map indicated in macro $x.

## Semantics of Rewriting Rule Sets

There are five rewriting sets that have specific semantics. These are related as depicted by Figure 18-3.

Figure 18-3 *Rewriting Set Semantics*



D - sender domain addition

S - mailer-specific sender rewriting

R - mailer-specific recipient rewriting

Ruleset three should turn the name into ‘‘canonical form.’’ This form should have the basic syntax:

```
local-part@host-domain-spec
```

If no @ sign is specified, then the host-domain-spec *may* be appended from the sender name (if the **C** flag is set in the mailer definition corresponding to the *sending* mailer). Ruleset three is applied by sendmail before doing anything with any name.

Ruleset zero is applied after ruleset three to names that are going to actually specify recipients. It must resolve to a *{mailer, host, user}* triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is defined into the **$h** macro for use in the argument expansion of the specified mailer; the *user* is defined into **$u.**

Rulesets one and two are applied to all **from:**, **to:**, and **cc:** recipient names respectively. Then the rulesets specified in the mailer definition line (S= and R=) are applied. Note that this will be done many times for one message, depending on how many mailers the message is routed to by ruleset zero.

Ruleset four is applied last to all names in the message. It is typically used to translate internal to external form.

### The "error" Mailer

The mailer with the special name "error" can be used to generate a user error. The user field is a message to be printed. For example, the entry:

```
$#error$:Host unknown in this domain
```

on the RHS of a rule will cause the specified error to be generated if the LHS matches. This mailer is only functional in ruleset zero.

### Semantics of Mailer Descriptions

Each mailer has an internal name. This can be arbitrary, except that the names "local" and "prog" must be defined first and second, respectively. Ruleset zero will resolve names to this mailer name (and a host and user name).

The pathname of the mailer must be given in the P field. If this mailer should be accessed via a TCP connection, use the string "[TCP]" instead.

The F field defines the mailer flags. You should specify an "f" or "r" flag to pass the name of the sender as a `-f` or `-r` flag respectively. These flags are only passed if they were passed to `sendmail,` so that mailers that give errors under some circumstances can be placated. If the mailer is not picky you can just specify **-f$g** in the `argv` template. If the mailer must be called as `root` the "S" flag should be given; this will not reset the userid before calling the mailer.[7] If this mailer is local (that is, will perform final delivery rather than another network hop) the "l" flag should be given. Quote characters (backslashes and " marks) can be stripped from names if the "s" flag is specified; if this is not given they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction the "m" flag should be stated. If this flag is on, then the `argv` template containing **$u** will be repeated for each unique user on a given host. The "e" flag will mark the mailer as being 'expensive,' which will cause `sendmail` to defer connection until a queue run.[8]

An unusual case is the "C" flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if set, the domain specification of the sender (that is, the "@host.domain" part) is saved and is appended to any names in the message that do not already contain a domain specification. For example, a message of the form:

```
From: eric@ucbarpa
To: wnj@monet, mckusick
```

will be modified to:

```
From: eric@ucbarpa
To: wnj@monet, mckusick@ucbarpa
```

*if and only if* the "C" flag is defined in the mailer corresponding to

[7] `sendmail` must be running setuid to root for this to work.

[8] The c configuration option must be given for this to be effective.

"eric@ucbarpa."

The S and R fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient names respectively. These are applied after the sending domain is appended and the general rewriting sets (number one or two) are applied, but before the output rewrite (ruleset four) is applied. A typical use is to append the current domain to names that do not already have a domain. For example, a header of the form:

```
From: eric@host
```

might be changed to be:

```
From: eric@host.Podunk.EDU
```

or

```
From: ucbvax!eric
```

depending on the domain it is being shipped into. These sets can also be used to do special purpose output rewriting in cooperation with ruleset four.

The E field defines the string to use as an end-of-line indication. A string containing return and newline is the default, if using TCP, otherwise just a newline indicates end-of-line. You can use the usual backslash escapes (\r, \n, \f, \b).

An argument vector template is given as the A field. It may have embedded spaces. The template is macro-expanded before being passed to the mailer. Useful macros include **$h,** the host name resolved by ruleset zero, and **$u,** the user name (or names) resolved. If there is no argument with a **$u** macro in it, `sendmail` will use SMTP to communicate with the mailer. If the pathname for this mailer is "[TCP]," the argument vector should be

```
TCP $h [ port ]
```

where *port* is the optional port number to connect to.

If an L field exists, it specifies the maximum length of the **$u** macro passed to the mailer. This can be used with the **m** flag to send multiple recipients with one call to the mailer, while avoiding mailer limitations on argument length. This makes UUCP mail more efficient. **$u** will always expand to at least one recipient even if that recipient exceeds the L= limit.

For example, the specification:

```
Mlocal, P=/bin/mail, F=rlsmi,       S=10, R=20,A=mail -d $u
Mether, P=[TCP],     F=meC,         S=11, R=21,A=TCP $h, M=1000
```

specifies a mailer to do local delivery and a mailer for Ethernet delivery. The first is called "local," is located in the file `/bin/mail`, takes a `-r` flag, does local delivery, quotes should be stripped from names, and multiple users can be delivered at once; ruleset ten should be applied to sender names in the message and ruleset twenty should be applied to recipient names; the argument vector to send to a message will be the word "mail," the word "–d," and words containing the name of the receiving user. If a `-r` flag is inserted it will be between the words "mail" and "–d." The second mailer is called "ether," it should be connected to via TCP, it can handle multiple users at once, connections should be deferred, and any domain from the sender name should be appended to any receiver name without a domain; sender names should be processed by ruleset eleven and recipient names by ruleset twenty-one. There is a 100,000 byte limit on messages passed through this mailer.

## 18.9. Command Line Arguments

Command-line arguments must appear on the `/usr/lib/sendmail` command line. These arguments are:

`-f` *name* — The sender's name is *name*. This flag is ignored unless the real user is listed as a "trusted user" or if *name* contains an exclamation point (because of certain restrictions in UUCP).

`-r` *name* — An obsolete form of `-f`.

`-h` *cnt* — Sets the "hop count" to *cnt*. This represents the number of times this message has been processed by `sendmail` (to the extent that it is supported by the underlying networks). *cnt* is incremented during processing, and if it reaches the value of configuration option "h,(rq `sendmail` throws away the message with an error.

`-F`*name* — Sets the full name of this user to *name*.

–n — Do not do aliasing or forwarding.

–t — Read the header for "To:," "Cc:," and "Bcc:" lines, and send to everyone listed in those lists. The "Bcc:" line will be deleted before sending. Any names in the argument vector will be deleted from the send list.

`-b`*x* — Set operation mode to *x*. Operation modes are:

| | |
|---|---|
| `m` | Deliver mail (default) |
| `a` | Run in arpanet mode |
| `s` | Speak SMTP on input side |
| `d` | Run as a daemon |
| `t` | Run in test mode |
| `v` | Just verify recipients |
| `i` | Initialize the alias database |
| `p` | Print the mail queue |
| `z` | Freeze the configuration file |

`-q`*time* — Try to process the queued up mail. If the time is given, `sendmail` will repeatedly run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.

`-C`*file* — Use a different configuration file.

`-d`*level* — Set debugging level.

`-o`*xvalue* — Set configuration option *x* to the specified *value.*

`-M` — *Run given message ID from the queue.*

`-R` *Run messages for given recipient only from the queue.*

These options are described in Section 18.10.

Several configuration options may be specified as primitive flags. These are the `c`, `e`, `i`, `m`, `T`, and `v` arguments. Also, you can specify the `f` configuration option as the `-s` argument.

## 18.10. Configuration Options

You can set the following options using the `-o` flag on the command line or the **O** line in the configuration file:

`A`*file* Use the named *file* as the alias file instead of `/etc/aliases`. If no file is specified, use `aliases` in the current directory.

`a`*time* If set, time to wait for an "@:@" entry to exist in the alias database before starting up. If it does not appear after that time, rebuild the database.

`B`*value* Blank substitute. Default is "."

`b`*n* Disallow empty messages to more than *n* recipients.

`C`*n* Checkpoint after *n* recipients.

`c` If an outgoing mailer is marked as being expensive, do not connect immediately. This requires that a queue run processes to actually send the mail.

`D` If set, rebuild the alias database if necessary and possible. If this option is not set, `sendmail` will never rebuild the alias database unless explicitly requested using `-bi`.

`d`*x* 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) |

`e`*x* Dispose of errors using mode *x*. The values for *x* are:

| | |
|---|---|
| `p` | Print error messages (default) |
| `q` | No messages, just give exit status |
| `m` | Mail back errors to sender |
| `w` | Write back errors (mail if user not logged in) |
| `e` | Mail back errors and give zero exit stat always |

`F`*n* The temporary queue file mode, in Octal. 644 and 600 are good choices.

`f` Save UNIX-style "From" lines at the front of headers. Normally they are assumed redundant and discarded.

`g`*n* Set the default group ID for mailers to run in to *n*.

`H`*file* Specify the help file for SMTP [Postel82].

`h` Set maximum hop count to *n*.

`i` Ignore dots in incoming messages.

`L`*n* Set the default log level to *n*.

`M`*xvalue* Set the macro *x* to *value*. This is intended only for use from the command line.

`m` Send to me too, even if I am in an alias expansion.

o — Assume that the headers may be in old format, that is, spaces delimit names. This actually turns on an adaptive algorithm: if any recipient name contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

P — The name of the local Postmaster. If defined, error messages from the MAILER-DAEMON cause the header to be sent to this name.

Q*dir* — Use the named *dir* as the queue directory.

q*limit* — Size limit of messages to be queued under heavy load. Default is 10,000 bytes.

R*server* — Remote mode. Deliver through remote SMTP server. Default is location of `/var/spool/mail`.

r*time* — Timeout reads after *time* interval.

S*file* — Save statistics in the named *file*.

s — Be super-safe when running things, that is, always instantiate the queue file, even if you are going to attempt immediate delivery. `sendmail` always instantiates the queue file before returning control the the client under any circumstances.

T*time* — Set the queue timeout to *time*. After this interval, messages that have not been successfully sent will be returned to the sender.

u*n* — 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.

X*n* — Set the load average value, which causes the `sendmail` daemon to refuse incoming SMTP connections to reduce system load. Default is zero, which disables this feature.

x*n* — Set the load average value which causes `sendmail` to simply queue mail (regardless of the `dx` option) to reduce system load. Default is zero, which disables this feature.

Y*name* — Yellow pages map name to be used for aliases. Default is `mail.aliases`.

y*n* — Recipient factor. Penalize messages with this many bytes-per-recipient.

Z*n* — Time factor. Penalize messages with this many bytes-per-delivery attempts.

z*n* — Message class factor. Penalize messages with this many bytes-per-class.

## 18.11. Mailer Flags

You can set the following flags in the mailer description.

C If mail is *received* from a mailer with this flag set, any names in the header that do not have an at sign (``@'') after being rewritten by ruleset three will have the ``@domain'' clause from the sender tacked on. This allows mail with headers of the form:

```
From: usera@hosta
To: userb@hostb, userc
```

to be rewritten as:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

automatically.

D This mailer wants a ``Date:'' header line.

E Escape ``From'' lines to be ``>From'' (usually specified with `U`).

e This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.

F This mailer wants a ``From:'' header line.

f The mailer wants a `-f` *from* flag, but only if this is a network forward operation (that is, the mailer will give an error if the executing user does not have special permissions).

h Uppercase should be preserved in host names for this mailer.

L Limit the line lengths as specified in RFC821.

l This mailer is local (that is, final delivery will be performed).

M This mailer wants a ``Message-Id:'' header line.

m This mailer can send to multiple users on the same host in one transaction. When a **$u** macro occurs in the `argv` part of the mailer definition, that field will be repeated as necessary for all qualifying users. The L= field of the mailer description can be used to limit the total length of the **$u** expansion.

n Do not insert a UNIX-style ``From'' line on the front of the message.

P This mailer wants a ``Return-Path:'' line.

p Always add local host name to the ``MAIL From:'' line of *SMTP*, even if there already is one.

r Same as **f**, but sends a `-r` flag.

S Do not reset the userid before calling the mailer. This would be used in a secure environment where `sendmail` ran as root. This could be used to avoid forged names.

s Strip quote characters off of the name before calling the mailer.

U This mailer wants UNIX-style ''From'' lines with the ugly UUCP-style ''remote from <host>'' on the end.

u Uppercase should be preserved in user names for this mailer.

X Uses the hidden dot algorithm as specified in 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.

x This mailer wants a ''Full-Name:'' header line.

# 19

# 4.3BSD Line Printer Spooler

# 19

# 4.3BSD Line Printer Spooler

This chapter is based on the *4.3BSD Line Printer Spooler Manual* by Ralph Campbell, Computer Systems Research Group Computer Science Division Department of Electrical Engineering and Computer Science University of California, Berkeley Berkeley, CA 94720. It describes the structure and installation procedure for the line printer spooling system developed for the 4.3BSD version of the UNIX* operating system.

