CPL

UTX/32™ Release 2.1 Operations Guide

January 1988

323-005420-100

GOULD
Electronics

History

The *UTX/32 Release 2.1 Operations Guide*, Publication Order Number 323-005420-100, was printed in January 1988.

This document contains the following pages:

Contents

Figures

Tables

1 Introduction

This introductory chapter provides the following information about this document:

- Its scope and purpose
- A summary of its contents
- Reader prerequisites
- Related documentation
- Typographic conventions
- Special messages

1.1 Scope and Purpose of this Guide

This guide is intended to help system administrators and their staff understand, operate, and maintain Gould's UTX/32 ™ operating system.

1.2 Summary of Contents

This guide is divided into 17 chapters, 4 appendixes, a glossary, references, and an index. The first two chapters acquaint readers with this document and other resources available to them. The remaining chapters, organized according to the tasks system administrators typically perform, describe the procedures necessary to operate UTX/32.

Chapter 1	Provides general information about this document
Chapter 2	Describes the distribution and format of the UNIX®-style reference manuals provided with the UTX/32 set
Chapter 3	Explains how to boot, halt, and switch between UTX/32 operating modes
Chapter 4	Explains how to set the system date and use the clock daemon, **cron**, to schedule system activities
Chapter 5	Describes how to format and partition disks and provides information on building file systems
Chapter 6	Describes how to install and manage source code
Chapter 7	Explains how to reconfigure the system
Chapter 8	Describes how to establish user environments
Chapter 9	Explains lineprinter, mail, and networking subsystems

Chapter 10	Describes the accounting system
Chapter 11	Describes how to administer disk quotas
Chapter 12	Describes how to use the auditing system
Chapter 13	Describes site-dependent system files
Chapter 14	Describes how to monitor system performance
Chapter 15	Explains how to perform system backups
Chapter 16	Provides some troubleshooting pointers
Chapter 17	Describes some hardware considerations
Appendix A	Provides disk partition templates and examples
Appendix B	Lists programs that system administrators might find helpful
Appendix C	Provides additional configuration file information
Appendix D	Describes the procedures for creating a tapeboot
Glossary	Defines special terms and acronyms used in this document
References	Provides detailed information on external references
Index	Provides an alphabetical listing of terms with chapter and page references

1.3 Reader Prerequisites

Readers should be familiar with this release of UTX/32 and have access to its documentation. Consult the *UTX/32 Software Release Notes* for release-specific information. See the *UTX/32 Documentation Guide* for an overview of the documentation set.

1.4 Related Documentation and Training

In addition to this *Operations Guide*, a variety of other sources provide information about UTX/32.

1.4.1 The UTX/32 Document Set

The *UTX/32 Operations Guide* belongs to a full set of UTX/32-related documents distributed with this release. These documents have been divided into three basic categories:

- User guides
- Programmer guides
- System Administrator guides

Each category contains the documents that relate most directly to the immediate concerns of that audience.

For example, this *Operations Guide*, along with the *Installation Guide*, *Software Release Notes* and the *System Administrator's Reference Manual* fall into the system administrator category. These documents are most likely to be used on a regular basis by system administrators.

The following documents constitute the full UTX/32 document set:

User Guides

UTX/32 Documentation Guide
UTX/32 Real-Time User's Guide
UTX/32 BSD User's Reference Manual
UTX/32 System V User's Reference Manual
UTX/32 Supplementary Documents: Getting Started
UTX/32 Supplementary Documents: Communications for Users
UTX/32 Supplementary Documents: Text Editors
UTX/32 Supplementary Documents: Document Preparation
UTX/32 Supplementary Documents: Basic Utilities

Programmer Guides

UTX/32 Input/Output Subsystem Guide
UTX/32 Network File System (NFS) Guide
UTX/32 Remote Job Entry (RJE) Guide
UTX/32 Assembler Reference Manual
UTX/32 Fortran 77 Reference Manual
UTX/32 Special Files Reference Manual
UTX/32 BSD Programmer's Reference Manual
UTX/32 System V Programmer's Reference Manual
UTX/32 Supplementary Documents: Programming Tools
UTX/32 Supplementary Documents: System Communications
UTX/32 Supplementary Documents: System Programming
UTX/32 Supplementary Documents: Programming Languages

System Administrator Guides

UTX/32 Software Release Notes
UTX/32 Installation Guide
UTX/32 Operations Guide
UTX/32 System Administrator's Reference Manual
UTX/32 Supplementary Documents: System Management

1.4.2 UTX/32 Training

Gould also offers a variety of training programs focusing on installation, operation, and preventative and corrective maintenance tasks associated with UTX/32. See the *UTX/32 Documentation Guide* or contact your Customer Service Representative for more information.

1.5 Typographic Conventions

The typographic conventions for this document are described below.

Prompts

The following prompts are used in this document:

`//`	Panel mode prompt
`#`	Superuser prompt
`%`	C shell prompt

Nonprinting and control characters

Nonprinting characters obtained by striking special keys are displayed within angle brackets. For example, <DEL> indicates the delete key, <CR> a carriage return.

In this guide, a <CR> is assumed at the end of every command line unless otherwise stated. The <CR> is displayed only if nothing else is entered on the line or if the sequence of keystrokes would otherwise be unclear.

Control characters are represented using the caret notation. For example, ^D indicates <CTRL>-d. In examples, control characters are shown as echoing on the terminal screen. Whether they echo on your terminal depends on its settings; see *stty*(1).

Boldface

Command and utility names, filenames, pathnames, and words from code are printed in boldface.

Example:

The **nroff** command is used to format text.

`Lineprinter` and **`lineprinter bold`**

Displays of code and user sessions are printed in lineprinter font. In displays of interactive user sessions, text typed by the user is printed in lineprinter bold.

Example:

```
% ls
file1       file2       file3
```

Italics

Variable expressions that must be replaced with a value are printed in italics. Square brackets ([ ]) around an italicized variable expression signify that specifying the value is optional.

Example:

```
% cd [directory]
```

Italics are also used to introduce new terms, for titles of documents or manual pages, and occasionally for emphasis.

Examples:

See *mount*(8) for further information.

The first tape, called the *boot tape*, contains three boot programs.

Ellipses
Vertical or horizontal ellipses (. . .) indicate that information has been omitted.

Example:

```
% rsh fang
    .
    .
    .
% logout
```

Blank pages
Since each major section of the document begins on a right-hand (odd-numbered) page, blank left-hand (even-numbered) pages occasionally precede new sections. You can be assured that such a page is intended to be blank if the preceding page has a double page number, such as 4-5/4-6.

1.6 Special messages

Interspersed throughout this document are special pieces of information that serve to highlight or augment instructions. These are of two kinds:

WARNINGS	emphasize procedures that are essential to proper system operation.
NOTES	provide useful information that is not critical to the system's operation.

2 Reference Manual Information

2.1 UTX/32 Reference Manuals

The reference manual pages (sometimes called *manpages*) provide detailed information on commands needed to administer, operate, monitor, and use the UTX/32 operating system. Each manual page includes:

- Command name
- Syntax
- Options available
- A description of the command
- Examples of how the command can be used
- Supplementary information about the use of the command
- References to related information

UTX/32 includes a full set of BSD and System V manual pages on line which can be accessed using the **man** command (see *man*(1)). Hardcopy versions of the manual pages are located in the following reference manuals:

UTX/32 BSD User's Reference Manual
UTX/32 BSD Programmer's Reference Manual
UTX/32 System V User's Reference Manual
UTX/32 System V Programmer's Reference Manual
UTX/32 Special Files Reference Manual
UTX/32 System Administrator's Reference Manual

Within each reference manual, the manual pages are in alphabetical order by command, function, or filename. This guide refers most frequently to manual pages found in the *UTX/32 System Administrator's Reference Manual.*

2.1.1 BSD Manual Page Distribution

Table 2-1 summarizes the content of BSD manual page information by section number and shows the correspondence between the UTX/32 and original BSD section numbers.

Table 2-1
Numbering of Manual Page Sections

UTX Section #	Original BSD Section #	Contents
1	1	Manual pages for all user-level commands and other utilities.
2	2	Manual pages for all system calls, which are the routines that provide basic OS primitives.
3	3	Manual pages for all library routines
4	5	Manual pages for all system files. These files (e.g., **/etc/passwd**) provide information that UTX/32 needs for normal operation.
5	7	Miscellaneous manual pages, providing information on such topics as **nroff** macro packages and the ASCII character set.
6	6	Manual pages for all games.
7	4	Manual pages for all special files (device drivers and pseudo-devices).
8	8	Manual pages for all administrator-level commands, daemons, and other system programs.

UTX/32 manual pages are distributed in both tape and hardcopy formats. The following subsections discuss these two modes of distribution.

Tape Distribution

The UTX/32 binary distribution tape, not the source tape, contains the source for the BSD manual pages. The eight sections of BSD manual pages and introduction sections are located in the following subdirectories:

/usr/man/man0	Introductory text files
/usr/man/man1 and **/usr/man/man6**	User manual pages Game manual pages
/usr/man/man[2-5]	Programmer manual pages
/usr/man/man7	Special files manual pages
/usr/man/man8	System administrator manual pages

Gould-specific manual pages are also included in this set. Gould-specific manual pages are formatted, viewed on-line, and distributed in the same way as BSD manual pages.

Hardcopy Distribution

The following table shows how sections from the UTX/32 manual page set are incorporated into the hardcopy manuals. Note that the section numbers are those of the UTX/32 distribution. See Table 2-1 for the correspondence between the UTX/32 section numbers and BSD manuals.

Table 2-2
BSD and Gould Sections in the UTX/32 Manual Page Set

UTX Section	Document Title
1 and 6	*UTX/32 BSD User's Reference Manual*
2, 3, 4, and 5	*UTX/32 BSD Programmer's Reference Manual*
7	*UTX/32 Special Files Reference Manual*
8	*UTX/32 System Administrator's Reference Manual*

2.1.2 BSD Manual Page Formatting

An unformatted BSD manual page contains **nroff** formatting information (see *man*(5)) and, in general, is not suitable for printing on a lineprinter or viewing on a terminal screen.

A formatted manual page is generated from the unformatted version by using **man** or **catman** and is deposited in a **/usr/man/cat***n* directory. The formatted version is suitable for printing on a lineprinter or viewing on a terminal screen.

The UTX/32 manual page macro sets are modified versions of the BSD manual page macro sets. In the **/usr/lib/tmac** directory, the file **tmac.an.new** formats BSD manual pages and the file **tmac.gould** formats Gould-specific manual pages. The file **tmac.an** will use either **tmac.an.new** or **tmac.gould** as required by the manual page input files. This means that the command **nroff –man** will work correctly for any manual page input file.

Formatting with man

In the BSD environment, the **man** command enables you to view the manual page that is specified as the argument to the command. If the manual page is not already formatted, **man** first formats it and then enables you to view it. For example, entering the command

```
% man apropos
```

formats the *apropos*(1) manual page, if it is not already formatted, and displays it on a terminal screen. If *apropos*(1) is already formatted, **man** simply sends it to the terminal screen.

man executes the command **nroff –man**, which in turn uses the control file **/usr/lib/tmac/tmac.an** to obtain the appropriate macro sets in the files **tmac.an.new** or **tmac.gould**. See *man*(1) for more information.

Formatting with catman

The **catman** command enables a system administrator to format all of the manual pages in the BSD **man***n* directories at once to make access more convenient for users. **catman** creates a formatted version of each manual page whose formatted version is missing or out of date. The output files are placed in the proper **cat***n* directory.

> NOTE: Producing formatted versions of all manual pages with **catman** takes about three hours and requires a significant amount of disk space (about 3000 1024-byte blocks in **/usr**). Be sure you can spare the space before executing **catman**.

Execute the command in the background by entering:

```
# catman &
```

catman also updates the **/usr/man/whatis** database, used by **apropos**. See *catman*(1) for more information.

NOTE: It is not mandatory to format manual pages with **catman**.

2.1.3 BSD Manual Page Viewing

As mentioned above, the **man** utility not only formats manual pages but also allows you to view them. Entering the command

```
% man apropos
```

will send the formatted copy of the *apropos*(1) manual page to the user's terminal.

When you use **man** to view a BSD manual page, and an up-to-date formatted version of that manual page exists, **man** does the following:

- Concatenates (using **cat**) the formatted manual page
- Pipes it through **ul** to process control characters, to embolden words, and to produce underlining

- Pipes it through **more** to control scrolling
- Sends it to the terminal

If no formatted manual page exists, **man** takes these steps:

- Informs the user that formatting is taking place
- Formats the manual page
- Stores the formatted page in the proper **/usr/man/cat***n* directory
- Pipes it through **ul** and **more** to the terminal

2.1.4 apropos and the /usr/man/whatis Database

The **apropos** command prints the names of manual pages whose titles contain instances of specified keywords. **apropos** gets its information from the file **/usr/man/whatis.** This file is created (or updated) by running either **catman** or the program **/usr/lib/makewhatis**. Note that the **/usr/lib/makewhatis** program has been run prior to distribution. **/usr/man/whatis** should be updated whenever manual pages are added to or removed from the system.

2.1.5 System V Manual Page Distribution

UTX/32 System V manual pages are distributed in both tape and hardcopy formats. The following subsections discuss these two modes of distribution.

Tape Distribution

The UTX/32 source tape contains the System V manual page source in the directory **/usr/src/5src/man**. The UTX/32 binary tape contains a version of manual pages already formatted in a *packed* format. Each manual page name in this format ends with the suffix **.z**. Individual System V chapters are within the directory **/usr/5lib/catman** in the subdirectories:

u_man/man1	User manual pages
p_man/man[2-5]	Programmer manual pages

Hardcopy Distribution

Table 2-3 shows how sections from the System V manual page set are incorporated into the UTX/32 manuals.

Table 2-3
System V Sections in the UTX/32 Manual Page Set

Section #	Document Title
1	*UTX/32 System V User's Reference Manual*
2, 3, 4 and 5	*UTX/32 System V Programmer's Reference Manual*
7	*UTX/32 Special Files Reference Manual*
1M and 8	*UTX/32 System Administrator's Reference Manual*

2.1.6 System V Manual Page Formatting

System V manual pages are formatted with the **sv_man** utility. This command will make manual pages suitable for printing on a lineprinter or viewing on a terminal screen.

Unlike BSD's **man** command, **sv_man** formats all the files in a directory rather than formatting individual files. Consequently, **sv_man** requires a directory name as its argument rather than a single file name. For example, the following command formats the manual pages in the **man1** directory:

```
# sv_man /usr/5lib/catman/u_man/man1
```

When System V manual pages are formatted, their file names have the suffix **.z**. They are stored in a packed format in the same directory as their unformatted counterparts.

The **sv_man** command formats with **nroff –manV**, which uses the **/usr/5lib/macros/an** macro set. See *man*(1) in the System V environment for more information.

2.1.7 System V Manual Page Viewing

Users read manual pages within each **catman** directory with the aid of the System V **man** utility. In System V, **man** is only good for viewing; it does not format manual pages as it does in BSD. For example, entering the command

```
$ man calendar
```

while in the System V environment, will send a formatted copy of the *calendar*(1) manual page to the terminal. If a formatted copy of the System V manual page does not exist, the System V **man** command will search for a BSD manual page. (Because this can be misleading to users, system administrators should ensure that a full set of System V manual pages are formatted at all time.) If neither exists, a message returns to the terminal screen stating that the manual page is not found.

The System V **man** command is a shell script that uses the **pcat** program to unpack the formatted, packed manual page and print it on the terminal screen. In System V, the output is not sent through **more** and **ul**. It goes directly to the terminal screen with no control of scrolling.

3 Booting and Halting UTX/32

The UTX/32 system and the hardware on which it runs have various operating modes. Being in one mode often implies being in another. The modes described in this chapter are:

- Console device modes
- Operating system modes

There is also a section explaining how to switch between the various modes.

3.1 Console Device Modes

The console device can be used in either of two modes: panel or console.

3.1.1 Panel Mode

In *panel mode*, all commands entered at the console keyboard directly manipulate memory or the CPU. When there is no operating system or other executive running on the CPU, this is the only console device mode available. Entering panel mode does not interfere with a running executive, unless panel mode commands are entered for that purpose.

Panel mode is indicated by the prompt

```
//
```

at the left of the console device display. All characters typed at the console device in panel mode must be capitalized. Some consoles convert lower case to upper case for you.

Some console devices have an extended keyboard with special console function keys on the right-hand side. These keys correspond to the most common panel mode functions: HALT (halt the CPU), CLE (clear memory), IPL (Initial Program Load), RST (reset registers), and so forth. These keys may be used in place of typing the corresponding command manually. Be aware, however, that the CLE key requires that a carriage return also be entered. For more information on the commands available in panel mode, ask your field representative for the manual appropriate for your system. This will be the *Input/Output Processor (IOP) Model 8000 and 8001 IPU Console IOP Reference Manual*) or the *Gould Multi-Function Processor Model 8002 Technical Manual.* These manuals are not included with the UTX/32 document set.

3.1.2 Console Mode

Console mode occurs only when an executive, such as UTX/32, is running on the machine. In console mode, commands typed at the keyboard are interpreted by the running executive. The executive could be an operating system or any of a number of diagnostic or special purpose programs.

3.2 UTX/32 Operating System Modes

The UTX/32 Operating System can be run in two modes: single-user and multi-user.

3.2.1 Single-User Mode

The term *single-user mode* defines a condition in which a limited subset of the services that make up UTX/32 are available to the user.

You must use the console device to operate UTX/32 in single-user mode. At the console, a user is superuser by default. Single-user mode is an intermediate, maintenance state of UTX/32: most actions of initial system installation are performed in this mode, and many periodic administrative activities require that the system be in it.

The prompt

```
#
```

displayed at the console often means that the user has superuser privileges. It may also mean that the system is in multi-user mode and that an administrator is logged on with superuser privileges. To check the actual situation, enter a **ps** or **w** command. See *ps*(1) and *w*(1) for more information on these commands.

Halting from Single-User Mode

To halt the system from single-user mode, enter the UTX/32 commands **shutdown** or **halt**. The panel mode command **HALT** can be used from the console in some circumstances. In all cases, be sure that the system is as quiescent as possible before halting; in other words, be sure that no extraneous processes are running and that all file systems except the root are dismounted if possible.

Using shutdown. In single-user mode, there is no need for the user amenities provided by **shutdown**. Simply enter

```
# /etc/shutdown -h now
```

This will synchronize the disks with their in-core images and then halt the system.

Using halt. Simply enter

```
# /etc/halt
```

This will synchronize the disks and halt the system.

> WARNING: If you are halting the system to reboot because **fsck** discovered an inconsistency in the root file system, *do not* synchronize the disks. To avoid synchronizing, use the command
>
> ```
> # /etc/halt -n
> ```
>
> If this fails, something is seriously wrong and you should use the console panel mode commands to halt the CPU.

See *shutdown*(8) and *halt*(8) for more information.

Using the Console Panel Mode Command. To halt the system using panel mode commands, type the characters

```
@@P
```

at the console. When the panel mode prompt appears, halt the CPU by entering the command

```
//HALT
```

Booting from Halted to Single-User Mode

When the machine is in the halted state and either no UTX/32 system is present in-core or you wish to execute a different version of UTX/32, you must boot the system in order to run the appropriate version of UTX/32. Follow these six steps:

1. At the console enter

```
//HALT                  (ensure that the CPU is halted)
//RST                   (reset the registers)
//CLE                   (clear memory)
//RST                   (reset any registers set by CLE)
```

   There will be some delay between pressing the return key after entering the **CLE** command and the appearance of the next // prompt. The amount of time required for **CLE** is approximately one second per megabyte of memory. (If using the **CLE** function key on the console, you must press the return key after pressing the function key.)

2. Next, enter the command

   ```
   //GPR7=3
   ```

   Setting general-purpose register 7 (GPR7) informs the bootstrap procedure how to boot the system. The register setting is in effect only for the subsequent IPL (Initial Program Load) instruction.

   Setting GPR7 to 0 (the default value after having entered the command **RST**) causes the system to be brought up directly to multi-user. File system checks will be done automatically, as long as no file named **/fastboot** exists in the root directory. See *fastboot*(8) for more information.

   Several bits in GPR7 alter the default behavior when set, as follows:

   - Bit 31 (value: 0x01) causes the bootstrap to prompt for the name of the file containing UNIX.
   - Bit 30 (value:0x02) causes the booted UNIX to be brought up in single-user mode rather than multi-user mode.
   - Bit 27 (value:0x10) causes the booted UNIX to run in physical memory only; swapping and paging will be disabled.
   - Bit 26 (value:0x20) causes the booted UNIX to use the device whose SelBUS address is in GPR6, rather than the device being booted from, as the root device.
   - Bit 25 (value:0x40) causes the bootstrap loader to print additional information while loading files (verbose mode).

   Any (or all) of these bits may be set before booting the system. All other bits are ignored.

3. Indicate where the executable image to boot is located, by entering the command

   ```
   //IPL=800                (or alternative boot address)
   ```

   where the IPL address is 0x800.

   For a Multi-Function Processor (MFP), enter the command

   ```
   //IPL=0x7Enn
   ```

   where *nn* is the remainder of the MFP address.

4. When the system is ready to execute an image, it will prompt for the name of the image to be booted if GPR7 is set to an odd value. To boot UTX/32 at this point, respond by entering the name **unix**:

   ```
       Boot: unix
   ```

   If you wish to see the contents of the root directory, /, instead of booting, you may type **ls** in response to the Boot: prompt. If you wish to boot another version of the kernel (**unix.bak**, for example, if it exists) enter the name of that version (or alternative) instead of **unix**.

5. After **unix** has been entered, the system will come up in single-user mode. At that time, run **fsck** or **preen** to check and correct any file system inconsistencies. See *fsck*(8) and *preen*(8) for more information.
6. Set the system date. See Section 4.1, ''Setting the System Date,'' for instructions.

3.2.2 Multi-User Mode

The term *multi-user mode* defines a condition in which all UTX/32 system services are running and the system is available to general users. When UTX/32 is executing in this mode, the system is *up*.

In multi-user mode, the console can be used as a regular terminal. However, it is inadvisable to allow console access to anyone but administrators.

Halting from Multi-User Mode

Occasionally it is necessary to halt the system to perform routine hardware maintenance or perhaps to power off the system. The commands **shutdown** and **halt** can be used for this purpose.

shutdown is preferable to **halt**, as it is more polite and flexible. It sends warning messages to all system users; it can shut the system down at a predetermined future time; it creates the file **/etc/nologin**, whose presence prevents non-administrators from establishing new login sessions. If desired, **shutdown** can be used to go to single-user mode without halting the system. **halt** halts the system without any of these features and may result in file system damage.

In cases where one of these two commands fails, the system can be halted by entering panel mode at the console and using panel commands. See *shutdown*(8) and *halt*(8) for more information.

Using shutdown. To halt the system immediately using **shutdown**, become superuser and enter the command

```
# /etc/shutdown -h now
```

shutdown with the **–h** option will **sync** the disks and halt the processor. The console device will be in panel mode at this point.

A time and message may be supplied with the **shutdown** command. For example,

```
# /etc/shutdown -h 17:30 Installing new firmware
```

will inform users that the system will be shut down at 5:30 P.M. to install new firmware. A relative time (+30) which indicates a **shutdown** in thirty minutes can also be specified.

> NOTE: **shutdown** creates the file **/etc/nologin** whose presence prevents non-administrators from establishing new login sessions; however, administrators can log in as **root** regardless of the existence of **/etc/nologin**. This allows them to stop **shutdown** if necessary. The file is created five minutes before the designated shutdown time. **/etc/nologin** is automatically removed when the system comes up multi-user.

Using halt. Using **halt** from multi-user mode may damage the file systems and is not recommended. However, because **halt** is quicker than **shutdown**, it may be useful during emergencies, when speed is of the essence.

To halt a running system using **halt**, become superuser and enter the command

```
# /etc/halt
```

In dire circumstances, the following command line will work even faster (because it avoids synchronizing the disks and killing processes)

```
# /etc/halt -q
```

If the **halt** command is used, you *must* do file system consistency checks upon rebooting the system.

Using the Console Panel Mode Command. If the system is up multi-user, but there is no response on any of the terminals or from the console used as a terminal, the desperation method of halting the system—the method to use when all else fails—is as follows. At the console, type the characters

```
@@P
```

When the panel mode prompt appears, halt the CPU by entering the command

```
//HALT
```

If this method of halting the system is used, you *must* do file system consistency checks upon rebooting the system.

Booting from Halted to Multi-User Mode

There are two methods for booting a halted system to multi-user mode. The first method involves immediately booting to multi-user mode. See "Booting from Halted to Single-user Mode," in Section 3.2.1, "Single-User Mode."

To boot to single-user mode before coming up multi-user, see ''Booting from Halted to Single-User Mode'' in Section 3.2.1, ''Single-User Mode.'' Then use the **fsck** or **preen** commands to perform file system checks. See *fsck*(8) and *preen*(8) for more information. When all file system checks are successfully completed, the system date is set correctly (see Section 4.1, ''Setting the System Date''), and any other maintenance tasks are completed, invoke multi-user mode by typing

```
# ^D
```

3.2.3 Switching between Modes

The following sections give instructions for switching between the various CPU, console, and UTX/32 modes

Console to Panel

It is possible to enter panel mode from the console device by entering the characters

```
@@P
```

at the console device keyboard when the panel key is in diagnostic or panel mode. These characters are interpreted by the Input/Output Processor (IOP) or Multi-Function Processor (MFP) without being sent to the CPU. The console device will display the panel mode prompt // in response to these characters.

If the system must be halted, type **HALT** at the panel mode prompt:

```
//HALT
```

WARNING: This procedure will halt the system abruptly. It may cause file system damage or loss of some contents of memory that have not yet been saved on disk. It should be used only when it is not possible to use the **shutdown, halt**, or **fastboot** commands.

3.2.4 Panel to Console

Being in console mode implies the presence of some executive. If none is currently present, the only way to attain console mode from panel mode is to initiate an executive: for instance, by booting **unix** or a diagnostic package. If an executive is present and has not been halted, you can continue execution and return to console mode from panel mode by entering the characters

```
@@C
```

on the console keyboard. If the executive has been halted while in panel mode (*not* as the result of an error or failure of the executive itself), entering the command

```
//RUN
```

will restart the CPU and return the console device to console mode.

Single- to Multi-User Mode

Before bringing your system up to multi-user mode from single-user mode, be sure to dismount all file systems (if any) that were mounted while in single-user mode or that were unintentionally left mounted by a system crash. If a file system is mounted when going to multi-user mode, the mount table, **/etc/mtab**, and commands that rely on **/etc/mtab** may be wrong. As distributed, UTX/32 dismounts all file systems as part of system startup. See *rc*(8), *mtab*(4), *umount*(8), and Section 5.11, ''Dismounting File Systems with **umount**,'' for more information.

> NOTE: Failure to dismount all file systems results in an incorrect **/etc/mtab** file in multi-user mode. **mount** and **df** rely on the information in **/etc/mtab** being correct. See *mount*(8) and *df*(8) for more information.