## 19.1. Overview

The line printer system supports:

- Multiple printers
- Multiple spooling queues
- Both local and remote printers
- Printers attached via serial lines that require line initialization such as the baud rate

Raster output devices such as a Varian or Versatec, 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 database |
| `/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 database describing line printers directly attached to a machine and, also, printers accessible across a network. The manual page entry `printcap`(5) provides the authoritative definition of the format of this database, as well as specifying default values for important items such as the directory in which spooling is performed. This document introduces some of the information that may be placed `printcap`.

## 19.2. Commands

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 /etc/printcap file. When lpd is started it makes a single pass through the printcap database restarting any printers that have jobs. In normal operation lpd listens for service requests on multiple sockets, one in the local 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 socket (2) and services (5) for more information on sockets and service specifications, respectively. lpd spawns a copy of itself to process the request; the master daemon continues to listen for new requests.

Clients communicate with lpd using a simple transaction oriented protocol. Authentication of remote clients is done based on the "privilege port" scheme employed by rshd (8C) and rcmd (3X). The following table shows the requests understood by 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.

| Request | Interpretation |
|---|---|
| ^Aprinter\n | check the queue for jobs and print any found |
| ^Bprinter\n | receive and queue a job from another machine |
| ^Cprinter [users ...] [jobs ...]\n | return short list of current queue state |
| ^Dprinter [users ...] [jobs ...]\n | return long list of current queue state |
| ^Eprinter person [users ...] [jobs ...]\n | remove jobs from a queue |

The lpr (1) command is used to enter a print job in a local queue and to notify the local lpd that there are new jobs in the spooling area. lpd either schedules the job to be printed locally, or if printing remotely, attempts to forward the job to the appropriate machine. If the printer cannot be opened or the destination machine is unreachable, the job will remain queued until it is possible to complete the work.

The lpq (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. lpq has two forms of output: in the default, short, format it gives a single line of output per queued job; in the long format it shows the list of files, and their sizes, that comprise a job.

The lprm (1) command deletes jobs from a spooling queue. If necessary, lprm will first kill off a running daemon that is servicing the queue and restart it after the required files are removed. When removing jobs destined for a remote printer, lprm acts similarly to lpq except it first checks locally for jobs to remove and then tries to remove files in queues off-machine.

The lpc(8) program 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.

## 19.3. 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 set-user-id to `root` and set-group-id to group *daemon.* The `root` access permits reading any file required. Accessibility is verified with an `access` (2) call. The group ID is used in setting up proper ownership of files in the spooling area for `lprm`.
- 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 set-user-id to *root* and set-group-id to group *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` or `/etc/hosts.lpd` and the request message must come from a reserved port number.

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 set-user-id to *daemon*, set-group-id to group *spooling*, and `lpq` and `lprm` ran set-group-id to group *spooling*.

## 19.4. Setting Up

The 4.3BSD release comes with the necessary programs installed and with the default line printer queue created. If the system must be modified, the makefile in the directory `/usr/src/usr.lib/lpr` should be used in recompiling and reinstalling the necessary programs.

The real work in setting up is to create the `printcap` file and any printer filters for printers not supported in the distribution system.

### 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 that do not conform to the default printer description (an LP-11 style interface to a standard, band printer).

### 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 Dec-Writer III printer connected locally via a 1200 baud serial line.

```
lp|LA-180 DecWriter III:\
        :lp=/dev/lp:br#1200:fs#06320:\
        :tr=\f:of=/usr/lib/lpf:lf=/var/adm/lpd-errs:
```

The **lp** entry specifies the file name to open for output. Here it could be left out since `/dev/lp` is the default. The **br** entry sets the baud rate for the tty line and the **fs** entry sets CRMOD, no parity, and XTABS (see *tty* (4)). The **tr** entry indicates that 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 that 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 `/var/adm/lpd-errs` instead of the console. Most errors from `lpd` are logged using `syslogd` (8) and will not be logged in the specified file. The filters should use `syslogd` to report errors; only those that write to standard error output will end up with errors in the `lf` file. (Occasionally errors sent to standard error output have not appeared in the log file; the use of *syslogd* is highly recommended.)

### Remote Printers

Printers that reside on remote hosts should have an empty **lp** entry. For example, the following `printcap` entry would send output to the printer named "lp" on the machine ucbvax.

```
lp|default line printer:\
        :lp=:rm=ucbvax:rp=lp:sd=/var/spool/vaxlpd:
```

The **rm** entry is the name of the remote machine to connect to; this name must be a known host name for a machine on the network. The **rp** capability indicates the name of the printer on the remote machine is "lp;" here it could be left out since this is the default value. The **sd** entry specifies `/var/spool/vaxlpd` as the spooling directory instead of the default value of `/var/spool/lpd`.

### Output Filters

Filters are used to handle device dependencies and to do accounting functions. The output filtering of **of** is used when accounting is not being done or when all text data must be passed through a filter. It is not intended to do 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 do accounting if there is an **af** entry.

They are used for producing the following types of jobs:

**if** plain text jobs

**rf** FORTRAN text file jobs (i.e., jobs containing lines of text beginning with FORTRAN carriage control codes?)

**tf** `troff` jobs (i.e., jobs whose files contain Wang C/A/T codes, as produced by `troff`)

**nf** typesetter-independent `troff` jobs (i.e., jobs whose files contain `ditroff` output, as produced by the new typesetter-independent `troff`)

**df** TeX jobs (i.e., jobs whose files contain DVI codes, as produced by TeX)

**cf** CIF jobs (i.e., jobs containing CIF data, as produced by `cifplot`)

**gf** *plot* jobs (i.e., jobs containing *plot* codes, as produced by *plot*(1G))

**vf** raster output file jobs

If a job is to be printed, but there is no filter specified for that type of job, plain text jobs are run through the **of** filter, if any; other jobs are rejected and an error is logged. If there is a filter specified for a particular type of job, and there is an **of** filter, a two-character message is written to the **of** filter, consisting of ^Y followed by ^A. When the **of** filter reads a ^Y from its standard input, it is expected to check whether the next character is a ^A. If so, it is expected to stop itself (e.g., by sending itself a SIGSTOP) so that the other filter can temporarily take control of the printer. Otherwise, it is expected to "unget" the next character and treat the ^Y as regular input. (This means that a **of** filter for a printer to which the sequence ^Y^A may be sent in the course of a normal job has a problem, as this sequence should never be passed through to a printer by a **of** filter.) `lpd` will restart the **of** filter by sending it a SIGCONT when the other filter exits.

An example of a printer that requires output filters is the Benson-Varian.

```
va|varian|Benson-Varian:\
        :lp=/dev/va0:sd=/var/spool/vad:of=/usr/lib/vpf:\
        :tf=/usr/lib/rvcat:mx#2000:pl#58:px=2112:py=1700:tr=\f:
```

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 *va* (4V)). 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** field as shown below, indicating the name of the accounting file.

```
va|varian|Benson-Varian:\
        :lp=/dev/va0:sd=/var/spool/vad:of=/usr/lib/vpf:\
        :if=/usr/lib/vpf:tf=/usr/lib/rvcat:af=/var/adm/vaacct:\
        :mx#2000:pl#58:px=2112:py=1700:tr=\f:
```

## Access Control

Local access to printer queues is controlled with the **rg** printcap entry.

```
:rg=lprgroup:
```

Users must be in the group *lprgroup* to submit jobs to the specified printer. The default is to allow all users access. Note that once the files are in the local queue, they can be printed locally or forwarded to another host depending on the configuration.

Remote access is controlled by listing the hosts in either the file `/etc/hosts.equiv` or `/etc/hosts.lpd`, one host per line. Note that `rsh(1)` and `rlogin(1)` use `/etc/hosts.equiv` to determine which hosts are equivalent for allowing logins without passwords. The file /etc/hosts.lpd is only used to control which hosts have line printer access. Remote access can be further restricted to only allow remote users with accounts on the local host to print jobs by using the **rs** printcap entry.

```
:rs:
```

## 19.5. Output Filter Specifications

The filters supplied with 4.3BSD handle printing and accounting for most common line printers, the Benson-Varian, the wide (36") and narrow (11") Versatec printer/plotters. For other devices or accounting methods, it may be necessary to create a new filter.

Filters are spawned by lpd with their standard input the data to be printed, and standard output the printer. The standard error is attached to the lf file for logging errors or syslogd may be used 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 SIGINT signal to all filters and descendents of filters. This signal can be trapped by filters that need to do 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.

```
filter -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 -n login -h host accounting_file
```

The **–c** flag is optional, and only supplied when control characters are to be passed uninterpreted to the printer (when using the **–l** option of lpr 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.

## 19.6. 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 forcibly restart a hung line printer daemon (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* halts a spooling daemon after the current job completes; this also disables printing. This is a clean way to shutdown a printer to do 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.

## 19.7. Troubleshooting

There are several messages that may be generated by the the line printer system. This section categorizes the most common and explains the cause for their generation. Where the message implies a failure, directions are given to remedy the problem.

In the examples below, the name *printer* is the name of the printer from the `printcap` database.

### lpr

```
lpr: printer : unknown printer
```

The *printer* was not found in the `printcap` database. Usually this is a typing mistake; however, it may indicate a missing or incorrect entry in the `/etc/printcap` file.

```
lpr: printer : jobs queued, but cannot start daemon.
```

The connection to `lpd` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check the local socket `/dev/printer` to be sure it still exists (if it does not exist, there is no `lpd` process running). Usually it is enough to get a superuser to type the following to restart `lpd`.

```
% /usr/lib/lpd
```

You can also check the state of the master printer daemon with the following.

```
% ps l`cat /var/spool/lpd.lock`
```

Another possibility is that the `lpr` program is not set-user-id to *root*, set-group-id to group *daemon*. 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 superuser with `lpc`.

### lpq

```
waiting for printer to become ready (offline ?)
```

The printer device could not be opened by the daemon. This can happen for several reasons, the most common is that the printer is turned 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 enough 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* located in the spooling directory. If the daemon is hung, a superuser can use `lpc` to abort the current daemon and start a new one.

```
waiting for host to come up
```

This implies there is a daemon trying to connect to the remote machine named *host* to send the files in the local queue. If the remote machine is up, lpd on the remote machine is probably dead or hung and should be restarted as mentioned for lpr.

```
sending to host
```

The files should be in the process of being transferred to the remote *host*. If not, the local daemon should be aborted and started with lpc.

```
Warning: printer is down
```

The printer has been marked as being unavailable with lpc.

```
Warning: no daemon present
```

The lpd process overseeing the spooling queue, as specified in the lock file in that directory, does not exist. This normally occurs only when the daemon has unexpectedly died. The error log file for the printer and the *syslogd* logs should be checked for a diagnostic from the deceased process. To restart an lpd, use

```
% lpc restart printer
```

```
no space on remote; waiting for queue to drain
```

This implies that there is insufficient disk space on the remote. If the file is large enough, there will never be enough space on the remote (even after the queue on the remote is empty). The solution here is to move the spooling queue or make more free space on the remote.

### lprm

```
lprm: printer : cannot restart printer daemon
```

This case is the same as when lpr prints that the daemon cannot be started.

### lpd

The lpd program can log many different messages using syslogd(8). Most of these messages are about files that can not be opened and usually imply that the printcap file or the protection modes of the files are incorrect. Files may also be inaccessible if people manually manipulate the line printer system (i.e. they bypass the lpr program).

In addition to messages generated by lpd, any of the filters that lpd spawns may log messages using syslogd or to the error log file (the file specified in the lf entry in printcap).

**lpc**

```
couldn't start printer
```

This case is the same as when `lpr` reports that the daemon cannot be started.

```
cannot examine spool directory
```

Error messages beginning with "cannot ..." are usually because of incorrect ownership or protection mode of the lock file, spooling directory or the `lpc` program.

## NAME

printcap - printer capability data base

## SYNOPSIS

**/etc/printcap**

## DESCRIPTION

**printcap** is a simplified version of the **termcap**(5) data base for describing printers. The spooling system accesses the **printcap** file every time it is used, allowing dynamic addition and deletion of printers. Each entry in the data base describes one printer. This data base may not be substituted for, as is possible for **termcap**, because it may allow accounting to be bypassed.

The default printer is normally **lp**, though the environment variable PRINTER may be used to override this. Each spooling utility supports a **-P***printer* option to explicitly name a destination printer.

Refer to for a discussion of how to set up the database for a given printer. On Sun386i systems, refer to **snap**(1) for information on setting up printers with the system and network administration program.

Each entry in the **printcap** file describes a printer, and is a line consisting of a number of fields separated by ':' characters. The first entry for each printer gives the names which are known for the printer, separated by '|' characters. The first name is conventionally a number. The second name given is the most common abbreviation for the printer, and the last name given should be a long name fully identifying the printer. The second name should contain no blanks; the last name may well contain blanks for readability. Entries may continue onto multiple lines by giving a '\' as the last character of a line, and empty fields may be included for readability.

Capabilities in **printcap** are all introduced by two-character codes, and are of three types:

*Boolean* Capabilities that indicate that the printer has some particular feature. Boolean capabilities are simply written between the ':' characters, and are indicated by the word 'bool' in the **type** column of the capabilities table below.