With all file systems dismounted, decide whether to check file systems for possible consistency problems. Always check /, the root file system, since the root is critical to the system. Check all file systems in these cases:

- Your system has been halted but was not halted gracefully with **shutdown** or **halt**.
- Your system reports problems during **shutdown** or **halt**. For example, you have received the message

  ```
  CAUTION: some process(es) wouldn't die
  ```

- Your disk packs or drives have changed since the last **shutdown** or **halt**.

If your system was not halted but only brought down to single-user mode from multi-user mode, it may not be necessary to check all the file systems. Use your best judgment, remembering always to err on the side of caution.

> NOTE: It is best to check all file systems if you are unsure whether there are problems. See Section 5.7, ''Checking File System Consistency'' for more information.

To check the consistency of all file systems listed in the **/etc/fstab** file, enter the command

```
# /etc/fsck -p
```

The **–p** option causes **fsck** to check several drives in parallel, according to the order indicated in **/etc/fstab**.

> NOTE: If the **–p** option is used with **fsck**, the information in **/etc/fstab** must be complete and correct. Otherwise, the file system checks may not complete properly.

If problems arise or if you want more information, see *fsck*(8), Section 5.7, "Checking File System Consistency" and "FSCK — The UNIX File System Check Program" in the UTX/32 supplementary documents.

After your file systems have successfully completed the consistency checks, you are ready to bring your system up multi-user. If your system is in single-user mode and all file systems are dismounted, type

```
# ^D
```

This will initiate the process of bringing the system up to multi-user mode. The **init** program executes **sh** to run **/etc/rc**. In addition to controlling the automatic reboot, **rc** executes **rc.local**. After successful execution of **rc**, **init** forks a **getty** process for each terminal specified in **/etc/ttys**. See *init*(8), *getty*(8), and *rc*(8) for more information.

Multi- to Single-User Mode

Occasionally it may be necessary to take the system down to single-user mode without halting it. Most such instances should be for regularly scheduled tasks. For example, because system backups should be done while file systems are inactive, they should probably be done in single-user mode. Typically, the system will be in single-user mode for a relatively short time and then returned to multi-user mode without halting or rebooting.

The **shutdown** command is ideal for this purpose. It can be used to shut the system down gracefully at a specified time. It also performs such services as sending periodic warning messages to users about the impending shutdown and preventing logins to the system for a period of time immediately before the shutdown is to occur.

For example, to take the system down to single-user mode at 7:30 P.M. for weekly archiving and to indicate that the system is expected to be back in service in one hour, the administrator, working as superuser, would enter the command

```
# /etc/shutdown 19:30 Weekly archiving - system back in one hour
```

Rebooting from Multi-User Mode

Occasionally, an administrative task that can be accomplished without changing to single-user mode does require that the system be rebooted. For example, a new version of **unix** might need to be booted. In this case, the administrator can reboot from multi-user mode to multi-user mode, bypassing single-user mode. This procedure can be followed for tasks that do not require administrator intervention, such as resetting some of the hardware.

The simplest way to accomplish a reboot of this kind is to use **shutdown** with the **–r** (reboot) option. See *shutdown*(8) and *reboot*(8).

As an example, to immediately reboot the system, type

```
    # /etc/shutdown -r now
```

as superuser. **shutdown** will perform an automatic reboot procedure.

This example is a bit abrupt, however, if any users are on the system. It is recommended that, if possible, a time and message be supplied with the **shutdown** command. For example,

```
# /etc/shutdown -r +5 "Increasing disk buffer cache"
```

would inform users that the system will be rebooted in five minutes to add more disk buffers.

> WARNING: When rebooting a kernel, it is possible that the boot device has not been configured into the kernel. Should this occur, the system will halt during the boot process with no diagnostic messages displayed.
>
> Should an unexplained halt occur while booting a new kernel, reboot the default/backup system and inspect the output of the reconfiguration to insure that the boot device is configured in the kernel.

4 Setting the Date and Using Cron

4.1 Setting the System Date

The *system date* represents the system's understanding of the current time and date. The system date is used by the kernel and by various user and system programs to:

- Provide time stamps for various types of events
- Schedule events (for example, using **cron**)

The system date must be reset to the correct time whenever it is incorrect (usually after a reboot). This is done with the **date** command.

An administrator must be superuser to set the date, and it is recommended that the system be in single-user mode when the date is set. To set the date, enter

```
# date yymmddhhmm
```

For example,

```
# date 8607041930
```

sets the date to 7:30 P.M., July 4, 1986.

See *date*(1) and *cron*(8) for more information.

4.2 Setting Timezone and Daylight Savings Time Information

UTX/32 keeps the system date using Universal Time. To convert to local time, the system must know what timezone it is in and whether standard or daylight saving time is in effect. The rules for determining whether DST is in effect on a particular date are complicated. The laws determining the changeover dates have been changed many times and are likely to change again. UTX/32 therefore contains a rule-based mechanism for determining the appropriate local time conversion, allowing individual sites to alter the rules to deal with local variations. UTX/32 processes obtain timezone and daylight savings time information from one of the binary data files in **/etc/zoneinfo** or its subdirectories. By default, the file used is **/etc/zoneinfo/localtime**.

/etc/zoneinfo/localtime must be created at installation time. It should be a link to one of the distributed standard timezone data files in **/etc/zoneinfo** or its subdirectories. The link may be created manually or by using **zic**, as in this example:

```
# zic -l US/Pacific
```

In the United States, the link would be to one of the files in the directory **US**, such as **US/Eastern** or **US/Central.** In Europe, the link would be to a file in **/etc/zoneinfo** such as **WET**, **MET** or **EET**. See *zic*(8) for more information.

The information in **localtime** is used by the **tzset** function, which sets the timezone for a process. **tzset** is called the first time a process calls **ctime**, **asctime**, **localtime** or **gmtime**. See *tzset*(3C), *ctime*(3C), *asctime*(3C), *localtime*(3C), and *gmtime*(3C) for more information.

Users can override the default data file by setting the environment variable **TZ** to an alternate data file in **/etc/zoneinfo** or one of its subdirectories. For example, **TZ** might be set to **US/Eastern**. Note that this is not an absolute pathname, but relative to the directory **/etc/zoneinfo**.

In the System V environment, there is a rigid format for **TZ** values. For convenience, there are files in **/etc/zoneinfo** with names (like EST5EDT) that follow the required format.

The source files for the binary data files are in **/etc/zoneinfo/ReadableDatafiles**. A file, **Theory**, explaining differences between this implementation and previous BSD and System V implementations is located in this directory along with one file for each general area of the world. Here is an excerpt from **/etc/zoneinfo/ReadableDatafiles/northamerica**:

```
# Rule   NAME     FROM     TO       TYPE     IN       ON       AT       SAVE     LETTER/S
Rule     US       1975     only     -        Feb      23       2:00     1:00     D
Rule     US       1976     1986     -        Apr      lastSun  2:00     1:00     D
Rule     US       1987     max      -        Apr      Sun>=1   2:00     1:00     D

# Zone   NAME              GMTOFF   RULES/SAVE        FORMAT    [UNTIL]
Zone     US/Eastern        -5:00    US                E%sT
Zone     US/Central        -6:00    US                C%sT
Zone     US/Mountain       -7:00    US                M%sT
Zone     US/Pacific        -8:00    US                P%sT
Zone     US/Yukon          -9:00    US                Y%sT


# Mainland US areas that are always Standard as of 1986.

Zone     US/East-Indiana   -5:00     US       E%sT     1946
                           -5:00     -        EST               # Always EST as of 1986
Zone     US/Arizona        -7:00     US       M%sT     1946
                           -7:00     -        MST               # Always MST as of 1986
```

If updates or corrections are necessary, the changes must be made in the appropriate source file in **/etc/zoneinfo/ReadableDatafiles** or in **/usr/src/etc/zic** for source distributions. To compile new binary data files, use the command **zic**. When **zic** is run on the file **northamerica**, new files **Eastern**, **Central**, **Mountain**, **Pacific**, **Yukon**, **East-Indiana** and **Arizona** are created in the **US** directory.

WARNING: Because of the way the new system works, it is not possible to support the old BSD **timezone**() function. Old programs that used it can usually be altered to instead examine

```
tzname[localtime(&clock)->tm_isdst]
```

or the less portable

```
localtime(&clock)->tm_zone
```

to get the correct timezone abbreviation.

Time functions no longer use the kernel timezone configuration parameter.

The undocumented BSD **dysize**() function is still included.

Some sites have programs that are restricted to run only during certain hours (games at night, or company accounting during the day). To avoid the problem of users setting their **TZ** variable to some other zone to get around such restrictions, an administrator can do one of three things:

1. Add a file by the name of **/etc/zoneinfo/NO_TZ_FROM_ENV**. This will cause **TZ** variables to be ignored by every program. This does not require any recompilation but may be more drastic than desired.
2. Write restricted programs to convert all times to a standard zone (Greenwich Mean Time (GMT) is easiest) and do time comparisons there. Unless GMT is your usual zone, though, this is not convenient.
3. Add a call to **tzsetwall**() to the beginning of each program you wish to restrict. (The **tzsetwall**() call must come before any calls to other time functions.) This will ensure that **/etc/zoneinfo/localtime** is always used when that program is run.

4.3 /usr/lib/crontab

The file **/usr/lib/crontab** specifies system activities that are to be performed at particular times. The clock daemon, **cron**, examines the file once a minute and performs all system activities whose time specifications match the current time.

Each line entry in the **crontab** file describes one system activity to be performed at specified times. Each line has this form:

min hr day_of_mon mon_of_yr day_of_wk user_name cmd_line

user_name is the name under which the activity will be performed. *cmd_line* is a command line specifying the activity to be performed. It must specify a command using a complete pathname. The remainder of the line is a five-part time specification indicating when the command line is to be executed.

The five components of the time specification are

- Minutes
- Hours
- Days of the month
- Months of the year
- Days of the week

Each component of the time specification can consist of the following:

- A number
- A set of numbers in a comma separated list
- A range of numbers indicated by two numbers separated by a minus sign
- All numbers indicated by an asterisk
- A combination of the above

The current time must match all of the parts of the **crontab** time specification before the specified command line will be executed.

The ranges of valid numbers for each part of the time specification are as follows:

Minutes
: 0 through 59

Hours
: 0 through 23

Days of the month
: 1 through 31

Months of the year
: 1 through 12 (1 = January)

Days of the week
: 1 through 7 (1 = Monday)

The following are sample **crontab** entries.

`0 0 * * *` *user_name cmd_line*
: *cmd_line* is executed at midnight every day.

`30 7,17 * * 5` *user_name cmd_line*
: *cmd_line* is executed at 7:30 A.M. and 5:30 P.M. every Friday.

`0 4 1 5-9 *` *user_name cmd_line*
: *cmd_line* is executed at 4:00 A.M. on the first day of the month for the months May through September.

`15 8 4 7 *` *user_name cmd_line*
: *cmd_line* is executed at 8:15 A.M. on the fourth of July.

If not carefully managed, the **crontab** file can become large and difficult to maintain. The following organization is recommended to minimize the maintenance burden:

- Execute shell scripts, not UTX/32 commands, from the **crontab** file. This reduces the need to edit the **crontab** file itself. It also helps to group tasks.
- Relate the names of the shell scripts to their functions. Name them with a **.sh** suffix so that they can be easily identified. Examples:

  daily.sh
  Daily system activities

  weeklyacct.sh
  Weekly accounting activities

- Place the shell scripts in the directory **/usr/adm**.
- Arrange that no more than one shell script is executed at any one time. This helps eliminate sudden, potentially large, increases in the system load.

Note that although this organization reduces the size and complexity of the **crontab** file, it also restricts the generality of the entries.

The following sample **crontab** file demonstrates the organization described above:

```
0,10,20,30,40,50 * * * *   root /usr/adm/tenminutes.sh
1 0 * * *                  root /usr/adm/daily.sh
5 5 * * 1                  root /usr/adm/weekly.sh
5 6 * * 1                  root /usr/adm/weeklyacct.sh
5 6 1 * *                  root /usr/adm/monthlyacct.sh
5 5 * * 4                  root /usr/adm/thurs.sh
```

The sample shell scripts are executed at the times indicated:

tenminutes.sh
On the hour, and 10, 20, 30, 40, and 50 minutes after the hour, every day

daily.sh
Every day at 1 minute after midnight

weekly.sh
Every Monday at 5:05 A.M.

weeklyacct.sh
Every Monday at 6:05 A.M.

monthlyacct.sh
On the first of every month at 6:05 A.M.

thurs.sh
Every Thursday at 5:05 A.M.

The following list shows activities that could be performed by each of the shell scripts:

tenminutes.sh

- Executing **dmesg**
- Sending timestamps to the console

daily.sh

- Executing **calendar**
- Removing core dumps
- Removing old files within **/usr/preserve**
- Removing old messages within **/usr/msgs**
- Saving old system logs (see *syslog*(8))

weekly.sh
Saving log files (for example, **/usr/adm/messages** and **/usr/adm/wtmp**)

weeklyacct.sh
Generating weekly accounting reports

monthlyacct.sh
Generating monthly accounting reports

thurs.sh
Sending out end-of-week reminders, in this case on Thursday, to users (for example, timecard reminders).

For more information, refer to *cron*(8) and Section 13.41, "**/usr/lib/crontab.**"

5 Maintaining the File System

The UTX/32 file tree requires periodic maintenance. Building and maintaining the tree involves a variety of administrator activities, including

- Formatting disks
- Initializing disks for UTX/32
- Partitioning disks
- Creating file systems
- Checking file systems for consistency
- Setting up the file system table in **/etc/fstab**
- Configuring and maintaining disk partition mirror sets
- Mounting and dismounting file systems

To limit the risk of data contained on the file systems being irretrievably lost, all file systems should be backed up to tape on a regular schedule. See Chapter 15, ''Performing System Backups.''

5.1 The UTX/32 Binary Tree

There are many references throughout this document that contain pathnames of various length. To help administrators see an overview of the distributed UTX/32 binary tree structure, the following figures have been provided. For an overview of the UTX/32 source tree structure, see Figure 6-1.

5.2 Building File Systems: An Overview

The steps necessary to add a new disk drive to the system and to create accessible file systems on it are as follows:

1. Format your disk. Use Gould diagnostic programs to format DPII (Disk Processor II), UDP (Universal Disk Processor), and HSDP (High-Speed Disk Processor) disk packs and deallocate bad blocks. Use **prep** to format SCSI (Small Computer System Interface) disks on MFP (Multi-Function Processor) controllers. See Section 5.3, ''Formatting Disks'' and **prep**(8) for more information.
2. Use **prep** to initialize your disks. See Section 5.4, ''Initializing Disks for UTX/32'' and *prep*(8) for more information.
3. Decide on the usage, arrangement, and sizes of the disk partitions. See Section 5.5.1, ''Organizing Partitions and File Systems.''
4. If not using the standard partitions, calculate the cylinder offsets and sizes for the chosen partition arrangement. See Section 5.5.2, ''Calculating Cylinder Offsets.''

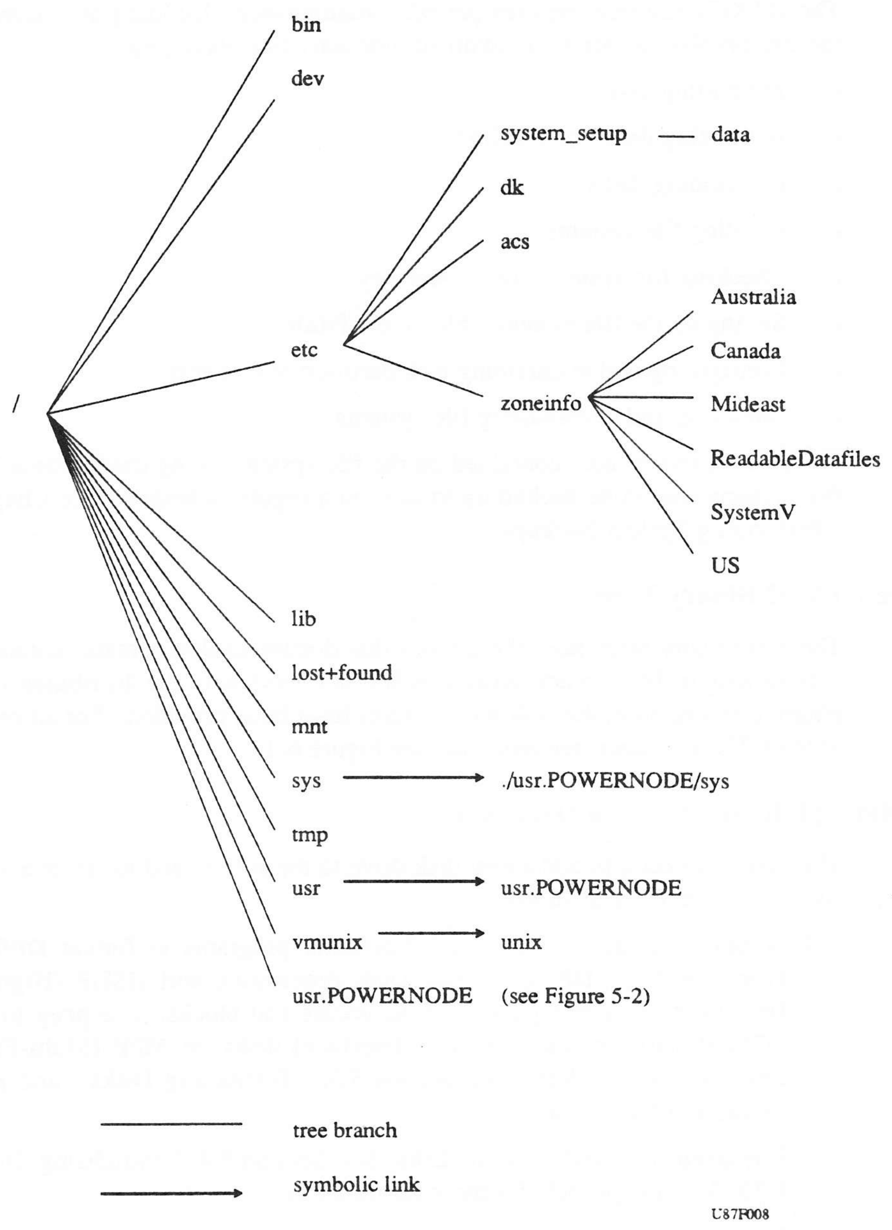


Figure 5-1. UTX/32 Binary Tree

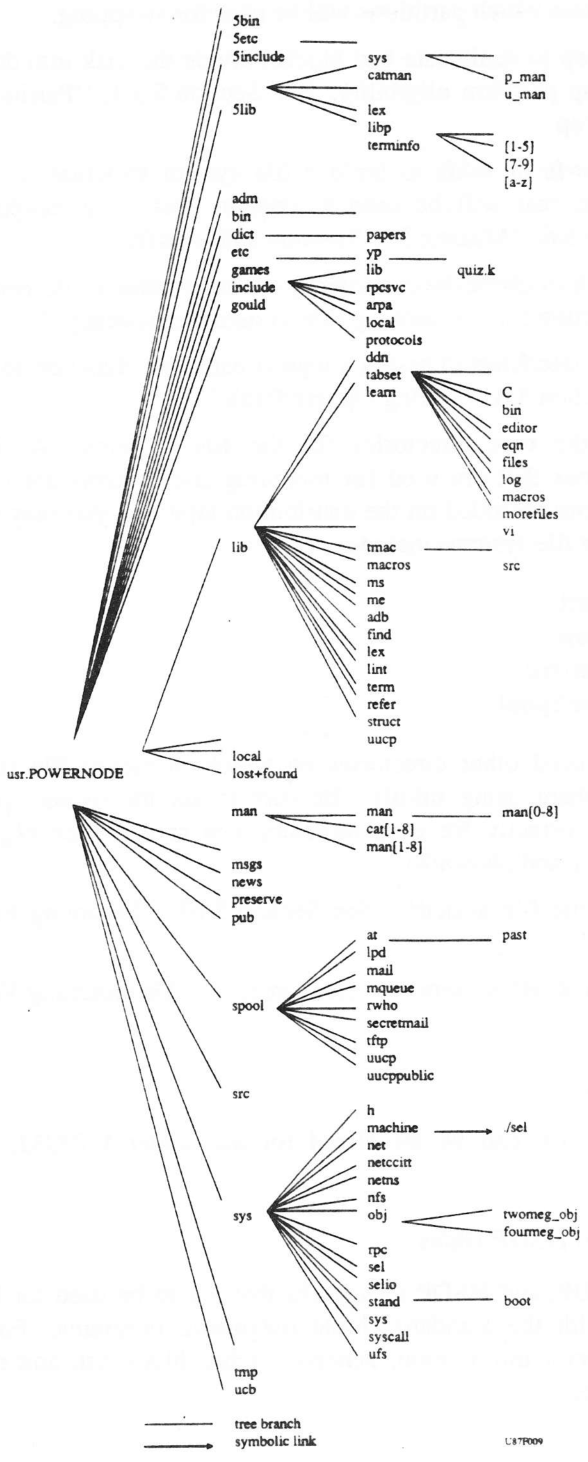


Figure 5-2. The Binary Tree under **/usr.POWERNODE**

5. Determine which partitions will be used for swapping.

6. Use **prep** to deallocate bad blocks, divide the disk into disk partitions, and set swap partition eligibility. See Section 5.5.3, ''Partitioning Disk Packs with **prep**.''

7. Use **newfs** or **mkfs** to build a file system structure on top of each disk partition that will be used to contain data. See *newfs*(8), *mkfs*(8), and Section 5.6, ''Making File Systems with **newfs**.''

8. Use **fsck** to check the consistency of newly created file systems. See *fsck*(8) and Section 5.7, ''Checking File System Consistency.''

9. Update **/etc/fstab** to assign a logical name and function to each file system. See Section 5.8, ''Setting Up **/etc/fstab**.''

10. Make the root directories for the file systems. At installation time, directories that are used for mounting file systems are / and **/usr**. Other directories included on the distribution tape that you may wish to mount as separate file systems include

    /mnt
    /tmp
    /usr/src
    /usr/spool

    If you need other directories on which to mount file systems, you may create them using **mkdir**. Be sure to set the owner, group, and access modes correctly for each directory you create. See *chgrp*(1), *chmod*(1), *mkdir*(1), and *chown*(8).

11. Mount the file systems. See Section 5.10, ''Mounting File Systems with **mount**.''

To dismount a file system, see Section 5.11, ''Dismounting File Systems with **umount**.''

5.3 Formatting Disks

Before any disk can be initialized for use under UTX/32, it must first be formatted.

5.3.1 DPII, UDP, and HSDP Disks

All DPII, UDP, and HSDP disk packs that are to be used for UTX/32 must be formatted with the standard Gould diagnostic programs. Formatting a pack initializes sector information, generates a bad-block list, and confirms that the disk is usable.

For more information, refer to the level-two diagnostic program of the Gould RPU Disc Processor Media Verification Program (RP.MVP) in the *RPU Disc Processor Media Verification (RP.MVP.J) Disc Utility Description* or the *Gould High Speed Disc Processor Media Verification Program (DP.MVP.R) Program Description.*

NOTE: UDP disks must be formatted for 16 sectors per track. HSDP disks must be formatted based on drive model as shown in Table 5-1.

Table 5-1
Formatting HSDP Disks

Gould Model #	Manuf. Model #	Sectors per Track
8888	CDC 9772	43 sectors at 1024 bytes per sector
8884	Fujitsu 2351A	24 sectors at 1024 bytes per sector
8889	Fujitsu 2361A	35 sectors at 1024 bytes per sector

5.3.2 MFP (SCSI) Disks

All SCSI disks on MFP controllers must be formatted with **prep** before they can be used under UTX/32. See *prep*(8) for more information.

5.4 Initializing Disks for UTX/32

Before any disk can be partitioned for use under UTX/32, it must first be initialized using **prep**. This initialization writes geometry, flaw map, and other information used by UTX/32 into reserved areas of the disk. See *prep*(8) for more information.

5.5 Partitioning Disks and Setting Swap Partition Eligibility

The UTX/32 operating system is capable of dividing a disk into as many as eight logical disks. These logical disks, called *partitions*, can be accessed independently. Each partition can be used to contain the information for a single file system. A *file system* is a subset of the UTX/32 file tree.

It is necessary to partition a disk whenever it is to be used on your UTX/32 system for the first time. This may occur when

- You are installing the system
- You need an additional disk to build new file systems or add swap space, and there are no free partitions on the disks currently being used

You may also want to rearrange the partitions on a used disk. In this case, be absolutely sure that you no longer need the information contained on the disk or that the information has been saved to tape so that it can be restored to disk after you have completed building the new file systems.

You may want to set swap partition eligibility on partitions other than the default **b** partition. The eligibility of a swap partition is set during the partition sizing operation.

The following steps are necessary to partition a disk and set swap eligibility:

1. Decide on the usage, arrangement, and sizes of the disk partitions. See Section 5.5.1 ''Organizing Partitions and File Systems.''
2. If not using the standard partitions, calculate the cylinder offsets necessary to conform the disk partitions to the chosen partition arrangement and sizes. See Section 5.5.2 ''Calculating Cylinder Offsets.''
3. Determine which partitions will be used for swapping.
4. Use **prep** to deallocate bad blocks, divide the disk into disk partitions and set swap partition eligibility. See Section 5.5.3 ''Partitioning Disk Packs with **prep**.''

NOTE: Disks must be formatted before they can be partitioned with **prep**. See Section 5.3, ''Formatting Disks.''

5.5.1 Organizing Partitions and File Systems

Before a disk can be partitioned, you must decide on the usage, arrangement, and sizes of the disk partitions. This information should be written down using the templates provided in Appendix A, "Disk Partition Templates."

In general, partitions are used

- To hold file systems
- As swap space

The standard arrangement of partitions on a disk is illustrated in Figure 5-3. This arrangement is the default assumed by **prep**, if no alternate arrangement is specified.

NOTE: Cylinder number zero is the outermost cylinder on the disk. Therefore, partition **a** exists in the outermost cylinder on the disk.

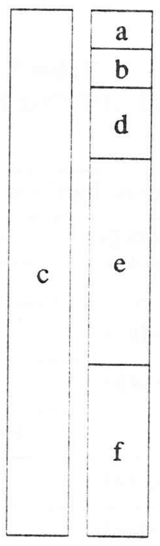


Figure 5-3. Standard Arrangement of Disk Partitions

Partitions are labeled **a** through **h**. As indicated in Table 5-2, partitions **a**, **b**, **d**, **e**, and **f** follow each other physically and encompass the entire disk; partition **c** overlaps all others. Partitions **g** and **h** are not configured as part of the standard arrangement. Consequently, they have no default size and may be configured with any size and used for any purpose. Partitions **e**, **f**, **g**, and **h** may be used as needed, depending on the disk layout desired and the size of the disk. By default, only the **b** partition is set eligible for swapping. Other partitions may be made eligible as required. It is recommended that you use this default arrangement when partitioning disks unless you have a specific need for an alternate arrangement. You may choose to use a different arrangement if, for instance, you want eight nonoverlapping partitions on the same disk.