*Numeric* Capabilities that supply information such as baud-rates, number of lines per page, and so on. Numeric capabilities are indicated by the word **num** in the **type** column of the capabilities table below. Numeric capabilities are given by the two-character capability code followed by the '#' character, followed by the numeric value. For example:

```
:br#1200:
```

is a numeric entry stating that this printer should run at 1200 baud.

*String* Capabilities that give a sequence which can be used to perform particular printer operations such as cursor motion. String valued capabilities are indicated by the word **str** in the **type** column of the capabilities table below. String valued capabilities are given by the two-character capability code followed by an '=' sign and then a string ending at the next following ':'. For example,

```
:rp=spinwriter:
```

is a sample entry stating that the remote printer is named **spinwriter**.

## Sun386i DESCRIPTION

On Sun386i systems, **lpr(1)** and related printing commands use the Yellow Pages name service to obtain the **printcap** entry for a named printer if the entry does not exist in the local **/etc/printcap** file. For example, when a user issues the command

```
lpr -Pnewprinter foo
```

**lpr** searches **/etc/printcap** on the local system for an entry for **newprinter**. If no local entry for **newprinter** exists, then **lpr** searches the YP map called **printcap**. The search is invisible to the user.

**lpr** creates the spooling directory for the printer automatically if no spooling directory exists.

System administrators can make a printer available to the entire YP domain by placing an entry for that printer in the YP **printcap** map, typically using **snap**. Otherwise, the system administrator must edit the **/etc/printcap** file on the YP master and then rebuild the YP map.

## CAPABILITIES

| *Name* | *Type* | *Default* | *Description* |
|---|---|---|---|
| **af** | str | NULL | name of accounting file |
| **br** | num | none | if lp is a tty, set the baud rate (ioctl call) |
| **cf** | str | NULL | cifplot data filter |
| **df** | str | NULL | TeX data filter (DVI format) |
| **du** | str | 0 | User ID of user 'daemon'. |
| **fc** | num | 0 | if lp is a tty, clear flag bits |
| **ff** | str | “\f” | string to send for a form feed |
| **fo** | bool | false | print a form feed when device is opened |
| **fs** | num | 0 | like 'fc' but set bits |
| **gf** | str | NULL | graph data filter (plot(3X) format) |
| **hl** | bool | false | print the burst header page last |
| **ic** | bool | false | driver supports (non standard) ioctl to indent printout |
| **if** | str | NULL | name of input/communication filter (created per job) |
| **lf** | str | “/dev/console” | error logging file name |
| **lo** | str | “lock” | name of lock file |
| **lp** | str | “/dev/lp” | device name to open for output |
| **mc** | num | 0 | maximum number of copies |
| **ms** | str | NULL | list of terminal modes to set or clear |
| **mx** | num | 1000 | maximum file size (in BUFSIZ blocks), zero = unlimited |
| **nd** | str | NULL | next directory for list of queues (unimplemented) |
| **nf** | str | NULL | ditroff data filter (device independent troff) |
| **of** | str | NULL | name of output/banner filter (created once) |
| **pc** | num | 200 | price per foot or page in hundredths of cents |
| **pl** | num | 66 | page length (in lines) |
| **pw** | num | 132 | page width (in characters) |
| **px** | num | 0 | page width in pixels (horizontal) |
| **py** | num | 0 | page length in pixels (vertical) |
| **rf** | str | NULL | filter for printing FORTRAN style text files |
| **rg** | str | NULL | restricted group. Only members of group allowed access |
| **rm** | str | NULL | machine name for remote printer |
| **rp** | str | “lp” | remote printer name argument |
| **rs** | bool | false | restrict remote users to those with local accounts |
| **rw** | bool | false | open printer device read/write instead of read-only |
| **sb** | bool | false | short banner (one line only) |
| **sc** | bool | false | suppress multiple copies |
| **sd** | str | “/var/spool/lpd” | spool directory |
| **sf** | bool | false | suppress form feeds |
| **sh** | bool | false | suppress printing of burst page header |
| **st** | str | “status” | status file name |
| **tc** | str | NULL | name of similar printer; must be last |
| **tf** | str | NULL | troff data filter (C/A/T phototypesetter) |
| **tr** | str | NULL | trailer string to print when queue empties |
| **vf** | str | NULL | raster image filter |
| **xc** | num | 0 | if lp is a tty, clear local mode bits |
| **xs** | num | 0 | like 'xc' but set bits |

If the local line printer driver supports indentation, the daemon must understand how to invoke it.

Note: the **fs**, **fc**, **xs**, and **xc** fields are flag *masks* rather than flag *values*. Certain default device flags are set when the device is opened by the line printer daemon if the device is connected to a terminal port. The flags indicated in the **fc** field are then cleared; the flags in the **fs** field are then set (or vice-versa, depending on the order of **fc**#*nnnn* and **fs**#*nnnn* in the **/etc/printcap** file). The bits cleared by the **fc** field and set by the **fs** field are those in the **sg_flags** field of the **sgtty** structure, as set by the **ioctl** call, and the bits cleared by the **xc** field and set by the **xs** field are those in the local flags word, as set by the **ioctl** call. See **ttcompat**(4M) for a description of these flags. For example, to set exactly the flags 06300 in the **fs** field, which specifies that the **EVENP**, **ODDP**, and modes are to be set, and all other flags are to be cleared, do:

```
:fc#0177777:fs#06300:
```

The same process applies to the **xc** and **xs** fields. Alternatively, the **ms** field can be used to specify modes to be set and cleared. These modes are specified as **stty**(1V) modes; any mode supported by **stty** may be specified, except for the baud rate which must be specified with the **br** field. This permits modes not supported by the older terminal interface described in **ttcompat**(4M) to be set or cleared. Thus, to set the terminal port to which the printer is attached to even parity, tab expansion, no newline to carriage-return/line-feed translation, and RTS/CTS flow control enabled, do:

```
:ms=evenp,-tabs,nl,crtscts:
```

On Sun386i systems, the **tc** field, as in the **termcap**(5) file, must appear last in the list of capabilities. It is recommended that each type of printer have a general entry describing common capabilities; then an individual printer can be defined with its particular capabilities plus a **tc** field that points to the general entry for that type of printer.

## FILES

**/etc/printcap**

## SEE ALSO

**lpq**(1), **lpr**(1), **lprm**(1), **snap**(1), **stty**(1V), **plot**(3X), **ttcompat**(4M,, **termcap**(5), **lpc**(8), **lpd**(8), **pac**(8)

# 20

# Modifying the `termcap` File

# 20

# Modifying the termcap File

termcap is a database describing terminals, used, for example by vi(1) and curses(3X). Terminals are described in termcap by giving a set of capabilities that they have and by describing how operations are performed. Padding requirements and initialization sequences are included in termcap.

Entries in termcap consist of a number of colon-separated (:) fields. The first entry for each terminal gives the names that are known for the terminal, separated by pipe (|) characters. The first name is always two characters long and is used by older systems that store the terminal type in a 16-bit word in a system-wide database. The second name given is the most common abbreviation for the terminal: the last name given should be a long name fully identifying the terminal: and all others are understood as synonyms for the terminal name. All names but the first and last should be in lowercase and contain no blanks; the last name may well contain upper case and blanks for readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions. The particular piece of hardware making up the terminal should have a root name chosen, thus "hp2621". This name should not contain hyphens. Modes that the hardware can be in or user preferences should be indicated by appending a hyphen and an indicator of the mode. Therefore, a "vt100" in 132-column mode would be "vt100-w". The following suffixes should be used where possible:

| Suffix | Meaning | Example |
|---|---|---|
| -w | Wide mode (more than 80 columns) | vt100-w |
| -am | With automatic margins (usually default) | vt100-am |
| -nam | Without automatic margins | vt100-nam |
| -*n* | Number of lines on the screen | aaa-60 |
| -na | No arrow keys (leave them in local) | concept100-na |
| -*n*p | Number of pages of memory | concept100-4p |
| -rv | Reverse video | concept100-rv |

## 20.1. Types of Capabilities

Capabilities in termcap are of three types: Boolean capabilities, which indicate particular features that the terminal has; numeric capabilities, giving the size of the display or the size of other attributes; and string capabilities, which give character sequences that can be used to perform particular terminal operations. All capabilities have two-letter codes. For instance, the fact that the Concept has *automatic* margins , (for example, an automatic return and linefeed when the end of a line is reached) is indicated by the Boolean capability **am**. Hence the description of the Concept includes **am**.

Numeric capabilities are followed by the character # then the value. In the example above **co**, which indicates the number of columns the display has, gives the value 80 for the Concept.

Finally, string-valued capabilities, such as **ce** (clear-to-end-of-line sequence) are given by the two-letter code, an =, then a string ending at the next following colon. A delay in milliseconds may appear after the = in such a capability, which causes padding characters to be supplied by `tputs` after the remainder of the string is sent to provide this delay. The delay can be either a number, *e.g.* 20, or a number followed by an *, for example, 3*. An * indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-line padding required. (In the case of insert-character, the factor is still the number of *lines* affected; this is always 1 unless the terminal has **in** and the software uses it.) When an * is specified, it is sometimes useful to give a delay of the form 3.5 to specify a delay per line to tenths of milliseconds. (Only one decimal place is allowed.)

A number of escape sequences are provided in the string-valued capabilities for easy encoding of control characters there. **\E** maps to an ESC character, **^X** maps to a control-X for any appropriate X, and the sequences **\n \r \t \b \f** map to linefeed, return, tab, backspace, and formfeed, respectively. Finally, characters may be given as three octal digits after a \, and the characters ^ and \ may be given as \^ and \\. If it is necessary to place a : in a capability it must be escaped in octal as **\072**. If it is necessary to place a NUL character in a string capability it must be encoded as **\200**. (The routines that deal with `termcap` use C strings and strip the high bits of the output very late, so that a **\200** comes out as a **\000** would.)

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the first **cr** and **ta** in the example above.

## 20.2. Preparing Descriptions

We now outline how to prepare descriptions of terminals. The most effective way to prepare a terminal description is by imitating the description of a similar terminal in *termcap* and to build up a description gradually, using partial descriptions with `vi` to check that they are correct. Be aware that a very unusual terminal may expose deficiencies in the ability of the `termcap` file to describe it or bugs in *vi*. To easily test a new terminal description you can set the environment variable TERMCAP to the absolute pathname of a file containing the description you are working on and programs will look there rather than in `/usr/text/lib/termcap`. TERMCAP can also be set to the `termcap` entry itself to avoid reading the file when starting up a program.

To get the padding for insert-line right (if the terminal manufacturer did not document it), a severe test is to use `vi` to edit `/etc/passwd` at 9600 baud, delete roughly 16 lines from the middle of the screen, then hit the 'u' key several times quickly. If the display messes up, more padding is usually needed. A similar test can be used for insert-character.

## 20.3. Basic Capabilities

The number of columns on each line of the display is given by the **co** numeric capability. If the display is a CRT, then the number of lines on the screen is given by the **li** capability. If the display wraps around to the beginning of the next line when the cursor reaches the right margin, then it should have the **am** capability. If the terminal can clear its screen, the code to do this is given by the **cl** string capability. If the terminal overstrikes (rather than clearing the position when a character is overwritten), it should have the **os** capability. If the terminal is a printing terminal, with no soft copy unit, give it both **hc** and **os**. (**os** applies to storage scope terminals, such as the Tektronix 4010 series, as well as to hard copy and APL terminals.) If there is a code to move the cursor to the left edge of the current row, give this as **cr**. (Normally this will be carriage-return, ^M.) If there is a code to produce an audible signal (bell, beep, *etc.*), give this as **bl**.

If there is a code (such as backspace) to move the cursor one position to the left, that capability should be given as **le**. Similarly, codes to move to the right, up, and down should be given as **nd**, **up**, and **do**, respectively. These *local cursor motions* should not alter the text they pass over; for example, you would not normally use "nd= " unless the terminal has the **os** capability, because the space would erase the character moved over.

A very important point here is that the local cursor motions encoded in `termcap` have undefined behavior at the left and top edges of a CRT display. Programs should never attempt to backspace around the left edge, unless **bw** is given, and never attempt to go up off the top using local cursor motions.

In order to scroll text up, a program goes to the bottom left corner of the screen and sends the **sf** (index) string. To scroll text down, a program goes to the top left corner of the screen and sends the **sr** (reverse index) string. The strings **sf** and **sr** have undefined behavior when not on their respective corners of the screen. Parameterized versions of the scrolling sequences are **SF** and **SR**, which have the same semantics as **sf** and **sr** except that they take one parameter and scroll that many lines. They also have undefined behavior except at the appropriate corner of the screen.

The **am** capability tells whether the cursor sticks at the right edge of the screen when text is output there, but this does not necessarily apply to **nd** from the last column. Leftward local motion is defined from the left edge only when **bw** is given; then an **le** from the left edge will move to the right edge of the previous row. This is useful for drawing a box around the edge of the screen, for example. If the terminal has switch-selectable automatic margins, the `termcap` description usually assumes that this feature is on, for example, **am**. If the terminal has a command that moves to the first column of the next line, that command can be given as **nw** (newline). It is permissible for this to clear the remainder of the current line, so if the terminal has no correctly-working CR and LF it may still be possible to craft a working **nw** out of one or both of them.