The following tables illustrate standard partition sizes for various disk types.

Table 5-2
Standard UDP Disk Partition Sizes

Partition	Size of Partition (in 1024-byte blocks)				
	80Mb drive models 9342 or 8136	160Mb drive model 8146	300Mb drive model 9344	340Mb drive model 8858	675Mb drive models 8155 or 8850
a	18000	18000	18240	18048	18560
b	20000	20000	20064	20352	20480
c	entire disk				
d	27520	50000	50160	50304	50560
e	size of c - size of (a+b+d)		100016	100224	100480
f	0	0	size of c - size of (a+b+d+e)		
g	0	0	0	0	0
h	0	0	0	0	0

Table 5-3
Standard HSDP Disk Partition Sizes

Partition	Size of Partition (in 1024-byte blocks)		
	CDC 9772 drive model 8888	Fujitsu 2351A drive model 8884	Fujitsu 2361A drive model 8889
a	18144	18400	18360
b	20160	20240	20400
c	entire disk		
d	50400	50140	50320
e	100128	100280	100640
f	size of c - size of (a+b+d+e)		
g	0		
h	0	0	0

Table 5-4
Standard MFP Disk Partition Sizes

Partition	Size of Partition (in 1024-byte blocks)	
	CDC 99161-150Mb drive model	CDC 99161-300Mb drive model
	18144	*
b	20088	*
c	entire disk	
d	50058	*
e	size of c – size of (a + b + d)	
f	0	0
g	0	0
h	0	0

* Partition sizes were not available at the time of printing.

The standard partition sizes are derived from a minimum size for each partition, rounded up to the nearest cylinder boundary for each disk type. For the partitions **a**, **b**, **d**, **e**, and **h**, the minimum sizes are 18000, 20000, 50000, 100000, and 0, respectively.

The sizes of some partitions may be smaller than the standard sizes (they may even be 0), depending on the size of the disk and the number of defects on it. As an example, an 80Mb disk normally uses only partitions **a** through **d**; the sizes of **e** through **h** are 0.

> WARNING: The sizes of the partitions used for the root file system and default swap space *must* be at least as large as the standard sizes of the **a** and **b** partitions for an 80Mb pack, respectively.

The following sections give details on

- Partitions necessary for the proper operation of UTX/32
- Swap partitions
- Optional file systems that can be defined to increase system flexibility
- Recommended disk partition organizations

Required Disk Partitions

Some partitions are necessary for the proper operation of UTX/32. Partitions must exist for the following:

Root file system
: This file system, /, contains the kernel and the most commonly used commands. The root file system must be on partition **a** of the root volume. The root partition must be at least 18000 blocks long, which is the standard size for an **a** partition on an 80Mb pack.

/usr file system
: This file system contains the remaining commands and libraries. This filesystem must be at least 50000 blocks long which is the standard size for the **d** partition on an 160Mb disk.

Because of the partition sizes required for the root and **/usr** file systems, a minimum of two 80Mb disks or one 160Mb (larger disk) is necessary for a UTX/32 system.

Swap Partitions

UTX/32 may operate in memory only or virtual memory modes. In the memory only mode, the total memory requirements of all processes must not exceed physical memory. In virtual memory mode, the total memory requirements may be many times that of physical memory. The portions of the processes that cannot fit into physical memory are stored in special disk partitions known as swap partitions.

If UTX/32 will always be run in the memory only mode, no swap partitions are necessary. However, the normal operating mode for UTX/32 is the virtual memory mode, and at least one swap partition (e.g. the default swap partition) must be allocated on the root volume. Partition **b** of the boot volume is conventionally the default swap partition; it must be at least **dmmax** disk blocks in size, and should be 3500 blocks larger than the amount of physical memory on your system. The 3500 extra blocks leave enough space to all UTX/32 to come up to multi-user and not corrupt the dump image. The standard size for disk partition **b** depends on the size of the disk, but it is at least 20000 blocks. Additional swap partitions may be allocated if there is high swap activity, preferably on different disk volumes.

> WARNING: Only partitions that have been marked as swappable by **prep** can be used for swapping. See *prep*(8) for more information on setting the swap eligibility of partitions.
>
> WARNING: If the default swap partition is too small, system crash dumps are truncated.

WARNING: If you are using the standard partition arrangement, partition **c** must never be the default swap partition.

Optional File Systems

Parts of the UTX/32 file tree can be distributed on various partitions to increase

- File system flexibility
- File access efficiency

The following is a list of UTX/32 file tree areas, with suggestions as to whether they need to be put on their own partition. Refer to Figure 5-1, "The UTX/32 Binary Tree," and Figure 5-2, "The Binary Tree under **/usr.POWERNODE.**"

/tmp
This directory contains temporary files. **/tmp** can be a small file system (for example, an **a** or **d** partition), or if activity in **/tmp** is likely to be low, it can be left as part of the root file system.

/usr/src/src
This directory contains the source (only available with source distributions) for the UTX/32 kernel and for BSD utilities. It should be made a separate file system. Refer to Chapter 6 "Source Code" and Figure 6-1 "The UTX/32 Source Tree" for more information.

/usr/src/5src
This directory contains the source for System V utilities (only available with source distributions). It should be made a separate file system. Many times **/usr/src/src** and **/usr/src/5src** occupy the same disk partition. Refer to Chapter 6 "Source Code" and Figure 6-1 "The UTX/32 Source Tree" for more information.

/mnt
This directory is the users' file tree. If there are many users on your system, it is preferable to make it a separate file system.

/usr/adm
This directory contains space for accounting and audit trail files. If either of these facilities are used, these files can grow very large, so it may be necessary to make **/usr/adm** a separate file system.

/usr/spool
This directory contains space for spool directories. Depending on the spooling requirements of utilities such as **mail** and **lpr**, it may be necessary to make **/usr/spool** a separate file system.

The partitions available for additional file systems and swap partitions depend on the size of the disk and the partition sizes and arrangement. For example, two usable combinations of default partitions on an 80Mb disk with the standard partition sizes and arrangement are

- **c**
- **a**, **b**, and **d**

If used as a boot disk, **a** contains the root file system and **b** is the default swap partition, so only the **d** partition on an 80Mb pack is free to be used for an additional file system.

Usable partitions on a 300Mb disk are either

- **c**
- **a**, **b**, **d**, **e**, and **f**

If used as a boot disk, **a** contains the root file system, **b** is the default swap partition, and the free partitions are either **d** and **g** or **d**, **e**, and **f**.

Recommended Disk Partition Organizations

Follow these recommendations *only* if using the standard disk partition sizes and arrangement.

If two 80Mb disks are available, the following partition organization is recommended:

Table 5-5
Partitioning Two 80Mb Disks

Disk 1	
/dev/dk0a	/ (root)
/dev/dk0b	default swap space
/dev/dk0d	**/mnt**
Disk 2	
/dev/dk1c	**/usr**

Root, /**mnt**, and the default swap partition are all located on the first disk. /**tmp** is a directory under root on the first disk. The /**usr** file system requires a **c** partition (/**dev/dk1c**) on the second disk.

If a single 300Mb disk is available, this disk layout is recommended:

Table 5-6
Partitioning One 300Mb Disk

/dev/dk0a	/ (root)
/dev/dk0b	default swap space
/dev/dk0d	**/usr**
/dev/dk0e	**/mnt**
/dev/dk0f	**/tmp** or **/usr/src**

/dev/dk0f can be mounted as **/tmp** and used for temporary space, or it can be mounted as **/usr/src** and hold the system source (if you have a source distribution).

> NOTE: If you want to include the entire UTX/32 source tree, plus object files and executables, a **g** partition is required. In this case, **/mnt** and **/tmp** have to be directories under root, and **/dev/dk0g** must be allocated for source.

If two 300Mb disks are available, this disk layout is recommended:

Table 5-7
Partitioning Two 300Mb Disks

	Disk 1
/dev/dk0a	/ (root)
/dev/dk0b	default swap space
/dev/dk0d	**/usr**
/dev/dk0g	**/usr/src**
	Disk 2
/dev/dk1a	**/tmp**
/dev/dk1b	additional swap space
/dev/dk1d	free
/dev/dk1e	**/mnt**
/dev/dk1f	**/usr/spool**

The **/dev/dk0g** partition can be used to hold the system source, if you have a source distribution. Using a **g** partition ensures that you will have room to compile the entire source tree if you wish. The **/dev/dk1f** partition can be used for **/usr/spool** or for **/usr/adm**. Partition **/dev/dk1d** could also be used.

5.5.2 Calculating Cylinder Offsets

If the standard disk partition sizes and arrangement are not used, then it is necessary to calculate the partition cylinder offsets before the disk can be partitioned using **prep**. If the standard disk partition sizes and arrangement are being used, these calculations are unnecessary; the information is built into **prep**.

NOTE: If possible, use the standard partition sizes and arrangement. This reduces the possibility of error when building file systems.

Cylinder offsets must be calculated for each partition, according to the chosen partition sizes and arrangement. The offsets should be recorded on the template provided in Appendix A, "Disk Partition Templates."

The cylinder offset for the first partition in the partition arrangement is 0. The cylinder offset for the other partitions is calculated using this formula

$$CO = \lceil previous\_CO + (previous\_size \,/\, SPC) \rceil$$

CO is the cylinder offset of the partition of interest. It is always rounded up to the nearest integer.

previous_ CO is the cylinder offset of the partition preceding the partition of interest according to the partition arrangement.

previous_ size is the size in blocks (rounded up to the next cylinder boundary) of the partition preceding the partition of interest.

SPC is the number of sectors per cylinder for the disk that is to be partitioned.

NOTE: One sector holds 1024 bytes of data.

SPC is calculated as

$$\text{SPC} = \text{SPT} \times \text{TPC}$$

SPT is the number of sectors per track for the disk that is to be partitioned.

TPC is the number of tracks per cylinder for the particular disk that is to be partitioned.

Table 5-8 shows TPC and SPC values for commonly used disks and the total number of cylinders on the disk.

Table 5-8
Cylinder, Track, and Sector Information for Commonly Used Disks

Disk Drive Model Number	Size or Type	TPC	SPT	SPC Calculation	SPC	Total No. of Cylinders
9342 or 8136	80Mb	5	16	5×16	80	823
8146	160Mb	10	16	10×16	160	823
9344	300Mb	19	16	19×16	304	823
8858	340Mb	24	16	24×16	384	711
8155 or 8850	675Mb	40	16	40×16	640	843
8888	CDC 9772	16	42	16×42	672	1064
8884	Fujitsu 2351A	20	34	20×34	480	842
8889	Fujitsu 2361A	20	23	20×23	680	842
8820	SCSI 150Mb	9	18	9×18	162	966
8821	SCSI 300Mb	*	*	*	*	*

* Information not available at the time of printing.

NOTE: The total number of cylinders for each disk is not necessarily the number of cylinders available for disk partitions, because some cylinders may be flawed. The number of usable cylinders is indicated by **prep**. If the disk has already been partitioned, **diskpart** can also supply this information.

Similarly, the number of sectors used in the calculations for HSDP and MFP drives is one less than the number of sectors formatted on the disk. The extra sector is used for bad block forwarding.

See the *Gould RPU Disc Processor Diagnostic Reference Manual*, or contact your local field representative for more information.

WARNING: **prep** does not object to overlapping partitions, even if the overlap is partial. Therefore, cylinder offset calculations must be made with great care to eliminate unwanted overlapping. Otherwise, file system information may be destroyed.

Example

A 300Mb disk for a model 9344 disk drive is to be partitioned according to these requirements: the **a** and **b** partitions are to be the standard sizes, partition **c** is to cover all available space, and partitions **d**, **e**, **f**, **g**, and **h** are to be as nearly equal in size as possible. This arrangement is pictured in Figure 5-10.

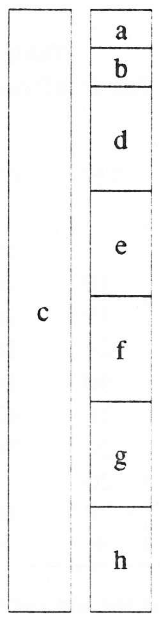


Figure 5-4. Sample Disk Partition Arrangement

As shown in Table 5-3, the standard sizes of the **a** and **b** partitions on a 300Mb disk for a model 9344 disk drive are 18240 and 20064 blocks, respectively. As shown in Table 5-8, the SPC value for such a disk is 304. Together, **a** and **b** will consume

$$(18240 / 304) + (20064 / 304) = 126$$

cylinders of the disk.

Also shown in Table 5-8, the disk to be partitioned has 823 cylinders available. There are, therefore, 697 cylinders left to be divided evenly between partitions **d** through **h**. To use the space on the disk efficiently, partition boundaries should align with cylinder boundaries. Calculate the number of cylinders for each of the five equally-sized partitions (rounded up) as follows:

$$(823 - 126) / 5 = 697 / 5 = 139$$

Hence, four of the partitions will consume 139 cylinders each; the fifth one will consume the remaining 138 cylinders. Each of the first four partitions will therefore contain

$$139 \times 304 = 42256$$

blocks.

The fifth partition takes up whatever space remains on the disk. The calculation of the cylinder offsets and the resulting partition sizes is summarized in Table 5-9.

Table 5-9
Sample Calculation of Cylinder Offsets

Partition	Partition Size (in blocks)	Cylinder Offset Calculation	Cylinder Offset
a	18240	0	0
b	20064	0 + (18240 / 304)	60
c	entire disk space	0	0
d	42256	60 + (20064 / 304)	126
e	42256	126 + (42256 / 304)	265
f	42256	265 + (42256 / 304)	404
g	42256	404 + (42256 / 304)	543
h	space left on disk	543 + (42256 / 304)	682

The exact sizes of the **h** and **c** partitions cannot, at this time, be determined. The sizes of these partitions can be determined when **prep** is used to partition the disk pack.

5.5.3 Partitioning Disk Packs with prep

The **prep** utility is used to partition disks and set swap partition eligibility. When partitioning disks, **prep** normally uses the standard partition sizes and arrangement illustrated in Tables 5-2, 5-3 and 5-4 in Section 5.5.1, ''Organizing Partitions and File Systems.'' These defaults can be changed by using **prep**. **prep** also handles the deallocation of bad disk blocks. See *prep*(8) for more information.

> NOTE: By default, **prep** automatically rounds up partition sizes to the next cylinder boundary. If you do not want rounding up, then specify the *absolute bounded partitions* when setting partitions sizes with **prep**. See *prep*(8) for details.
>
> For example, if you attempted to make the size of the **f** partition in Table 5-9 42000 blocks, the size would be rounded up to 42256 blocks unless absolute partitioning was specified. Note that if absolute partitioning was specified, the remaining 256 blocks would then be inaccessible.

The size of the last partition on the disk pack can be determined by specifying a very large number for the size of the last partition (for example, 1,000,000 blocks). If the amount of remaining space is less than this, the size specification will fail, with the amount of remaining space on the disk pack being displayed. The size of the last partition can then be set to the size of the remaining space on the disk pack.

Because the deallocation of bad blocks may vary between disk packs, the amount of usable space on each disk pack is variable. Also, some disk space is set aside for file system overhead and error maps. On the average, the percentage of usable disk space on a disk is about 81% of the total capacity. For each partition, the amount of space that can be used to hold file system data is about 94% of the partition size.

5.6 Making File Systems with newfs

To create a file system on a partition, use the **mkfs** or **newfs** utility. **newfs** is a front end to **mkfs**, is easier to use, and is recommended over **mkfs**.

Both **newfs** and **mkfs** perform these functions:

- Determine the number of blocks to set aside for inodes (structures containing information concerning individual data files) and data blocks in the file system.
- Create the **lost+found** directory used by **fsck** to reconnect dislocated files.

newfs invokes **mkfs**, automatically supplying the appropriate options. For example, to build a file system on the **d** partition on disk drive number **0**, enter

```
# /etc/newfs /dev/rdk0d
```

NOTE: File systems should never be built on disk partitions currently being used as swap or dump partitions.

WARNING: Never make file systems on overlapping partitions.

It is extremely important that file systems built on partitions *do not* overlap each other. For example, given the standard disk partition sizes and arrangement, if a file system is built on partition **c** and other file systems are built on any other partitions, there will be no overlapping file systems.

WARNING: There should always be a **lost+found** directory at the root of every file system created by **mkfs** or **newfs**. **fsck** depends on the existence of the **lost+found** directory. Do not remove it. If a **lost+found** directory is inadvertently removed, use **mklost+found** to recreate it.

For more information, see *mkfs*(8), *newfs*(8), *fsck*(8), and *mklost+found*(8).

5.7 Checking File System Consistency

5.7.1 Using fsck

Use **fsck** to check the consistency of a file system in these situations:

- After building a file system with **newfs** or **mkfs.**
- When in single-user mode (before bringing a system up multi-user). File system checks are especially important if the system was brought down in an uncontrolled manner (that is, if some processes would not die, the system hung or halted unexpectedly, etc.) In this situation, the root file system must first be checked using **preen**, as described in the following section.
- When hardware is changed.

Rebooting a system will automatically perform the necessary file system checks. See "Rebooting from Multi-user Mode" in Section 3.2.3, "Switching Between Modes." You can check any individual file system partition by entering

```
# /etc/fsck /dev/raw_partition_name
```

If errors are reported, see *fsck*(8).

WARNING: Never perform file system consistency checks on a mounted file system unless that system is the root file system in single-user mode.

5.7.2 Using preen

The root file system must always be checked immediately after successfully bringing up the system to single-user mode. Check the root file system by entering

```
# /etc/preen -o
```

If there are errors on the root file system, answer with a **y** to any prompts that ask if you want to correct or salvage the file system. If **preen** prints the message REBOOT UNIX, you must halt the system immediately after **preen** completes *without* synchronizing the disks (see *sync*(8)). **preen** corrects the information on the disk itself; if the disks are synchronized, the faulty in-core image will overwrite the corrections. To halt the system without synchronizing the disks, enter

```
# /etc/halt -n
```

After the system is halted, bring the system back up to single-user mode and again check the root file system using **preen**. There should be no errors.

When **preen** deems the root file system consistent, the other file systems may be checked using **fsck.**

The **/etc/rc** command automatically checks all file systems except the root file system by executing the command **/etc/preen –n**. You may wish to check file systems manually before bringing up the system multi-user. This can be done by entering

```
# /etc/preen -n
```

preen –n will check each file system (except the root) listed in **/etc/fstab**. It has the same effect as **/etc/fsck –p** except that it can be much faster because it tries to keep one instance of **fsck** running on each disk drive until all the file systems have been checked.

5.7.3 Recovering Lost Files

All dislocated files found by **fsck** or **preen** that cannot be relinked to their proper place are put into the appropriate **lost+found** directory. The file's name is the previous inode number of the file, and the owner will be the original owner. The administrator should notify the owners of any files placed in the **lost+found** directory. The owner can then decide what to do with the file.

The **lost+found** directory is re-created automatically at the root of a file system by **newfs** and **mkfs**. If necessary, this directory can be created with **mklost+found.**

See *fsck*(8), *newfs*(8), *mkfs*(8), *mklost+found*(8), and *preen*(8) for more information.

5.8 Setting Up /etc/fstab

After determining your disk partition organization, you must inform the system how the partitions are to be used by modifying the file **/etc/fstab**. This file contains an entry for each file system and each swap partition.

This is the content of the distributed **fstab** file:

```
/dev/dk0a / 4.3 rw,noquota 1 1
# /dev/dk0b is the default swap partition
/dev/dk0d /usr 4.3 rw,noquota 1 2
```

The second line begins with a pound sign (#), which indicates that the entire line is a comment rather than an entry.

An entry in the file consists of six fields separated by white space. The meaning of each of those fields is described in the following list.

Field 1: Partition name

The full pathname of the device entry for the partition.

Field 2: File system root

The full pathname of the directory on which the file system is mounted, if a file system has been built on the partition. When the system comes up in multi-user mode, the file system is mounted on this directory, allowing users to access it.

For swap partitions, this field is the same as the partition name.

Field 3: File system type

The type of the file system to mount. It will be one of the following:

4.3	A local disk Berkeley file system
nfs	A remote network (Sun) file system

Field 4: File system mode

Indicates the type and access modes of the partition entry. The mode is a combination of one or more of these:

rw	The partition is a file system that can be read and written.
ro	The partition is a file system that can only be read; it is protected from writing.
swap	The partition is a swap partition.
xx	The entry is to be ignored.
noquota	There is no limit to diskspace.
suid	Set-UID execution is allowed.
nosuid	Set-UID execution is not allowed.

Field 5: Dump frequency

Indicates the frequency with which file systems are backed up; this is used by the **dump** program. The dump frequency is given in number of days. The most commonly used entries are

0	Backups are not to be done by **dump** for this file system.
1	Backups are to be done daily by **dump** for this file system.

If the field is empty, it is assumed to be **0**. Normally, all file systems have **1** in this field, except for **/tmp**, which has **0**.

For swap partitions, this field is empty. See *dump*(8) for more information.

Field 6: **fsck** pass number

The order in which a file system is to be checked when using **fsck** with the **–p** option. If the field is left empty or contains **0**, the entry is not checked by **fsck**.

The root file system, /, must have a **1** in this field so that it is checked first, and it must be the only file system with a **1**. For swap partitions, this field is empty. Other file systems should have numbers greater than **1**. See *fstab*(4) and *fsck*(8).

NOTE: The default swap partition (the **b** partition on the bootpack) should not be included in **/etc/fstab**; otherwise, an error message is generated on reboot.

NOTE: The default swap partition is normally used for swapping. It is usually not used for anything else. Unless swapping is disabled, the default swap partition is always used for swapping.

NOTE: The order in which entries appear in the file system table is the order in which file systems are mounted. The file system table must be in the correct order for your system to work properly. If a file system directory is located underneath a second file system directory in the directory tree hierarchy, then the second file system has to be mounted first.

5.8.1 For Two 80Mb Disks

If the system is being set up to use two 80Mb disks, with the partition organization described in the subsection, "Recommended Disk Partition Organizations," in Section 5.5.1 "Organizing Partitions and File Systems," then edit your file system table using the **ed**, **ex**, or **vi** editor. See *ed*(1), *ex*(1) and *vi*(1).

NOTE: If the **/usr** file system has not been built or is not mounted, only the **ed** editor is available.

The **fstab** file should look like this:

```
/dev/dk0a / 4.3 rw,noquota 1 1
# /dev/dk0b is the default swap partition
/dev/dk0d /mnt 4.3 rw,noquota 1 2
/dev/dk1c /usr 4.3 rw,noquota 1 2
```

The **1** in the fifth field of the / (root), **/mnt** and **/usr**, entries specifies that those file systems are to be backed up daily by the **dump** utility. The sixth field of each entry indicates that, when checking file systems with **fsck -p**, this sequence will be followed: first, the root file system will be checked; then, the **/mnt** and **/usr** file systems will be checked in parallel.

The system described in this section will not be able to hold the system source. If you want source, you will need either a third 80Mb pack or a 300Mb pack.

5.8.2 For a Single 300Mb Disk

If the system is being set up to use a single 300Mb disk, with the partition organization described in the subsection, "Recommended Disk Partition Organizations," in Section 5.5.1 "Organizing Partitions and File Systems," then edit your file system table, using **ed**, **ex**, or **vi** to look like this:

```
/dev/dk0a / 4.3 rw,noquota 1 1
# /dev/dk0b is the default swap partition
/dev/dk0d /usr 4.3 rw,noquota 1 2
/dev/dk0e /mnt 4.3 rw,noquota 1 3
/dev/dk0f /usr/src 4.3 rw,noquota 1 4
```

If you do not have source, **/tmp** can occupy the **f** partition. If source is to be included, **/tmp** will have to be a dismounted directory on the **a** partition. If you want to include the entire source tree, plus object files and executables (this requires 96000 kilobytes of disk space), a **g** partition will be required. In that case, both **/mnt** and **/tmp** will have to be made directories under root.

5.8.3 For Two 300Mb Disks

If the system is being set up to use two 300Mb disks, with the partition organization described in the subsection, "Recommended Disk Partition Organizations," in Section 5.5.1 "Organizing Partitions and File Systems," then edit your file system table, using **ed**, **ex**, or **vi** to look like this:

```
/dev/dk0a / 4.3 rw,noquota 1 1
# /dev/dk0b is the default swap partition
/dev/dk0d /usr 4.3 rw,noquota 1 3
/dev/dk0g /usr/src 4.3 rw,noquota 1 4
/dev/dk1a /tmp 4.3 rw,noquota 1 2
/dev/dk1b /dev/dk1b swap swap
/dev/dk1e /mnt 4.3 rw,noquota 1 3
```

The inclusion of a second 300Mb pack makes it possible to add a second swap partition for interleaved swapping with the default swap partition. In addition, **/tmp** is made into a separate file system and is moved, along with **/mnt**, onto the new pack on a new branch. The source is added to an **f** or **g** partition, depending on whether you want executables and object files with it.

NOTE: All partitions used as swap partitions must be specified in the **CONFIGURATION** file; see Chapter 7, ''Reconfiguring the System.''

5.9 Configuring and Maintaining Disk Partition Mirror Sets

The *mirrored disk* feature allows the replication of data from one disk partition onto one or more other disk partitions. This section describes the system administrator's responsibilities and procedures for:

- Configuring mirror sets
- Entering information in the *bad block list*
- Setting *read distribution modes* for distributing reads among the mirror set partitions
- Removing mirror sets
- Reconfiguring mirror sets after a crash
- Referencing log messages

5.9.1 Configuring Mirror Sets

Configuring a *mirror set* is the process of allocating one or more secondary partitions to be mirrors of a single primary partition. It can be done at any time after the system is booted—before or after mounting a file system on the primary partition—as long as the secondary partitions are not mounted. The recommended method is to configure mirror sets first and then to mount the primary partitions.

When a mirror set is configured successfully, the kernel mirror table is updated. After a mirror set is configured, all writes to the primary partition are propagated to the secondary partitions. Read requests are allowed on any partition in a mirror set and can be distributed among the available partitions.

Configure mirror sets with the **mirror** system administrator command. Before using this command, check that:

- You have superuser privileges. Otherwise you can not use the command.
- Disk partitions in a mirror set are the same size. If the primary partition is larger than any of the secondaries, the command will fail. Secondary partitions may be larger than the primary, but this is strongly discouraged for the following reasons:

  a. The additional space on the secondary is wasted.

  b. Should it become necessary to use the secondary as a primary, the original (smaller) primary cannot be used as a secondary.

  Use the **diskpart** system administrator command to get the size of a disk partition.

Configure a mirror set with one secondary partition by entering

```
# /etc/mirror /dev/primary_partition /dev/secondary_partition
```

For example, to configure a mirror set with **/dev/dk01d** as the primary partition and **/dev/dk03d** as the secondary, enter

```
# /etc/mirror /dev/dk01d  /dev/dk03d
```

You can also configure a mirror set with up to three secondary partitions. See the *mirror*(8) man page for syntax and examples.