These capabilities suffice to describe hardcopy and glass-tty terminals. Thus the Teletype model 33 is described as

```
T3|tty33|33|tty|Teletype model 33:\
        :bl=^G:co#72:cr=^M:do=^J:hc:os:
```

and the Lear Siegler ADM–3 is described as

```
l3|adm3|3|LSI ADM-3:\
        :am:bl=^G:cl=^Z:co#80:cr=^M:do=^J:le=^H:li#24:sf=^J:
```

## 20.4. Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with `printf`(3S)-like escapes **%x** in it, while other characters are passed through unchanged. For example, to address the cursor the **cm** capability is given, using two parameters: the row and column to move to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory. If the terminal has memory-relative cursor addressing, that can be indicated by an analogous **CM** capability.)

The % encodings have the following meanings:

| | |
|---|---|
| %% | output '%' |
| %d | output value as in *printf* %d |
| %2 | output value as in *printf* %2d |
| %3 | output value as in *printf* %3d |
| %. | output value as in *printf* %c |
| %+*x* | add *x* to value, then do %. |
| %>*xy* | if value > *x* then add *y*, no output |
| %r | reverse order of two parameters, no output |
| %i | increment by one, no output |
| %n | exclusive-or all parameters with 0140 (Datamedia 2500) |
| %B | BCD (16*(value/10)) + (value%10), no output |
| %D | Reverse coding (value – 2*(value%16)), no output (Delta Data) |

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent \E&a12c03Y padded for 6 milliseconds. Note that the order of the row and column coordinates is reversed here and that the row and column are sent as two-digit integers. Thus its **cm** capability is cm=6\E&%r%2c%2Y.

The Microterm ACT-IV needs the current row and column sent simply encoded in binary preceded by a **^T**, cm=^T%.%.. Terminals that use %. need to be able to backspace the cursor (**le**) and to move the cursor up one line on the screen (**up**). This is necessary because it is not always safe to transmit **\n**, **^D**, and **\r**, as the system may change or discard them. (Programs using `termcap` must set terminal modes so that tabs are not expanded, so \t is safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the Lear Siegler ADM–3a, which offsets row and column by a blank character, thus cm=\E=%+ %+ .

Row or column absolute cursor addressing can be given as single parameter capabilities **ch** (horizontal position absolute) and **cv** (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to **cm**. If there are parameterized local motions (*e.g.*, move *n* positions to the right) these can be given as **DO**, **LE**, **RI**, and **UP** with a single parameter indicating how many positions to move. These are primarily useful if the terminal does not have **cm**, such as the Tektronix 4025.

## 20.5. Cursor Motions

If the terminal has a fast way to home the cursor (to the very upper left corner of the screen), this can be given as **ho**. Similarly, a fast way of getting to the lower left-hand corner can be given as **ll**; this may involve going up with **up** from the home position, but a program should never do this itself (unless **ll** does), because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as cursor address (0,0): to the top left corner of the screen, not of memory. (Therefore, the \EH sequence on Hewlett-Packard terminals cannot be used for **ho**.)

## 20.6. Area Clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as **ce**. If the terminal can clear from the current position to the end of the display, this should be given as **cd**. **cd** must only be invoked from the first column of a line. (Therefore, it can be simulated by a request to delete a large number of lines, if a true **cd** is not available.)

## 20.7. Insert/Delete Line

If the terminal can open a new blank line before the line containing the cursor, this should be given as **al**; this must be invoked only from the first position of a line. The cursor must then appear at the left of the newly blank line. If the terminal can delete the line that the cursor is on, this should be given as **dl**; this must only be used from the first position on the line to be deleted. Versions of **al** and **dl** which take a single parameter and insert or delete that many lines can be given as **AL** and **DL**. If the terminal has a settable scrolling region (like the VT100), the command to set this can be described with the **cs** capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is, alas, undefined after using this command. It is possible to get the effect of insert or delete line using this command — the **sc** and **rc** (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using **sr** or **sf** on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

If the terminal has the ability to define a window as part of memory which all commands affect, it should be given as the parameterized string **wi**. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order. (This `terminfo` capability is described for completeness. It is unlikely that any `termcap` -using program will support it.)

If the terminal can retain display memory above the screen, then the **da** capability should be given; if display memory can be retained below, then **db** should be given. These indicate that deleting a line or scrolling may bring non-blank lines up from below or that scrolling back with **sr** may bring down non-blank lines.

## 20.8. Insert/Delete Character

There are two basic kinds of intelligent terminals with respect to insert/delete character that can be described using `termcap`. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept–100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen then typing text separated by cursor motions. Type "abc    def" using local cursor motions (not spaces) between the "abc" and the "def". Then position the cursor before the abc and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the "abc" shifts over to the "def" which then move together around the end of the current line and onto the next as you insert, then you have the second type of terminal and should give the capability **in**, which stands for "insert null". While these are two logically separate attributes (one line *vs.* multi-line insert mode, and special treatment of untyped spaces), we have seen no terminals whose insert mode cannot be described with the single attribute.

`termcap` can describe both terminals that have an insert mode and terminals that send a simple sequence to open a blank position on the current line. Give as **im** the sequence to get into insert mode. Give as **ei** the sequence to leave insert mode. Now give as **ic** any sequence that needs to be sent just before each character to be inserted. Most terminals with a true insert mode will not give **ic**; terminals that use a sequence to open a screen position should give it here. (If your terminal has both, insert mode is usually preferable to **ic**. Do not give both unless the terminal actually requires both to be used in combination.) If post-insert padding is needed, give this as a number of milliseconds in **ip** (a string option). Any other sequence that may need to be sent after insertion of a single character can also be given in **ip**. If your terminal needs to be placed into an 'insert mode' and needs a special code preceding each inserted character, then both **im**/**ei** and **ic** can be given, and both will be used. The **IC** capability, with one parameter ***n***, will repeat the effects of **ic** ***n*** times.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (*e.g.*, if there is a tab after the insertion position). If your terminal allows motion while in insert mode, you can give the capability **mi** to speed up inserting in this case. Omitting **mi** will affect only speed. Some terminals (notably Datamedia's) must not have **mi** because of the way their insert mode works.

Finally, you can specify **dc** to delete a single character, **DC** with one parameter *n* to delete *n* characters, and delete mode by giving **dm** and **ed** to enter and exit delete mode (which is any mode the terminal needs to be placed in for **dc** to work).

## 20.9. Highlighting, Underlining, and Visible Bells

If your terminal has one or more kinds of display attributes, these can be represented in a number of different ways. You should choose one display form as *standout mode*, representing a good high-contrast, easy-on-the-eyes format for highlighting error messages and other attention getters. (If you have a choice, reverse video plus half-bright is good, or reverse video alone.) The sequences to enter and exit standout mode are given as **so** and **se**, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces or garbage characters on the screen, as the TVI 912 and Teleray 1061 do, then **sg** should be given to tell how many characters are left.

Codes to begin underlining and end underlining can be given as **us** and **ue**, respectively. Underline mode change garbage is specified by **ug**, similar to **sg**. If the terminal has a code to underline the current character and move the cursor one position to the right, such as the Microterm Mime, this can be given as **uc**.

Other capabilities to enter various highlighting modes include **mb** (blinking), **md** (bold or extra bright), **mh** (dim or half-bright), **mk** (blanking or invisible text), **mp** (protected), **mr** (reverse video), **me** (turn off *all* attribute modes), **as** (enter alternate character set mode), and **ae** (exit alternate character set mode). Turning on any of these modes singly may or may not turn off other modes.

If there is a sequence to set arbitrary combinations of mode, this should be given as **sa** (set attributes), taking 9 parameters. Each parameter is either 0 or 1, as the corresponding attributes is on or off. The 9 parameters are, in order: standout, underline, reverse, blink, dim, bold, blank, protect, and alternate character set. Not all modes need be supported by **sa**, only those for which corresponding attribute commands exist. (It is unlikely that a `termcap` -using program will support this capability, which is defined for compatibility with `terminfo` .)

Terminals with the "magic cookie" glitches (**sg** and **ug**), rather than maintaining extra attribute bits for each character cell, instead deposit special "cookies", or "garbage characters", when they receive mode-setting sequences, which affect the display algorithm.

Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when they move to a new line or when the cursor is addressed. Programs using standout mode should exit standout mode on such terminals before moving the cursor or sending a newline. On terminals where this is not a problem, the **ms** capability should be present to say that this overhead is unnecessary.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), this can be given as **vb**; it must not move the cursor.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to change, for example, a non-blinking underline into an easier-to-find block or blinking underline), give this sequence as **vs**. If there is a way to make the cursor completely invisible, give that as **vi**. The capability **ve**, which undoes the effects of both of these modes, should also be given.

If your terminal correctly displays underlined characters (with no special codes needed) even though it does not overstrike, then you should give the capability **ul**. If overstrikes are erasable with a blank, this should be indicated by giving **eo**.

### 20.10. Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, this information can be given. Note that it is not possible to handle terminals where the keypad only works in local mode (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, give these codes as **ks** and **ke**. Otherwise the keypad is assumed to always transmit. The codes sent by the left-arrow, right-arrow, up-arrow, down-arrow, and home keys can be given as **kl**, **kr**, **ku**, **kd**, and **kh**, respectively. If there are function keys such as f0, f1, ..., f9, the codes they send can be given as **k0**, **k1**,..., **k9**. If these keys have labels other than the default f0 through f9, the labels can be given as **l0**, **l1**,..., **l9**. The codes transmitted by certain other special keys can be given: **kH** (home down), **kb** (backspace), **ka** (clear all tabs), **kt** (clear the tab stop in this column), **kC** (clear screen or erase), **kD** (delete character), **kL** (delete line), **kM** (exit insert mode), **kE** (clear to end of line), **kS** (clear to end of screen), **kI** (insert character or enter insert mode), **kA** (insert line), **kN** (next page), **kP** (previous page), **kF** (scroll forward/down), **kR** (scroll backward/up), and **kT** (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, then the other five keys can be given as **K1**, **K2**, **K3**, **K4**, and **K5**. These keys are useful when the effects of a 3 by 3 directional pad are needed. The obsolete **ko** capability formerly used to describe other function keys has been completely supplanted by the above capabilities.

The **ma** entry is also used to indicate arrow keys on terminals that have single-character arrow keys. It is obsolete but still in use in version 2 of *vi* which must be run on some minicomputers due to memory limitations. This field is redundant with **kl**, **kr**, **ku**, **kd**, and **kh**. It consists of groups of two characters. In each group, the first character is what an arrow key sends, and the second character is the corresponding `vi` command. These commands are **h** for **kl**, **j** for **kd**, **k** for **ku**, **l** for **kr**, and **H** for **kh**. For example, the Mime would have ma=^Hh^Kj^Zk^Xl indicating arrow keys left (^H), down (^K), up (^Z), and right (^X). (There is no home key on the Mime.)

### 20.11. Tabs and Initialization

If the terminal needs to be in a special mode when running a program that uses these capabilities, the codes to enter and exit this mode can be given as **ti** and **te**. This arises, for example, from terminals like the Concept with more than one page of memory. If the terminal has only memory-relative cursor addressing and not screen-relative cursor addressing, a screen-sized window must be fixed into the display for cursor addressing to work properly. This is also used for the Tektronix 4025, where **ti** sets the command character to be the one used by `termcap`.

Other capabilities include **is**, an initialization string for the terminal, and **if**, the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the `termcap` description. They are normally sent to the terminal by the `tset` program each time the user logs in. They will be printed in the following order: **is**; setting tabs using **ct** and **st**; and finally **if**. (`terminfo` uses **i1** before **is** and runs the program **iP** and prints **i3** after the other initializations.) A pair of sequences that does a harder reset from a totally unknown state can be analogously given as **rs** and **if**. These strings are output by the *reset* program, which is used when the terminal gets into a wedged state. (`terminfo` uses **r1** before **rs** and **r3** after.) Commands are normally placed in **rs** and **rf** only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set the VT100 into 80-column mode would normally be part of **is**, but it causes an annoying glitch of the screen and is not normally needed since the terminal is usually already in 80-column mode.

If the terminal has hardware tabs, the command to advance to the next tab stop can be given as **ta** (usually ^I). A backtab command which moves leftward to the previous tab stop can be given as **bt**. By convention, if the terminal driver modes indicate that tab stops are being expanded by the computer rather than being sent to the terminal, programs should not use **ta** or **bt** even if they are present, since the user may not have the tab stops properly set. If the terminal has hardware tabs that are initially set every *n* positions when the terminal is powered up, then the numeric parameter **it** is given, showing the number of positions between tab stops. This is normally used by the `tset` command to determine whether to set the driver mode for hardware tab expansion, and whether to set the tab stops. If the terminal has tab stops that can be saved in nonvolatile memory, the `termcap` description can assume that they are properly set.

If there are commands to set and clear tab stops, they can be given as **ct** (clear all tab stops) and **st** (set a tab stop in the current column of every row). If a more complex sequence is needed to set the tabs than can be described by this, the sequence can be placed in **is** or **if**.

## 20.12. Delays

Certain capabilities control padding in the terminal driver. These are primarily needed by hardcopy terminals and are used by the `tset` program to set terminal driver modes appropriately. Delays embedded in the capabilities **cr**, **sf**, **le**, **ff**, and **ta** will cause the appropriate delay bits to be set in the terminal driver. If **pb** (padding baud rate) is given, these values can be ignored at baud rates below the value of **pb**. For 4.2BSD *tset*, the delays are given as numeric capabilities **dC**, **dN**, **dB**, **dF**, and **dT** instead.

## 20.13. Miscellaneous

If the terminal requires other than a NUL (zero) character as a pad, this can be given as **pc**. Only the first character of the **pc** string is used.

If the terminal has commands to save and restore the position of the cursor, give them as **sc** and **rc**.

If the terminal has an extra status line that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, then the capability **hs** should be given. Special strings to go to a position in the status line and to return from the status line can be given as **ts** and **fs**. (**fs** must leave the cursor position in the same place that it was before **ts**. If necessary, the **sc** and **rc** strings can be included in **ts** and **fs** to get this effect.) The capability **ts** takes one parameter, which is the column number of the status line to which the cursor is to be moved. If escape sequences and other special commands such as tab work while in the status line, the flag **es** can be given. A string that turns off the status line (or otherwise erases its contents) should be given as **ds**. The status line is normally assumed to be the same width as the rest of the screen, for example, **co**. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded), then its width in columns can be indicated with the numeric parameter **ws**.

If the terminal can move up or down half a line, this can be indicated with **hu** (half-line up) and **hd** (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as **ff** (usually ^L).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters), this can be indicated with the parameterized string **rp**. The first parameter is the character to be repeated and the second is the number of times to repeat it. (This is a `terminfo` feature that is unlikely to be supported by a program that uses `termcap`.)

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with **CC**. A prototype command character is chosen which is used in all capabilities. This character is given in the **CC** capability to identify it. The following convention is supported on some UNIX systems: The environment is to be searched for a `CC` variable, and if found, all occurrences of the prototype character are replaced by the character in the environment variable. This use of the `CC` environment variable is a very bad idea, as it conflicts with `make(1)`.

Terminal descriptions that do not represent a specific kind of known terminal, such as *switch*, *dialup*, *patch*, and *network*, should include the **gn** (generic) capability so that programs can complain that they do not know how to talk to the terminal. (This capability does not apply to *virtual* terminal descriptions for which the escape sequences are known.)

If the terminal uses xoff/xon (DC3/DC1) handshaking for flow control, give **xo**. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted.

If the terminal has a meta key which acts as a shift key, setting the 8th bit of any character transmitted, then this fact can be indicated with **km**. Otherwise, software will assume that the 8th bit is parity and it will usually be cleared. If strings exist to turn this meta mode on and off, they can be given as **mm** and **mo**.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with **lm**. An explicit value of 0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

If the terminal is one of those supported by the UNIX system virtual terminal protocol, the terminal number can be given as **vt**.

Media copy strings which control an auxiliary printer connected to the terminal can be given as **ps**: print the contents of the screen; **pf**: turn off the printer; and **po**: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. It is undefined whether the text is also displayed on the terminal screen when the printer is on. A variation **pO** takes one parameter and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. All text, including **pf**, is transparently passed to the printer while **pO** is in effect.

Strings to program function keys can be given as **pk**, **pl**, and **px**. Each of these strings takes two parameters: the function key number to program (from 0 to 9) and the string to program it with. Function key numbers out of this range may program undefined keys in a terminal-dependent manner. The differences among the capabilities are that **pk** causes pressing the given key to be the same as the user typing the given string; **pl** causes the string to be executed by the terminal in local mode; and **px** causes the string to be transmitted to the computer. Unfortunately, due to lack of a definition for string parameters in `termcap`, only `terminfo` supports these capabilities.

## 20.14. Glitches and Braindamage

Hazeltine terminals, which do not allow '~' characters to be displayed, should indicate **hz**.

The **nc** capability, now obsolete, formerly indicated Datamedia terminals, which echo \r\n for carriage return then ignore a following linefeed.

Terminals that ignore a linefeed immediately after an **am** wrap, such as the Concept, should indicate **xn**.

If **ce** is required to get rid of standout (instead of merely writing normal text on top of it), **xs** should be given.

Teleray terminals, where tabs turn all characters moved over to blanks, should indicate **xt** (destructive tabs). This glitch is also taken to mean that it is not possible to position the cursor on top of a "magic cookie", and that to erase standout mode it is necessary to use delete and insert line.

The Beehive Superbee, which is unable to correctly transmit the ESC or ^C characters, has **xb**, indicating that the "f1" key is used for ESC and "f2" for ^C. (Only certain Superbees have this problem, depending on the ROM.)

Other specific terminal problems may be corrected by adding more capabilities of the form **x***x*.

## 20.15. Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability **tc** can be given with the name of the similar terminal. This capability must be *last*, and the combined length of the entries must not exceed 1024. The capabilities given before **tc** override those in the terminal type invoked by **tc**. A capability can be canceled by placing **xx@** to the left of the **tc** invocation, where *xx* is the capability. For example, the entry

```
hn | 2621-nl:ks@:ke@:tc=2621:
```

defines a 2621–nl that does not have the **ks** or **ke** capabilities, hence does not turn on the function key labels when in visual mode. This is useful for different modes for a terminal, or for different user preferences.

NOTES:

`termcap` *was replaced by* `terminfo` *in UNIX System V Release 2.0. The transition will be relatively painless if capabilities flagged as "obsolete" are avoided.*

`vi` *allows only 256 characters for string capabilities, and the routines in* `termlib`*(3) do not check for overflow of this buffer. The total length of a single entry (excluding only escaped newlines) may not exceed 1024.*

Not all programs support all entries.

# 21

# UUCP Implementation Description

# 21

# UUCP Implementation Description

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. 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. |
| `uux` | 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. |
| `uuxqt` | 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. |
| `uuname` | lists the `uucp` names of systems that can be accessed using `uucp`. |

The remaining sections of this manual describe the operation of each program, security, installation details, files required for execution, and administration of the *uucp* system.

## 21.1. 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.

–e*sys* Send this job to system *sys* to execute. Note that this only works when the system *sys* allows `uuxqt` to execute a `uucp` command. See the sections entitled `uuxqt` and *Security*.

–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:

–s*dir* Use *dir* as the spool directory.

–r Queue the job but do not start `uucico` program.

–x*num* *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 pathnames, the current directory is pre-pended 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).

### Type 1 — Local Copy

The copy is done locally. The `–m` and `–n` options are not honored in this case.

### 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 `uucp.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 pathname of the source or a *~something/pathname*. The *~something* part is expanded on the remote system.

[3] The full pathname of the destination file. If the *~something* notation is used, it is immediately expanded.

[4] The user's login name.

[5] A – followed by an option list. The options `-m` and `-d` may appear.

### 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

[2] The full pathname of the source file.

[3] The full pathname of the destination or *~something/filename*.

[4] The user's login 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.

[7] The file mode bits of the source file in octal print format (666 for instance).

[8] The user on the remote system who should be notified upon completion of the file copy when the `-n` option is specified.

### 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.

### 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 `uuxqt` program allows the `uucp` command to be executed.

## 21.2. UUX — UNIX to UNIX Execution

The `uux` 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 `uux` command is

```
uux [-] [ option ] ... command-string
```

where *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 that do not contain a ! character are assumed to be part of the command string and are not treated as files (that is, `uux` 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 `uux` command.

The following options are available for `uux`:

– Read from the standard input.

–x Read from the standard input.

–n Do not notify the requester (by mail) of completion status. Note that `-n` controls whether or not the exit status from the command that is executed is returned to the requester.

–z Only notify the requester (by mail) of non-zero completion status.

The following options are available for debugging:

–r Don't start `uucico` or `uuxqt` after queuing the job.

–x*num* *num* is a level number between 1 and 9; higher numbers give more debugging output.

The command:

```
pr abc |  uux - usg ! lpr
```

sets up the output of the `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 `uuxqt` 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 requester'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 `uuxqt` to execute.

### User Line

```
U user system
```

where the *user* and *system* are the requester's login name and system.

### 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 `uuxqt` program checks for the existence of all these files before the command is executed.

### 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.

### 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.

### Command Line

```
C   command fR[ 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/spool/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 pre-pended to the command line as specified in the *uuxqt* program.

Note that a check is made to see that the command is allowed as specified in the `uuxqt` program. After execution, the standard output is copied or sent to the proper place.

## 21.3. UUXQT — UUCP Command Execution

The `uuxqt` program executes scripts generated by `uux`.

The `uuxqt` program may be started by either the `uucico` or `uux` programs or a daemon specified by a `crontab` entry. `uuxqt` 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 "`uux`," 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.

`uuxqt` accepts the standard debugging option:

–x*num* *num* is a level number between 1 and 9; higher numbers give more debug output.

## 21.4. UUCICO — Copy In, Copy Out

The `uucico` program performs several major functions:

- □ Scans the spool directory for work.
- □ Places a call to a remote system.
- □ Negotiates a line protocol to be used.
- □ Executes all requests from both systems.
- □ Logs 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`, `uux`, `uuxqt`, 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:

–r1 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:

–d*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.

## Scan for Work

The names of the work-related files in the spool directory have the format:

```
type . system-name grade number
```

| | |
|---|---|
| *type* | An uppercase letter ( *C* – copy command file, *D* – data file, *X* – execute file). |
| *system-name* | The remote system, *truncated to seven characters.* |
| *grade* | A character. |
| *number* | 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.

## Call Remote System

The call is made using information from several files that reside in `/etc/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 log in.

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.

## Line Protocol Selection

The remote system sends a message:

```
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

```
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.

## 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 `-r1` option of `uucico`). Each message used during the work processing is specified by the first character of the message:

| | |
|---|---|
| S | Send a file. |
| R | Receive a file. |
| C | Copy complete. |
| X | Execute a `uucp` command. |
| H | Hang up. |

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 reson 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.

### 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 terminates. The *MASTER* then proceeds to call other systems unless a `–s` option was specified.

## 21.5. UULOG — UUCP Log Inquiry

When a `uucp` program cannot 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:

–s*sys* Print entries where *sys* is the remote system name,

–u*user* Print entries for user *user*.

The intersection of lines satisfying the two options is output. A null *sys* or *user* means all system names or users respectively.

Debugging options available are:

–x*num* *num* is a level number between 1 and 9; higher numbers give more debug output.

–n*secs* Time out lock on log file if older than *secs* seconds.

## 21.6. UUNAME — Systems that uucp can access

`uuname` lists the `uucp` names of systems that can be accessed using `uucp`. There are several options:

–l Displays the local system name.

–v Verbose. Displays a description of each known system.

## 21.7. 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:

–d*dir* 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.

–n*hours* Change the aging time from 72 hours to *hours* hours.

–p*pre* Examine files with prefix *pre* for deletion. Up to 10 of these options may be specified.

–x*num* This is the level of debugging output desired.

## 21.8. Security

CAUTION **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 uuxqt program reads a file containing a list of commands that it will execute. A PATH shell statement is pre-pended to the command line as specified in the uuxqt program. The path is set to /bin:/usr/bin.
- 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, uuxqt, uulog, and uuclean should be owned by the uucp administrative login, have the setuid bit set, and have only execute permissions.

## 21.9. 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 you 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 Uusg for the usg 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.

### Files Required for Execution

Six files are required for execution. They should reside in the /etc/uucp 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-devices Contains call unit information.

L-dialcodes
Contains dialcode abbreviations.

USERFILE Contains user accessibility and constraint information.

L.sys Contains information about the systems that local uucp programs can call.

L.cmds Contains commands that *uuxqt* can execute.

### 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
```

*type* 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.

*line* The device for the line (for example, cul0 if using a DN11, otherwise it is the same as the call-unit field).

*call-unit* The automatic call unit associated with *line* (for example, cua0). Hardwired lines have a number "0" in this field.

*speed* 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 devicel /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 — Dial Code Abbreviations File

L-dialcodes contains the dialcode abbreviations used in the L .sys file (for example, py, mh, boston). The entry format is:

```
abb  dial-seq
```

*abb* is the abbreviation,

*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.

### 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 fR[path-name] ...
```

*login* The login name for a user or the remote computer.

*sys* The system name for a remote computer.

*c* The optional *call-back required* flag,

*path-name* is a pathname 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 pathnames 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 pathnames 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 /spool. If it is system m, it can send files from path /usr/xyz as well as /usr/spool.

The lines:

```
root,  /
,   /usr
```

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 *USERFILE* is normally set up as follows:

```
,myname /
, /usr/spool/uucppublic
, /var/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.

### 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.

You can continue long lines with a backslash (\) as the last character on the line.

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 comma. 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.

CAUTION **Do not put a blank line in your `L.sys` file. `uucp` will not evaluate anything past the blank line. Also note that the maximum length of an `L.sys` entry is 300 characters. If this is exceeded, `uucico` may core dump.**

Also note that there is a 19-field limit on `L.sys` lines.