By default the **mirror** command will synchronize the partitions in the mirror set. Because synchronization is time-consuming, execute the **mirror** command in the background if synchronization is necessary. Under ideal circumstances, synchronization is only necessary when the mirror set is configured for the first time.

> WARNING: Do not synchronize from a primary partition with known bad blocks.

Use the **–q** option to configure a mirror set without synchronization. Enter

```
# /etc/mirror -q /dev/disk_partition /dev/disk_partition
```

See the *mirror*(8) man page for a description of disk mirroring and for details on using all of the options of the **mirror** command.

5.9.2 Entering Bad Block Information

The kernel maintains a *bad block list* for every partition that is a member of a mirror set. This is not the same as the diagnostic map (DMAP) that is maintained on the disk. The mirrored disk system adds bad block information to the list; the administrator is responsible for maintaining this list after reboots.

> WARNING: The presence of bad blocks on a disk partition indicates that the disk should undergo maintenance. Bad block lists do not fix bad disks, they merely extend the time until total failure. Maintain the disks with **prep**. See the *prep*(8) man page for details.

How the System Adds Bad Block Information

When a write error occurs on any partition in a mirror set, the mirrored disk system adds the affected blocks to the bad block list for that partition and retries the operation on another partition. Each entry in the list consists of a starting block number and the count of how many bad blocks should be added. Together these fields represent a range of bad blocks. Whenever bad blocks are added to the list, a message is sent to the system console.

How the System Administrator Adds Bad Block Information

Because bad block lists are not maintained across reboots, you must re-enter bad block lists manually with the **–b** option of the **mirror** command whenever you configure mirror sets.

1. First, check the log messages to see if any bad blocks were logged. Log messages are stored in the file **/usr/adm/messages**. **grep** this file for "mirror:" to obtain all mirror log messages. Then, search for any bad block entries.

2. If you find any bad block entries, use the **–b** option to enter bad blocks for a particular partition *after* the mirror set is configured. Enter

   ```
   # /etc/mirror -b /dev/disk_partition starting_block block_count
   ```

   where the *starting_block* is the number of the first block on a disk partition to be added from the bad block list and the *block_count* is the number of bad blocks to be added.

   For example, to add blocks from 1600 through 1607 to the bad block list for partition **/dev/dk03d**, enter

   ```
   # /etc/mirror -b /dev/dk03d 1600 8
   ```

   Because the count defaults to eight if no value is supplied, the same blocks can be added by entering

   ```
   # /etc/mirror -b /dev/dk03d 1600
   ```

3. If you made a mistake in step 2, you can use the **–c** option to remove any blocks that were accidentally entered in the bad block list. Enter

   ```
   # /etc/mirror -c /dev/disk_partition starting_block block_count
   ```

   where *block_count* is the number of bad blocks to be removed. For example, to remove blocks 1600 to 1607 from the bad block list of the above partition, enter

   ```
   # /etc/mirror -c /dev/dk03d 1600 8
   ```

   Because the count defaults to eight, you do not have to enter it. See the *mirror*(8) man page for additional information.

5.9.3 Setting Read Distribution Mode

Read distribution improves read throughput by distributing reads among the partitions in a mirror set. You can select one of the following read distribution modes for each mirror set:

Mode 0: Read from primary
: All reads are issued to the primary partition.

Mode 1: Read next partition
: Reads are issued sequentially to all of the partitions in the mirror set. That is, the first read is issued to the first partition, the second is issued to the second partition, and so on until all of the partitions have serviced a read request. The reads then start over again with the first partition.

Mode 2: Read burst
: Reads are issued sequentially to all of the partitions in the mirror set, but each partition services several reads before they are issued to the next partition.

Mode 3: Read minimum queue
: Reads are issued to the partition that has the fewest I/O requests queued.

During the synchronization process, the default mode is 0. When synchronization completes, the mode is set to 3, which optimizes read throughput.

In all read distribution modes, if a read fails on the first partition it is issued to, it is retried sequentially until it succeeds or all of the partitions in the mirror set have been tried. The read fails only if it fails on every partition in the mirror set.

Use the **–d** option of the **mirror** command to select the read distribution mode by entering:

```
# /etc/mirror -d /dev/partition mode
```

where *mode* is an integer specifying the read distribution mode. For example, to set the read distribution mode to 1 (*read next partition*) for the mirror set that includes **/dev/dk02d**, enter

```
# /etc/mirror -d /dev/dk02d 1
```

To set the read distribution mode to 2 (*read burst*) you must supply the *burst size* argument. Enter:

```
# /etc/mirror -d /dev/partition 2 burst_size
```

where *burst_size* is an integer that specifies the number of consecutive reads on each partition. For example, to set the read distribution mode to 2 (*read burst*) with a burst size of 8 for the mirror set that includes /dev/dk01a, enter

```
# /etc/mirror -d /dev/dk01a 2 8
```

5.9.4 Removing Mirror Sets

Removing a mirror set is the equivalent of "turning off" mirroring for a particular primary partition. When you specify the **–r** option, the mirror set containing the given partition is removed from the mirror table. A check is made to ensure that the primary partition of this mirror set is not mounted. This check is important, because if mirroring were turned off before the primary was unmounted, writes in progress to the primary partition would not be propagated to the secondary partitions, and the mirror set would no longer be synchronized. If this check fails, the command will fail. This check can be overridden by specifying the **–f** option.

Use the **–r** option of the **mirror** command to remove a mirror set. Enter

```
# /etc/mirror -r /dev/disk_partition
```

where *disk_partition* is any member of the mirror set which you want to remove.

For example, to remove the mirror set that consists of **/dev/dk01d**, **/dev/dk02d**, and **/dev/dk03d**, enter

```
# /etc/mirror -r /dev/dk01d
```

You could use **/dev/dk02d** or **/dev/dk03d** instead of **/dev/dk01d** with the same results. See the *mirror*(8) man page for additional information.

5.9.5 Reconfiguring Mirror Sets After a Crash

When a system crashes, you can follow three basic approaches for reconfiguring mirror sets. These approaches differ in the order that the **fsck**, **mirror**, and **mount** system administrator commands are executed. Choose an approach based upon:

- The time available to bring up the system
- The importance of data integrity
- The current state of the mirror sets

Time-critical Approach

Use this approach if you want to minimize the time required to reconfigure mirror sets and you believe that the mirror sets are synchronized.

1. Configure mirror sets using the **mirror** command with the **–q** option.
2. Set the read distribution mode to 0 (*read from primary*).

3. Execute **fsck** on the primary partition.
4. Reset the read distribution mode.
5. Mount the primary partition using **mount**.

Time and Data-critical Approach

Use this approach if you want to minimize the time required to reconfigure mirror sets, but you suspect that synchronization is required.

1. Execute **fsck** on the primary partition. If it fails, continue executing **fsck** on each partition in the mirror set until one of them succeeds. Use the good partition as the primary partition for this mirror set.

   If **fsck** fails for every partition in a mirror set, then either go to a backup tape, or choose the partition that you think is best for the primary partition.
2. Configure mirror sets with the **mirror** command. Do not specify the **–q** option.
3. After synchronization begins, mount the primary partition with the **mount** command. Note that if you switch to a new primary, you must update **/etc/fstab** to reflect this information.

Data-critical Approach

If data integrity is critical, use this approach.

1. Execute **fsck** on all of the partitions in the mirror set in parallel.
2. Choose a good partition for the primary partition.
3. Configure and synchronize the partitions pairwise. That is, if you have two secondary partitions in a mirror set, you should:
   a. Configure and synchronize a mirror subset with the primary and one secondary partition.
   b. Remove this mirror subset.
   c. Configure and synchronize a mirror subset with the primary and the other secondary partition.
   d. Remove this mirror subset.
4. Configure the mirror set after the above synchronizations have completed. Use the **–q** option of the **mirror** command.
5. Mount the primary partition with the **mount** command.

See the man pages for *fsck*(8), *mirror*(8), and *mount*(8) for more information.

5.9.6 Mirrored Disk Log Messages

The mirrored disk system logs informative messages via the **syslog** kernel subroutine. The following events are logged:

- A mirror set is added or removed
- Bad blocks are added by the system operator
- Bad blocks are deleted by the system operator
- Bad blocks are dynamically added
- The bad block list overflows for some partition

Messages, which are sent via the **syslog** kernel subroutine, will have the following format:

```
mirror: informative_message
```

The following is a list of possible log messages:

```
mirror: add set /dev/dk01d /dev/dk02d /dev/dk03d
mirror: remove set /dev/dk01d
mirror: /dev/dk03d add bad blocks 2400 to 2407
mirror: /dev/dk01d remove bad blocks 4000 to 4004
mirror: /dev/dk03d bad block list overflow blocks 50 to 57
mirror: device /dev/dk03d could not be opened
mirror: /dev/dk01d biodone can't read buf
```

These messages are logged with priority LOG_WARNING. See the *syslog(8)* man page for more detail.

5.10 Mounting File Systems with mount

Mounting a file system is the process of associating the name of a file system with a disk partition. The root directory of the file system must already exist, and the file system must have been created previously by **newfs** or **mkfs**. Create a directory using the **mkdir** command. Change directory into the newly created directory and enter the **mount** command. After a file system is mounted, users can access the data contained in the file system.

> NOTE: Mounting a file system on a directory will leave all files previously underneath the directory inaccessible. Mount a file system only on an empty directory.

Use the **mount** command to **mount** a file system. To use this command, you must have superuser privileges and there should be no processes whose current working directory is the file system or directory on which you plan to mount the file system. Doing so will cause a failure with the message, `Device busy`. When a file system is successfully mounted, the mount table kept in the file **/etc/mtab** is updated.

Mount a single file system by entering

```
# cd /
# /etc/mount /dev/block_special_disk_partition file_system_name
```

For example, to mount the **/usr** file system, enter

```
# cd /
# /etc/mount /dev/dk0d /usr
```

To mount all file systems listed in **/etc/fstab, enter**

```
# cd /
# /etc/mount -a
```

This option tells **mount** to mount the file systems listed in **/etc/fstab** in the order specified there.

See *mount*(8), *umount*(8), Section 5.11, ''Dismounting File Systems with **umount**,'' and Section 5.8, ''Setting Up **/etc/fstab**,'' for more information.

> WARNING: If you will be mounting file systems while in single-user mode, you should first ensure that the mount table is cleared. To do this, execute the commands
>
> ```
> # /etc/umount -a
> # /etc/mount
> ```
>
> The first command dismounts all file systems known to the system through **/etc/fstab**. The second command checks to see whether any file systems remain mounted. If any file systems are reported as mounted by the second command, dismount them individually using **umount**.

5.11 Dismounting File Systems with umount

Dismounting a file system is the process of disassociating the file system's directory name from a disk partition, thereby making the file system unavailable to users (that is, user processes). An attempt to dismount a file system will fail if there is any process whose current working directory is within the file system being dismounted.

Dismounting a file system is accomplished using the **umount** command as superuser. When a file system is successfully dismounted, the mount table in the file **/etc/mtab** is updated.

Dismount a single file system by entering

```
# cd /
# /etc/umount /dev/block_special_disk_partition
```

To dismount all of the file systems in **/etc/fstab** except the root file system (which can never be dismounted), enter

```
# cd /
# /etc/umount -a
```

Verify that no file systems other than root are still mounted by executing the command

```
# /etc/mount
```

If any mounted file systems are reported in response to this command, dismount each one individually. If you receive the message

```
/dev/partition_name: Mount device busy
```

in response to a dismount command, there is an active process on the associated file system that must be killed before **umount** can succeed.

> NOTE: It is very difficult to identify a process on a particular file system. The best course of action is to send a system-wide message (see *wall*(8)) asking users to change directory out of the file system.
>
> WARNING: If all attempts to dismount a file system fail and you are forced to halt the system without dismounting a file system, be sure to check that file system with **fsck** before bringing the system back up to multi-user mode.

See *umount*(8), *mount*(8), Section 5.10, ''Mounting File Systems with **mount**,'' and Section 5.11, ''Dismounting File Systems with **umount**,'' for more information.

6 Source Code

Source code for UTX/32 is included only with source distributions of the system. If you do not have a source distribution, you can ignore this chapter.

Source code for UTX/32 consists of user source and system source. *User source* is the source for all the user and administrative utilities and for the system libraries. *System source* consists of source files that are necessary to generate a UTX/32 kernel from scratch and source files for stand-alone programs.

All source is kept in a directory named **/usr/src/src**, which is often mounted as a separate file system. See Figure 6-1 "The UTX/32 Source Tree" for an overview of the upper layers of the source directory tree structure.

6.1 Storing BSD Source

BSD user source is kept in various subdirectories of **/usr/src/src**, depending (in most cases) on where the executable is installed. For each directory containing executables, there is a corresponding source directory in **/usr/src/src**. Within each of these directories is a subdirectory for each command. For example, the executable for the command **ls** is in **/bin**, so the source for **ls** is located in **/usr/src/src/bin/ls**.

UTX/32 system source is kept in a subtree of the BSD source directory called **/usr/src/src/sys**.

6.2 Storing System V Source

System V user source is kept in various subdirectories of **/usr/src/5src**. Unlike the BSD source tree, many utilities compiled from only one source file are found in the same directory, **/usr/src/5src/cmd**. Compilation of these commands is controlled by a single makefile.

For example, the executable for the command **ls** is in the file **/usr/src/5src/cmd/ls.c**, not in a directory of its own. However, a few commands, such as **/usr/5bin/cc**, have their own directories.