### `L.cmds`

`L.cmds` contains a list of commands that `uuxqt` 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`, `uusend`, and `lpr`.

### Device Types

The currently supported device types are:

| Type Code | Device |
|---|---|
| ACU | DEC DN11 |
| ACUDN11 | DEC DN11 |
| ACUDF02 | DEC DF02 (300 baud only) |
| ACUDF03 | DEC DF03 (1200 baud only) |
| ACUVENTEL | Ven-Tel MD212 Plus |
| ACUHAYES | Hayes Smartmodem 1200 |
| DIR | Hardwired Line. |

The DN11 is only available on DEC PDP-11 and VAX systems. The DEC DF02, DF03, and Ven-Tel 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/tty`*x*) should be renamed (to `/dev/cua0`, for instance) to emphasize that the line is for dialout only and to prevent accidentally starting a login process for that line. See Chapter 11 for information about attaching modems.

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  cua0  cua0  1200
ACUDF03wats  cua0  cua0  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.

## 21.10. 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.

### SQFILE — Sequence Check File

*SQFILE* is set up in the `/etc/uucp` 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 recent 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.

### TM — Temporary Data Files

These files are created in the `/usr/spool/uucp` 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 (automatically?) moved or copied to the requested destination. If processing terminates prematurely the file remains in the spool directory. You can use the `uuclean` command to periodically remove the leftover files. For example,

```
/usr/lib/uucp/uuclean -pTM
```

removes all `TM` files older than three days.

### 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*.

### 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.

### LCK — Lock Files

Lock files are created for each device in use (for example, 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.

### 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
/usr/lib/uucp/uucico   -r1  -sinter
/usr/lib/uucp/uucico   -r1
```

The `-r1` 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 cannot be accomplished for reasons like bad phone number, login changes, and so on. A shell file containing commands like:

```
/usr/lib/uucp/uuclean   -pTM -pC. -pD.
/usr/lib/uucp/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 ./LOGFILE   /usr/sppol/uucp/o.LOGFILE
rm /usr/spool/uucp/LOGFILE
```

can be used.

The shell files in `/etc/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.

### Login Entry

You should set up two or more logins for `uucp`. One should be an administrative login, that is, for 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 `/usr/lib/uucp/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.

### File Modes

The programs `uucp`, `uux`, `uucico`, `uulog`, `uuclean`, and `uuxqt` should be owned by the `uucp` administrative login with the setuid bit set and only execute permissions (mode 04111). The `L.sys`, `SQFILE`, and the `USERFILE`, which are put in the `/etc/uucp` directory should be owned by the `uucp` administrative login and set with mode 0400. The mode of `/usr/spool/uucp` should be 0755. The mode of `/usr/spool/uucp/.XQTDIR` should be 0777. The `L-dialcodes` and the `L-devices` files should have mode 0444.

# 22

# Name Server Operations

# 22

# Name Server Operations

This chapter is based on the *Sun Release 4.0 Name Server Operations Guide*, published by the Regents of the University of California at Berkeley.

## 22.1. Introduction

The `/usr/etc/in.named` program implements the DARPA Internet name server for the SunOS operating system. This program is taken from the 4.3BSD from Berkeley, where it is sometimes called the Berkeley Internet Name Domain (BIND) Server. A name server is a network service that enables clients to name resources or objects and share this information with other objects in the network. This in effect is a distributed data base system for objects in a computer network. The system administrator can configure the system to use *in.named* in addition to the Yellow Pages *hosts.byname* and *hosts.byaddr* maps, for names outside of the current domain. Direct access to the name server is also available through an additional library.

## 22.2. Building A System with a Name Server

The domain name system is comprised of two parts. One is the client interface called the *resolver*, which consists of a group of routines that reside in the library `/usr/lib/libresolv.a`. Second is the actual server called *in.named.* This is a daemon that runs in the background and services queries on a given network port. The standard port for UDP and TCP is specified in `/etc/services`.

### Resolver Routines in libresolv.a

Linking with the resolver library (through the use of the *-lresolv*) to use the name server changes the way `gethostbyname` (3N), `gethostbyaddr` (3N), and `sethostent` (3N) do their functions. The name server renders `gethostent` (3N) obsolete, since it has no concept of a next line in the database. These library calls are built with the resolver routines needed to query the name server.

The *resolver* is comprised of a few routines that build query packets and exchange them with the name server. Normally only the Yellow Pages server needs to be linked directly with the resolver library, and other programs use the normal Yellow Pages functions to access names. However, a version of `/usr/lib/sendmail` directly linked with the resolver is also provided (as `/usr/lib/sendmail.mx`) since mail delivery involves more information than simple name to address mapping, such as access to Mail Exchanger (MX) information.

## 22.3. The Name Service

The basic function of the name server is to provide information about network objects by answering queries. The specifications for this name server are defined in RFC882, RFC883, RFC973 and RFC974. It is also recommeded that you read the related manual pages, `named(8)`, `resolver(3)`, and `resolver(5)`.

The advantage of using a name server over the host table lookup for host name resolution is to avoid the need for a single centralized clearing house for all names. The authority for this information can be delegated to the different organizations on the network responsible for it.

The host table lookup routines require that the master file for the entire network be maintained at a central location by a few people. This works fine for small networks where there are only a few machines and the different organizations responsible for them cooperate. But this does not work well for large networks where machines cross organizational boundaries.

With the name server, the network can be broken into a hierarchy of domains. The name space is organized as a tree according to organizational or administrative boundaries. Each node, called a *domain*, is given a label, and the name of the domain is the concatenation of all the labels of the domains from the root to the current domain, listed from right to left separated by dots. A label need only be unique within its domain. The whole space is partitioned into several areas called *zones*, each starting at a domain and extending down to the leaf domains or to domains where other zones start. Zones usually represent administrative boundaries. An example of a host address for a host at Podunk University would look as follows:

*monet . Podunk . EDU*

The top level domain for educational organizations is EDU; Podunk is a subdomain of EDU and monet is the name of the host.

## 22.4. Types of Servers

There are three types of servers, Master, Caching and Remote.

### Master Servers

A Master Server for a domain is the authority for that domain. This server maintains all the data corresponding to its domain. Each domain should have at least two master servers, a primary master and some secondary masters to provide backup service if the primary is unavailable or overloaded. A server may be a master for multiple domains, being primary for some domains and secondary for others.

Primary

A Primary Master Server is a server that loads its data from a file on disk. This server may also delegate authority to other servers in its domain.

Secondary

A Secondary Master Server is a server that is delegated authority and receives its data for a domain from a primary master server. At boot time, the secondary server requests all the data for the given zone from the primary master server. This server then periodically checks with the primary server to see if it needs to update its data.

### Caching Only Server

All servers are caching servers. This means that the server caches the information that it receives for use until the data expires. A *Caching Only Server* is a server that is not authoritative for any domain. This server services queries and asks other servers, who have the authority, for the information needed. All servers keep data in their cache until the data expires, based on a time to live field attached to the data when it is received from another server.

### Remote Server

A Remote Server is an option given to people who would like to use a name server on their workstation or on a machine that has a limited amount of memory and CPU cycles. With this option you can run all of the networking programs that use the name server without the name server running on the local machine. All of the queries are serviced by a name server that is running on another machine on the network.

## 22.5. Setting up Your Own Domain

When setting up a domain that is going to be on a public network the site administrator should contact the organization in charge of the network and request the appropriate domain registration form. An organization that belongs to multiple networks (such as *CSNET*, *DARPA Internet* and *UUCP*) should register with only one network.

The contacts are as follows:

### DARPA Internet

Sites that are already on the DARPA Internet and need information on setting up a domain should contact HOSTMASTER@NIC.SRI.COM. You may also want to be placed on the BIND mailing list, which is a mail group for people on the DARPA Internet running BIND. The group discusses future design decisions, operational problems, and other related topic. The address to request being placed on this mailing list is:

*bind-request@ucbarpa.Podunk.EDU.*

### CSNET

A *CSNET* member organization that has not registered its domain name should contact the *CSNET* Coordination and Information Center (*CIC*) for an application and information about setting up a domain.

An organization that already has a registered domain name should keep the *CIC* informed about how it would like its mail routed. In general, the *CSNET* relay will prefer to send mail via *CSNET* (as opposed to *UUCP* or the *Internet*) if possible. For an organization on multiple networks, this may not always be the preferred behavior. The *CIC* can be reached via electronic mail at *cic@sh.cs.net*, or by phone at (617) 497-2777.

### BITNET

If you are on the BITNET and need to set up a domain, contact INFO@BITNIC.

## 22.6. Files

The name server uses several files to load its data base. This section covers the files and their formats needed for `named`.

### Boot File

This is the file that is first read when `named` starts up. This tells the server what type of server it is, which zones it has authority over and where to get its initial data. The default location for this file is `/etc/named.boot`. However this can be changed by setting the `BOOTFILE` variable when you compile `named` or by specifying the location on the command line when `named` is started up.

#### Domain

The line in the boot file that designates the default domain for the server looks as follows:

```
domain    Podunk.Edu
```

The name server uses this information when it receives a query for a name without a "." When it receives one of these queries, it appends the name in the second field to the query name.

#### Sortlist

The line in the boot file that designates a preference order for sorting address looks as follows:

```
sortlist          10.0.0.0    128.45.0.0
```

This line is only needed if you want to control the order that addresses are returned from the name server. This is useful in cases where you have a point-to-point link to another network and you what the address using the link to take preference over going over another network.

#### Primary Master

The line in the boot file that designates the server as a primary server for a zone looks as follows:

```
primary    Podunk.Edu    /etc/puhosts
```

The first field specifies that the server is a primary one for the zone stated in the second field. The third field is the name of the file from which the data is read.

#### Secondary Master

The line for a secondary server is similar to the primary except for the word secondary and the third field.

```
secondary  Podunk.Edu  128.32.10  128.32.0.4
```

The first field specifies that the server is a secondary master server for the zone stated in the second field. The rest of the line, lists the network addresses for the name servers that are primary for the zone. The secondary server gets its data across the network from the listed servers. Each server is tried in the order listed until it successfully receives the data from a listed server.

#### Caching Only Server

You do not need a special line to designate that a server is a caching server. What denotes a caching only server is the absence of authority lines, such as *secondary* or *primary* in the boot file.

All servers should have a line as follows in the boot file to prime the name servers cache:

```
cache     .     /etc/named.ca
```

For information on cache file see section on *Cache Initialization.*

### Remote Server

To set up a host that will use a remote server instead of a local server to answer queries, the file `/etc/resolv.conf` needs to be created. This file designates the name servers on the network that should be sent queries. It is not advisable to create this file if you have a local server running. If this file exists it is read almost every time `gethostbyname ()` or `gethostbyaddr()` is called.

### Cache Initialization

The name server needs to know the server that is the authoritative name server for the network. To do this we have to prime the name server's cache with the address of these higher authorities. This file should only contain the information pertaining to the root authoritative servers for the network. The location of this file is specified in the boot file. This file uses the Standard Resource Record Format covered further on in this paper.

### Domain Data Files

There are three standard files for specifying the data for a domain. These are `named.local`, `hosts` and *host.rev*. These files use the Standard Resource Record Format covered later in this paper.

#### named.local

This file specifies the address for the local loopback interface, better known as *localhost* with the network address 127.0.0.1. The location of this file is specified in the boot file.

#### hosts

This file contains all the data about the machines in this zone. The location of this file is specified in the boot file.

#### hosts.rev

This file specifies the IN-ADDR.ARPA domain. This is a special domain for allowing address to name mapping. As internet host addresses do not fall within domain boundaries, this special domain was formed to allow inverse mapping. The IN-ADDR.ARPA domain has four labels preceding it. These labels correspond to the 4 octets of an Internet address. All four octets must be specified even if an octet is zero. The Internet address 128.32.0.4 is located in the domain 4. .32.128.IN-ADDR.ARPA. This reversal of the address is awkward to read but allows for the natural grouping of hosts in a network.

### Standard Resource Record Format

The records in the name server data files are called resource records. The Standard Resource Record Format (RR) is specified in RFC882 and RFC973. The following is a general description of these records:

*{name}* *{ttl}* *addr-class* *Record Type* *Record Specific data*

Resource records have a standard format shown above. The first field is always the name of the domain record. For some RR's the name may be left blank; in that case it takes on the name of the previous RR. The second field is an optional time to live field. This specifies how long this data will be stored in the data base. By leaving this field blank the default time to live is specified in the *Start Of Authority* resource record (see below). The third field is the address class; there are currently two classes: *IN* for internet addresses and *ANY* for all address classes. The fourth field states the type of the resource record. The fields after that are dependent on the type of the RR. Case is preserved in names and data fields when loaded into the name server. All comparisons and lookups in the name server data base are case insensitive.

The following characters have special meanings:

. A free standing dot in the name field refers to the current domain.

@ A free standing @ in the name field denotes the current origin.

.. Two free standing dots represent the null domain name of the root when used in the name field.

X Where X is any character other than a digit (0-9), quotes that character so that its special meaning does not apply. For example, '\.' can be used to place a dot character in a label.

.if 0. Where each D is a digit, is the octet corresponding to the decimal number described by DDD. The resulting octet is assumed to be text and is not checked for special meaning.

( ) Parentheses are used to group data that crosses a line. In effect, line terminations are not recognized within parentheses.

; Semicolon starts a comment; the remainder of the line is ignored.

* An asterisk signifies wildcarding.

Most resource records will have the current origin appended to names if they are not terminated by a "." This is useful for appending the current domain name to the data, such as machine names, but may cause problems where you do not want this to happen. A good rule of thumb is that, if the name is not in of the domain for which you are creating the data file, end the name with a "."

### $INCLUDE

An include line begins with $INCLUDE, starting in column 1, and is followed by a file name. This feature is particularly useful for separating different types of data into multiple files. An example would be:

```
$INCLUDE /usr/named/data/mailboxs
```

The line would be interpreted as a request to load the file `/usr/named/data/mailboxes`. The $INCLUDE command does not cause data to be loaded into a different zone or tree. This is simply a way to allow data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

### $ORIGIN

The origin is a way of changing the origin in a data file. The line starts in column 1, and is followed by a domain origin. This is useful for putting more then one domain in a data file.

### SOA - Start Of Authority

```
name   {ttl}   addr-class   SOA        Origin                 Person in charge
@              IN           SOA        pusun.Podunk.Edu.      kjd.pusun.Podunk.Edu. (
                            1.1        ; Serial
                            3600       ; Refresh
                            300        ; Retry
                            3600000    ; Expire
                            3600 )     ; Minimum
```

The *Start of Authority, SOA,* record designates the start of a zone. The name is the name of the zone. Origin is the name of the host on which this data file resides. Person in charge is the mailing address for the person responsible for the name server. The serial number is the version number of this data file, this number should be incremented whenever a change is made to the data. The name server cannot handle numbers over 9999 after the decimal point. The refresh indicates how often, in seconds, a secondary name servers is to check with the primary name server to see if an update is needed. The retry indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh. Expire is the upper limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh. Minimum is the default number of seconds to be used for the time to live field on resource records. There should only be one *SOA* record per zone.

### NS - Name Server

```
{name}   {ttl}   addr-class   NS   Name servers name
                 IN           NS   puarpa.Podunk.Edu.
```

The *Name Server* record, *NS*, lists a name server responsible for a given domain. The first name field lists the domain that is serviced by the listed name server. There should be one *NS* record for each Primary Master server for the domain.

### A - Address

```
{name}   {ttl}   addr-class   A   address
puarpa           IN           A   128.32.0.4
                 IN           A   10.0.0.78
```

The *Address* record, *A*, lists the address for a given machine. The name field is the machine name and the address is the network address. There should be one *A* record for each address of the machine.

### HINFO - Host Information

```
{name}   {ttl}   addr-class   HINFO   Hardware     OS
                 ANY          HINFO   Sun-3/280    UNIX
```

*Host Information* resource record, *HINFO*, is for host specific data. This lists the hardware and operating system that are running at the listed host. It should be noted that only a single space separates the hardware info and the operating system info. If you want to include a space in the machine name you must quote the name. Host information is not specific to any address class, so *ANY* may be used for the address class. There should be one *HINFO* record for each host.

### WKS - Well Known Services

| *{name}* | *{ttl}* | *addr-class* | *WKS* | *address* | *protocol* | *list of services* |
|---|---|---|---|---|---|---|
| | | IN | WKS | 128.32.0.10 | UDP | who route timed domain |
| | | IN | WKS | 128.32.0.10 | TCP | ( echo telnet<br>discard sunrpc sftp<br>uucp-path systat daytime<br>netstat qotd nntp<br>link chargen ftp<br>auth time whois mtp<br>pop rje finger smtp<br>supdup hostnames<br>domain<br>nameserver ) |

The *Well Known Services* record, *WKS*, describes the well known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in `/etc/services`. There should be only one *WKS* record per protocol per address.

### CNAME - Canonical Name

| *aliases* | *{ttl}* | *addr-class* | *CNAME* | *Canonical name* |
|---|---|---|---|---|
| pumonet | | IN | CNAME | monet |

*Canonical Name* resource record, *CNAME*, specifies an alias for a canonical name. An alias should be unique and all other resource records should be associated with the canonical name and not with the alias. Do not create an alias and then use it in other resource records.

### PTR - Domain Name Pointer

| *name* | *{ttl}* | *addr-class* | *PTR* | *real name* |
|---|---|---|---|---|
| 7.0 | | IN | PTR | `monet.Podunk.Edu.` |

A *Domain Name Pointer* record, *PTR*, allows special names to point to some other location in the domain. The above example of a *PTR* record is used in setting up reverse pointers for the special *IN-ADDR.ARPA* domain. This line is from the example *hosts.rev* file. *PTR* names should be unique to the zone.

### MB - Mailbox

| *name* | *{ttl}* | *addr-class* | *MB* | *Machine* |
|---|---|---|---|---|
| miriam | | IN | MB | vineyd.Sun.COM. |

*MB* is the *Mailbox* record. This lists the machine where a user wants to receive mail. The name field is the users login; the machine field denotes the machine to which mail is to be delivered. Mail Box names should be unique to the zone. **Note that the *MB, MR, MINFO,* and *MG* records are not used by SunOS at the present time (release 4.0).**

### MR - Mail Rename Name

| *name* | *{ttl}* | *addr-class* | *MR* | *corresponding MB* |
|---|---|---|---|---|
| Postmistress | | IN | MR | miriam |

*Main Rename, MR,* can be used to list aliases for a user. The name field lists the alias for the name listed in the fourth field, which should have a corresponding *MB* record.

MINFO - Mailbox Information

| *name* | *{ttl}* | *addr-class* | *MINFO* | *requests* | *maintainer* |
|---|---|---|---|---|---|
| BIND | | IN | MINFO | BIND-REQUEST | kjd.Podunk.Edu. |

*Mail Information* record, *MINFO*, creates a mail group for a mailing list. This resource record is usually associated with a mail group *Mail Group*, but may be used with a *Mail Box* record. The *name* specifies the name of the mailbox. The *requests* field is where mail such as requests to be added to a mail group should be sent. The *maintainer* is a mailbox that should receive error messages. This is particularly appropriate for mailing lists when errors in members names should be reported to a person other than the sender.

MG - Mail Group Member

| *{mail group name}* | *{ttl}* | *addr-class* | *MG* | *member name* |
|---|---|---|---|---|
| | | IN | MG | Bloom |

*Mail Group, MG* lists members of a mail group.

An example for setting up a mailing list is as follows:

```
Bind    IN   MINFO   Bind-Request            kjd.Podunk.Edu.
        IN   MG      Ralph.Podunk.Edu.
        IN   MG      Zhou.Podunk.Edu.
        IN   MG      Painter.Podunk.Edu.
        IN   MG      Riggle.Podunk.Edu.
        IN   MG      Terry.pa.Xerox.Com.
```

MX - Mail Exchanger

| *name* | *{ttl}* | *addr-class* | *MX* | *preference value* | *mailer exchanger* |
|---|---|---|---|---|---|
| Munnari.OZ.AU. | | IN | MX | 0 | Seismo.CSS.GOV. |
| *.IL. | | IN | MX | 0 | RELAY.CS.NET. |

*Main Exchanger* records, *MX*, are used to specify a machine that knows how to deliver mail to a machine that is not directly connected to the network. In the first example, above, Seismo.CSS.GOV. is a mail gateway that knows how to deliver mail to Munnari.OZ.AU. but other machines on the network can not deliver mail directly to Munnari. These two machines may have a private connection or use a different transport medium. The preference value is the order that a mailer should follow when there is more then one way to deliver mail to a single machine. See RFC974 for more detailed information.

Wildcard names containing the character "*" may be used for mail routing with *MX* records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a relay. Second example, above, all mail to hosts in the domain IL is routed through RELAY.CS.NET. This is done by creating a wildcard resource record, which states that *.IL has an *MX* of RELAY.CS.NET.

## Sample Files

The following section contains sample files for the name server. This covers example boot files for the different types of servers and example domain data base files.

## Boot File

### Primary Master Server

```
;
; Boot file for Primary Master Name Server
;

; type      domain                  source file or host
;
domain      Podunk.Edu
primary     Podunk.Edu              /etc/puhosts
cache       .                       /etc/named.ca
primary     32.128.in-addr.arpa     /etc/puhosts.rev
primary     0.0.127.in-addr.arpa    /etc/named.local
```

### Secondary Master Server

```
;
; Boot file for Primary Master Name Server
;

; type        domain                  source file or host
;
domain        Podunk.Edu
secondary     Podunk.Edu              128.32.0.4 128.32.0.10 128.32.136.22
cache         .                       /etc/named.ca
secondary     32.128.in-addr.arpa     128.32.0.4 128.32.0.10 128.32.136.22
primary       0.0.127.in-addr.arpa    /etc/named.local
```

### Caching Only Server

```
;
; Boot file for Primary Master Name Server
;

; type      domain                  source file or host
;
domain      Podunk.Edu
cache       .                       /etc/named.ca
primary     0.0.127.in-addr.arpa    /etc/named.local
```

## Remote Server

### /etc/resolv.conf

```
domain Podunk.Edu
nameserver 128.32.0.4
nameserver 128.32.0.10
```

### named.ca

```
; ; Initial cache data for root domain servers.  ;
.                       99999999   IN   NS    USC-ISIC.ARPA.
                        99999999   IN   NS    BRL-AOS.ARPA.
                        99999999   IN   NS    SRI-NIC.ARPA.
;  Prep the cache (hotwire the addresses).
SRI-NIC.ARPA.           99999999   IN   A     10.0.0.51
USC-ISIC.ARPA.          99999999   IN   A     10.0.0.52
BRL-AOS.ARPA.           99999999   IN   A     128.20.1.2
BRL-AOS.ARPA.           99999999   IN   A     192.5.22.82
```

### named.local

```
@      IN     SOA     pusun.Podunk.Edu. kjd.pusun.Podunk.Edu. (
                      1       ; Serial
                      3600    ; Refresh
                      300     ; Retry
                      3600000 ; Expire
                      3600 )  ; Minimum
       IN     NS      pusun.Podunk.Edu.
1      IN     PTR     localhost.
```

### Hosts

```
;;  @(#)ucb-hosts  1.1  (berkeley)  86/02/05 ;
@             IN    SOA     pusun.Podunk.Edu. kjd.monet.Podunk.Edu. (
                            1.1      ; Serial
                            3600     ; Refresh
                            300      ; Retry
                            3600000  ; Expire
                            3600 )   ; Minimum
              IN    NS      puarpa.Podunk.Edu.
              IN    NS      pusun.Podunk.Edu.
localhost     IN    A       127.1
puarpa        IN    A       128.32.4
              IN    A       10.0.0.78
              ANY   HINFO   Sun-3/280 UNIX
arpa          IN    CNAME   puarpa
ernie         IN    A       128.32.6
              ANY   HINFO   Sun-3/280 UNIX
puernie       IN    CNAME   ernie
monet         IN    A       128.32.7
              IN    A       128.32.130.6
              ANY   HINFO   Sun-3/50 UNIX
pumonet       IN    CNAME   monet
pusun         IN    A       10.2.0.78
              IN    A       128.32.10
              ANY   HINFO   Sun-3/50 UNIX
              IN    WKS     128.32.0.10 UDP syslog route timed domain
              IN    WKS     128.32.0.10 TCP ( echo telnet
                            discard sunrpc sftp
                            uucp-path systat daytime
                            netstat qotd nntp
                            link chargen ftp
                            auth time whois mtp
                            pop rje finger smtp
                            supdup hostnames
                            domain
                            nameserver )
sun           IN    CNAME   pusun
toybox        IN    A       128.32.131.119
              ANY   HINFO   Pro350 RT11
toybox        IN    MX      0  monet.Podunk.Edu
miriam        ANY   MB      vineyd.Sun.COM.
postmistress  ANY   MR      Miriam
Bind          ANY   MINFO   Bind-Request kjd . Podunk . Edu .
              ANY   MG      Ralph . Podunk . Edu .
              ANY   MG      Zhou . Podunk . Edu .
              ANY   MG      Painter . Podunk . Edu .
              ANY   MG      Riggle . Podunk . Edu .
              ANY   MG      Terry . pa . Xerox . Com .
```

### host.rev

```
;;   @(#)ucb-hosts.rev   1.1   (Berkeley)   86/02/05 ;
@        IN    SOA    pusun.Podunk.Edu. kjd.monet.Podunk.Edu. (
                      1.1     ; Serial
                      3600    ; Refresh
                      300     ; Retry
                      3600000 ; Expire
                      3600 )  ; Minimum
         IN    NS     puarpa.Podunk.Edu.
         IN    NS     pusun.Podunk.Edu.
4.0      IN    PTR    puarpa.Podunk.Edu.
6.0      IN    PTR    ernie.Podunk.Edu.
7.0      IN    PTR    monet.Podunk.Edu.
10.0     IN    PTR    pusun.Podunk.Edu.
6.130    IN    PTR    monet.Podunk.Edu.
```

## 22.7. Domain Management

This section contains information for starting, controlling and debugging *named.*

### /etc/rc.local

The hostname should be set to the full domain style name in /etc/rc.local using *hostname (1).* The following entry should be added to /etc/rc.local to start up *named* at system boot time:

```
if [ -f /etc/named ]; then
    /etc/named [options] & echo -n ' named'   >/dev/console
fi
```

This usually directly follows the lines that start syslogd. **Do Not** attempt to run *named* from *inetd.* This will continuously restart the name server and defeat the purpose of having a cache.

### /etc/named.pid

When named is successfully started up it writes its process id into the file /etc/named.pid. This is useful to programs that want to send signals to *named.* The name of this file may be changed by defining PIDFILE to the new name when compiling named.

### /etc/hosts

The gethostbyname () library call can detect if *named* is running. If it is determined that *named* is not running it will look in /etc/hosts to resolve an address. This option was added to allow ifconfg (8C) to configure the machines local interfaces and to enable a system manager to access the network while the system is in single user mode. It is advisable to put the local machines interface addresses and a couple of machine names and address in /etc/hosts so the system manager can rcp files from another machine when the system is in single user mode. The format of /etc/host has not changed. See hosts (5) for more information. Since the process of reading /etc/hosts is slow, it is not advised to use this option when the system is in multi user mode.

**Signals**

There are several signals that can be sent to the named process to have it do tasks without restarting the process.

Reload

SIGHUP -
Causes named to read named.boot and reload the database. All previously cached data is lost. This is useful when you have made a change to a data file and you want *named*'s internal database to reflect the change.

Debugging

When named is running incorrectly, look first in /usr/adm/messages and check for any messages logged by syslog. Next send it a signal to see what is happening.

SIGINT -
Dumps the current data base and cache to /var/ tmp/ named_dump . db This should give you an indication to whether the data base was loaded correctly. The name of the dump file may be changed by defining *DUMPFILE* to the new name when compiling named.

*Note:* the following two signals only work when named is built with *DEBUG* defined.

SIGUSR1 -
Turns on debugging. Each following USR1 increments the debug level. The output goes to /var/tmp/named.run The name of this debug file may be changed by defining DEBUGFILE to the new name before compiling named.

SIGUSR2 -
Turns off debugging completely.

For more detailed debugging, define DEBUG when compiling the resolver routines into */lib/libc.a.*

## ACKNOWLEDGEMENTS

Many thanks to the users at U.C. Berkeley for falling into many of the holes involved with integrating BIND into the system so that others would be spared the trauma. I would also like to extend gratitude to Jim McGinness and Digital Equipment Corporation for permitting me to spend most of my time on this project.

Ralph Campbell, Doug Kingston, Craig Partridge, Smoot Carl-Mitchell, Mike Muuss and everyone else on the DARPA Internet who has contributed to the development of BIND. To the members of the original BIND project, Douglas Terry, Mark Painter, David Riggle and Songnian Zhou.

Anne Hughes, Jim Bloom and Kirk McKusick and the many others who have reviewed this paper giving considerable advice.

This work was sponsored by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 monitored by the Naval Electronics Systems Command under contract No. N00039-84-C-0089. The views and conclusions

contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency, of the US Government, or of Digital Equipment Corporation.

## 22.8. REFERENCES

[Birrell]

Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M.D., *Grapevine: An Exercise in Distributed Computing.* In *Comm. A.C.M. 25,* 4:260-274 April 1982.

[RFC819]

Su, Z. Postel, J., *The Domain Naming Convention for Internet User Applications. Internet Request For Comment 819* Network Information Center, SRI International, Menlo Park, California. August 1982.

[RFC882]

Mockapetris, P., *Domain Names - Concept and Facilities. Internet Request For Comment 882* Network Information Center, SRI International, Menlo Park, California. November 1983.

[RFC883]

Mockapetris, P., *Domain Names - Implementation and Specification. Internet Request For Comment 883* Network Information Center, SRI International, Menlo Park, California. November 1983.

[RFC973]

Mockapetris, P., *Domain System Changes and Observations. Internet Request For Comment 973* Network Information Center, SRI International, Menlo Park, California. February 1986.

[RFC974]

Partridge, C., *Mail Routing and The Domain System. Internet Request For Comment 974* Network Information Center, SRI International, Menlo Park, California. February 1986.

[Terry]

Terry, D. B., Painter, M., Riggle, D. W., and Zhou, S., *The Berkeley Internet Name Domain Server.* Proceedings USENIX Summer Conference, Salt Lake City, Utah. June 1984, pages 23-31.

[Zhou]

Zhou, S., *The Design and Implementation of the Berkeley Internet Name Domain (BIND) Servers.* UCB/CSD 84/177. University of California, Berkeley, Computer Science Division. May 1984.

# A

# Timezones

# A

# Timezones

| A.1. TIMEZONE NAME: | TIMEZONE AREA: |
|---|---|
| North America: | |
| US/Eastern | Eastern time zone, U.S.A. |
| US/Central | Central time zone, U.S.A. |
| US/Mountain | Mountain time zone, U.S.A. |
| US/Pacific | Pacific time zone, U.S.A. |
| US/Pacific-New | Pacific time zone, U.S.A., with proposed twiddling of Daylight Savings Time near election time in Presidential election years |
| US/Yukon | Yukon time zone, U.S.A. |
| US/East-Indiana | Eastern time zone, U.S.A., no Daylight Savings Time |
| US/Arizona | Mountain time zone, U.S.A., no Daylight Savings Time |
| US/Hawaii | Hawaii |
| Canada/Newfoundland | Newfoundland |
| Canada/Atlantic | Atlantic time zone, Canada |
| Canada/Eastern | Eastern time zone, Canada |
| Canada/Central | Central time zone, Canada |
| Canada/East-Saskatchewan | Central time zone, Canada, no Daylight Savings Time |
| Canada/Mountain | Mountain time zone, Canada |
| Canada/Pacific | Pacific time zone, Canada |
| Canada/Yukon | Yukon time zone, Canada |
| Europe: | |
| GB-Eire | Great Britain and Eire |
| WET | Western European time |
| Iceland | Iceland |
| MET | Middle European time (also known as Central European time) |
| Poland | Poland |
| EET | Eastern European time |

| | |
|---|---|
| Turkey | Turkey |
| W-SU | Western Soviet Union |

Asia (including Australia and New Zealand):

| | |
|---|---|
| PRC | People's Republic of China |
| Korea | Republic of Korea |
| Japan | Japan |
| Singapore | Singapore |
| Hongkong | Hong Kong |
| ROC | Republic of China |

| | |
|---|---|
| Australia/Tasmania | Tasmania, Australia |
| Australia/Queensland | Queensland, Australia |
| Australia/North | Northern Territory |
| Australia/West | Western Australia |
| Australia/South | South Australia |
| Australia/Victoria | Victoria, Australia |
| Australia/NSW | New South Wales, Australia |

| | |
|---|---|
| NZ | New Zealand |

Other (if the locale isn't listed above); none of these have Daylight Savings Time:

| | |
|---|---|
| GMT | Greenwich Mean time |
| GMT-1 | 1 hours west of Greenwich Mean Time |
| GMT-2 | 2 hours west of Greenwich Mean Time |
| GMT-3 | 3 hours west of Greenwich Mean Time |
| GMT-4 | 4 hours west of Greenwich Mean Time |
| GMT-5 | 5 hours west of Greenwich Mean Time |
| GMT-6 | 6 hours west of Greenwich Mean Time |
| GMT-7 | 7 hours west of Greenwich Mean Time |
| GMT-8 | 8 hours west of Greenwich Mean Time |
| GMT-9 | 9 hours west of Greenwich Mean Time |
| GMT-10 | 10 hours west of Greenwich Mean Time |
| GMT-11 | 11 hours west of Greenwich Mean Time |
| GMT-12 | 12 hours west of Greenwich Mean Time |
| GMT+13 | 13 hours east of Greenwich Mean Time |
| GMT+12 | 12 hours east of Greenwich Mean Time |
| GMT+11 | 11 hours east of Greenwich Mean Time |
| GMT+10 | 10 hours east of Greenwich Mean Time |
| GMT+9 | 9 hours east of Greenwich Mean Time |
| GMT+8 | 8 hours east of Greenwich Mean Time |
| GMT+7 | 7 hours east of Greenwich Mean Time |
| GMT+6 | 6 hours east of Greenwich Mean Time |
| GMT+5 | 5 hours east of Greenwich Mean Time |
| GMT+4 | 4 hours east of Greenwich Mean Time |
| GMT+3 | 3 hours east of Greenwich Mean Time |

| | |
|---|---|
| GMT+2 | 2 hours east of Greenwich Mean Time |
| GMT+1 | 1 hours east of Greenwich Mean Time |

# References

[Birrell82] Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M. D., “Grapevine: An Exercise in Distributed Computing.” In *Comm. A.C.M. 25,* 4, April 82.

[Borden79] Borden, S., Gaines, R. S., and Shapiro, N. Z., *The MH Message Handling System: Users' Manual.* R-2367-PAF. Rand Corporation. October 1979.

[Crocker77a] Crocker, D. H., Vittal, J. J., Pogran, K. T., and Henderson, D. A. Jr., *Standard for the Format of ARPA Network Text Messages.* RFC 733, NIC 41952. In [Feinler78]. November 1977.

[Crocker77b] Crocker, D. H., *Framework and Functions of the MS Personal Message System.* R-2134-ARPA, Rand Corporation, Santa Monica, California. 1977.

[Crocker79] Crocker, D. H., Szurkowski, E. S., and Farber, D. J., *An Internetwork Memo Distribution Facility —MMDF.* 6th Data Communication Symposium, Asilomar. November 1979.

[Crocker82] Crocker, D. H., *Standard for the Format of Arpa Internet Text Messages.* RFC 822. Network Information Center, SRI International, Menlo Park, California. August 1982.

[Metcalfe76] Metcalfe, R., and Boggs, D., “Ethernet: Distributed Packet Switching for Local Computer Networks ,” *Communications of the ACM 19,* 7. July 1976.

[Feinler78] Feinler, E., and Postel, J. (eds.), *ARPANET Protocol Handbook.* NIC 7104, Network Information Center, SRI International, Menlo Park, California. 1978.

[NBS80] National Bureau of Standards, *Specification of a Draft Message Format Standard.* Report No. ICST/CBOS 80-2. October 1980.

[Neigus73] Neigus, N., *File Transfer Protocol for the ARPA Network.* RFC 542, NIC 17759. In [Feinler78]. August, 1973.

[Nowitz78a] Nowitz, D. A., and Lesk, M. E., *A Dial-Up Network of* UNIX Systems. Bell Laboratories. In UNIX Programmer's Manual, Seventh Edition, Volume 2. August, 1978.

[Nowitz78b] Nowitz, D. A., *Uucp Implementation Description.* Bell Laboratories. In UNIX Programmer's Manual, Seventh Edition, Volume 2. October, 1978.

[Postel74] Postel, J., and Neigus, N., Revised FTP Reply Codes. RFC 640, NIC 30843. In [Feinler78]. June, 1974.

[Postel77] Postel, J., *Mail Protocol.* NIC 29588. In [Feinler78]. November 1977.

[Postel79a] Postel, J., *Internet Message Protocol.* RFC 753, IEN 85. Network Information Center, SRI International, Menlo Park, California. March 1979.

[Postel79b] Postel, J. B., *An Internetwork Message Structure.* In *Proceedings of the Sixth Data Communications Symposium,* IEEE. New York. November 1979.

[Postel80] Postel, J. B., *A Structured Format for Transmission of Multi-Media Documents.* RFC 767. Network Information Center, SRI International, Menlo Park, California. August 1980.

[Postel82] Postel, J. B., *Simple Mail Transfer Protocol.* RFC821 (obsoleting RFC788). Network Information Center, SRI International, Menlo Park, California. August 1982.

[Schmidt79] Schmidt, E., *An Introduction to the Berkeley Network.* University of California, Berkeley California. 1979.

[Shoens79] Shoens, K., *Mail Reference Manual.* University of California, Berkeley. In UNIX Programmer's Manual, Seventh Edition, Volume 2C. December 1979.

[Sluizer81] Sluizer, S., and Postel, J. B., *Mail Transfer Protocol.* RFC 780. Network Information Center, SRI International, Menlo Park, California. May 1981.

[Solomon81] Solomon, M., Landweber, L., and Neuhengen, D., "The Design of the CSNET Name Server." CS-DN-2, University of Wisconsin, Madison. November 1981.

[Su82] Su, Zaw-Sing, and Postel, Jon, *The Domain Naming Convention for Internet User Applications.* RFC819. Network Information Center, SRI International, Menlo Park, California. August 1982.

[UNIX83] *The* UNIX Programmer's Manual, Seventh Edition, Virtual VAX-11 Version, Volume 1. Bell Laboratories, modified by the University of California, Berkeley, California. March, 1983.

# Index

# Notes