6.3 Reading BSD and System V Source Code from the Source Tape

The source code for UTX/32 comes on two source tapes that are distinct from the system binary tapes. If you have room on your system and wish to keep the system source online, the source may be installed by following these steps:

1. If your system is not in single-user mode, become superuser. See *su*(1).
2. Mount the first source distribution tape on a tape drive that can read 1600-bpi tapes.

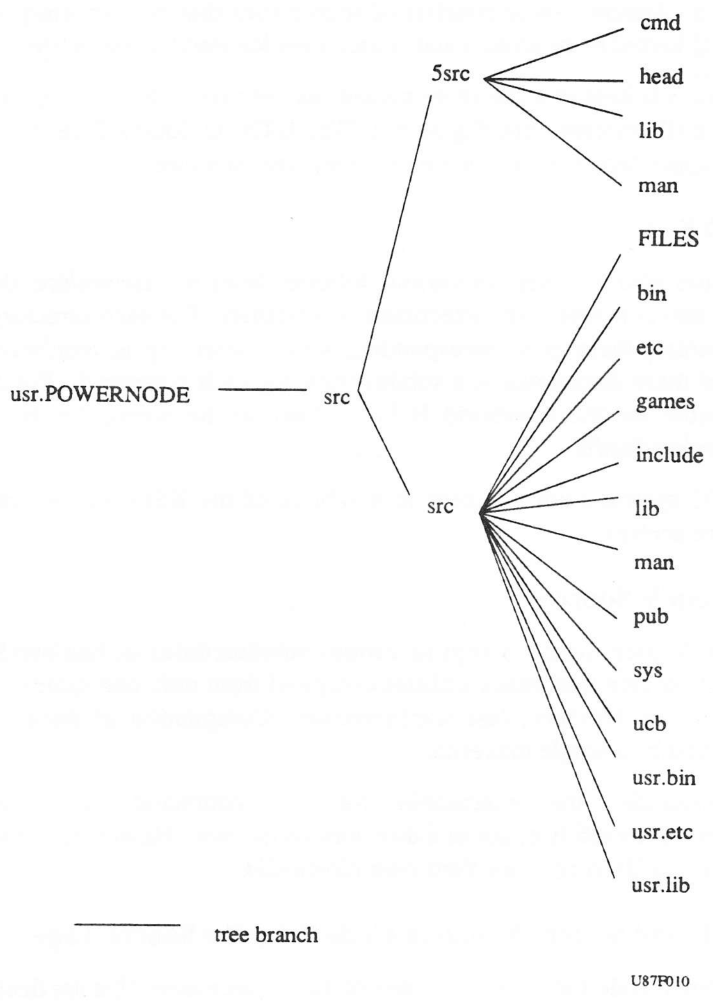


Figure 6-1. The UTX/32 Source Tree

3. Make sure the target directory **/usr/src** exists and is empty. At least 57000Kb of disk space is needed to hold the contents of the UTX/32 source tape if both the BSD and System V sources are to be restored. 123000Kb of space is necessary to contain the source if it is completely compiled. The source tape is not a **tar** tape, but a dump tape. There are two tapes with one dump image spread over the two tapes.

4. If **/usr/src** is to be mounted as a separate file system,

   a. Make sure the disk that is to hold the **/usr/src** file system has been **prepped**. (See Section 5.5.3, ''Partitioning Disk Packs with **prep**.'')

   b. Make sure a file system for **/usr/src** exists. (See Section 5.6, ''Making File Systems with **newfs**.'')

   c. Mount the file system designated for containing source as **/usr/src**:

   ```
   # fsck /dev/rdknn
   # mount /dev/dknn /usr/src
   ```

   dk*nn* is the device special file for the file system.

5. Change into the **/usr/src** directory:

   ```
   # cd /usr/src/src
   ```

6. The BSD and System V source are contained in one dump image. If you want to restore both environments, type

   ```
   # restore r
   ```

   If you want only the BSD environment, type

   ```
   # restore r src
   ```

   If you want only the System V environment, type

   ```
   # restore r 5src
   ```

   NOTE: **restore** creates the file **./restoresymtable**. You may want to remove it by using the **rm** command.

7. Dismount the source tape.

Since the system source spans two tapes, you will need to dismount the first tape and mount the second, depending on which source you want (BSD source, System V source, or both). Because many programs expect the kernel source to be in **/sys**. If you modify kernel source, that directory must be set up as a symbolic link to **/usr/src/src/sys**. Follow these steps:

1. Save the existing kernel directory **/usr/sys** (the directory to which **/sys** is linked) that was read in from the binary-only distribution. Do this by entering

   ```
   # mv /usr/sys /usr/sys.old
   ```

   The source for all object files, include files, and data files that were in **/usr/sys** in the binary-only distribution are present in **/usr/src/src/sys**.

   WARNING: If you have edited any files in the old **/sys** tree, you may want to replace the new versions of those files with your edited ones.

   If you do not care to save the existing contents of the **/sys** directory, remove them by entering the command

   ```
   # rm -rf /usr/sys
   ```

2. Relink **/sys** to the new system directory by entering

   ```
   # rm /sys
   # ln -s ./usr/src/src/sys /sys
   ```

For more information, see *fsck*(8), *mount*(8), *restore*(8), and *ln*(1).

6.4 Source Tree Permissions

As distributed, the source tree is owned by root and has a group ID of 10. To prevent inadvertent modification only **root** has write privileges. The administrator should select the group and permissions that best suit the environment. The standard choices are to

Prevent modification and compilation except by **root**:

Directory permissions	755
File permissions	644 or 640

Prevent modification except by **root**, but allow compilation by a certain group:

Directory permissions	775
File permissions	644 or 640

The administrator could use a combination of the following commands to establish the appropriate environment.

```
chown -R owner.group /usr/src/src
chmod -R 644 /usr/src/src
find /usr/src/src -type d -exec chmod 775 {};
```

6.5 Recompiling BSD and System V User Programs

If you have installed the source code according to the instructions in section 6.3, the source for BSD and System V user programs is contained in **/usr/src/src** and **/usr/src/5src** respectively. For BSD source, the **/usr/src/src** contains a subdirectory for each command. Within each of these subdirectories, each command has its own directory. For System V source, the **/usr/src/5src** does not contain a subdirectory for each command. The source for most System V utilities is found at the same directory in **/usr/src/5src/cmd.**

Each of the command directories contains a makefile. Use these makefiles to control the recompilation of the user utilities. Note that BSD and System V use different **make** commands. For more information about how makefiles are used, see the appropriate *make*(1) manual pages and UTX/32 supplementary documents entitled *Make—A Program for Maintaining Computer Programs [BSD]* and *Augmented Version of Make [SV]*.

To compile System V utilities, you must be in the System V environment so that the System V versions of **make** and **cc** are invoked. If you logged into your system under the BSD environment, you can switch to the System V environment by entering

```
# sv
```

This command gives you a System V shell by executing **/usr/5bin/sh** and redefining your path variable so that **/usr/5bin**, where the System V commands reside, is searched first. See *sv*(1) for more information.

Alternatively, you can run the System V **make** command without changing your working environment by entering

```
# sv make arguments
```

NOTE: Some makefiles call programs in the current directory and assume that the current directory, **.** (dot), is in your PATH variable. To verify that this is the case, type

```
# echo $PATH
```

6.5.1 Recompiling All of the BSD and System V User Source Code

To recompile code that is part of the distributed system, you must be superuser. The source for all BSD user commands and utilities can be recompiled by changing directory to the top of the user source code tree and entering the command **make**.

To recompile all of the BSD user source code enter these commands:

```
# cd /usr/src/src
# make clean
# make
```

make assumes that no object files that could interfere with the **make** dependencies remain from some previous attempt to compile the code. To be sure that this assumption is correct, enter **make clean** before entering **make**.

Note that **make** will not recompile the kernel or standalone utilities. The invocation of **make** recompiles every user utility in each subdirectory of **/usr/src/src** *except* the kernel and standalone utilities in **/usr/src/src/sys**. Also note that recompiling all of the BSD user source code takes a long time.

To recompile all of the System V user source code enter these commands:

```
# cd /usr/src/5src
# sv make clobber
# sv make
```

The invocation of sv **make** recompiles the user utilities in each subdirectory of **/usr/src/5src**. (No system source resides in the System V source tree.) To ensure that no object files remain from some previous attempt to compile the code, enter sv **make clobber** before entering sv **make**.

NOTE: This note is of interest to UTX/32 customers, outside the US, who do not receive **crypt** tapes. In **/usr/src/5src/cmd** two modules will not make because they use the **crypt** routine that is not present in **/usr/5lib/libc.a**.

```
          cc -O -n makekey.c -o makekey
Undefined:
_crypt
          cc -O -n newgrp.c -o newgrp
Undefined:
_crypt
```

Several modules generate compiler warning messages that can be ignored. Most of the messages are of the following type:

```
warning: function contains both return and return(expr)
warning: function has a declared type but never
          returns a value
```

The modules are

5src/cmd/rje	**src/ucb/ex**
5src/cmd/make	**src/ucb/finger**
src/bin/awk	**src/ucb/more**
src/bin/csh	**src/ucb/more**
src/bin/df	**src/ucb/pascal**
src/bin/expr	**src/ucb/sccs**
src/bin/make	**src/ucb/talk**
src/bin/mail	**src/ucb/window**
src/bin/sh	**src/usr.bin/dc**
src/etc/fsck	**src/usr.bin/diction**
src/usr.bin/csplit	**src/usr.bin/f77/src/fsplit**
src/etc/icheck	**src/usr.bin/find**
src/etc/route	**src/usr.bin/find**
src/etc/savecore	**src/usr.bin/learn**
src/etc/timed	**src/usr.bin/lex**
src/etc/acs	**src/usr.bin/lint**
src/games/adventure	**src/usr.bin/nroff**
src/games/arithmetic	**src/usr.bin/refer**
src/games/backgammon	**src/usr.bin/span**
src/games/mille	**src/usr.bin/struct**
src/games/monop	**src/usr.bin/tbl**
src/games/quiz	**src/usr.bin/yacc**
src/games/sail	
src/games/trek	
src/ucb/Mail	
src/ucb/ctags	

6.5.2 Recompiling All BSD and System V Executables for a Specific Directory

To recompile code that is part of the distributed system, you must be superuser. To recompile code for an entire directory of executables (for example, **/bin** and **/etc**), change directory to the corresponding subdirectory in the source tree and type **make**.

For example, to recompile every BSD user utility in the **usr.bin** subdirectory of **/usr/src/src** enter these commands:

```
# cd /usr/src/src/usr.bin
# make clean
# make
```

For example, to recompile every user utility in the **cmd** subdirectory of **/usr/src/5src** enter these commands:

```
# cd /usr/src/5src/cmd
# sv make clobber
# sv make
```

6.5.3 Recompiling a Particular BSD or System V Command

To recompile code for a single utility that is part of the distributed system, you must be superuser. To recompile a single command, change directory to the subdirectory that (in most cases) bears that command name. Then type the command **make**. For example, to compile the command **ls** enter these commands:

```
# cd /usr/src/src/bin/ls
# make clean
# make
```

To recompile a single System V command, change directory to **/usr/src/5src/cmd**, then enter the **make** command, specifying the name of the program you want to recompile. For example, to compile the command **ls** enter these commands:

```
# cd /usr/src/5src/cmd
# sv make clobber
# sv make ls
```

6.6 Installing BSD and System V User Programs

When commands are recompiled, the executables are automatically created in the source directory. To make these executables available for general use, they must be installed in the executable directory.

To do this, the following procedure is recommended.

1. Become superuser.
2. It is suggested that you make backup copies of the existing executables that you will replace. Keep them until the new executables have been verified to work correctly. If you are replacing a large number of commands, you may want to make a subdirectory within the executable directory to hold the old copies.

3. To install all BSD and System V executables, which you might want to do if you have just recompiled the entire source tree, change directory to the top of the **/usr/src** tree and enter these commands:

```
# cd src
# make install
```

This will install the BSD executables. Install the System V executables by typing

```
# cd ../5src
# sv make install
```

These commands copy the newly compiled executables into the default destination executable directory defined in the makefile. This will be a subdirectory of /, the default root directory.

To copy the commands to a different destination directory, you must redefine the appropriate makefile constant on the **make** command line. For example, to install all BSD and System V commands in a subdirectory named **/test**, change to the top of the **/usr/src** tree and enter these commands:

```
# cd src
# make install DESTDIR=/test
# cd ../5src
# sv make install ROOT=/test
```

4. To install smaller collections of BSD executables, change to the appropriate source directory. This will be either one of the source subdirectories or the particular directory containing the code for a single command. Compile the source code, if necessary, then enter these commands:

```
# make install
```

To install System V executables, type

```
# sv install -f  destination_dir program
```

6.7 Recompiling the Kernel

If you have made changes to the kernel source code, you will need to recompile the kernel.

This release of UTX/32 is distributed with two bootable kernels. One kernel has a default virtual address space (virtual size) of 400000 (hex) and is referred to as a 4Mb kernel. The other kernel has a default virtual address space (virtual size) of 200000 (hex) and is refered to as a 2Mb kernel. Normally, sites will use the 4Mb kernel to support more users, but some run a smaller 2 Mb kernel to provide more memory for user programs. The recompilation procedure allows you to make this choice.

To recompile the kernel, follow these steps:

1. Become superuser
2. Change to the **/sys/obj** directory. **/sys** should be a symbolic link to wherever you keep your source, usually **/usr/src/src/sys.**
3. Enter the command

   ```
   # make
   ```

   This will cause those source files that have been modified since the last time the kernel was made to be compiled. When all compilations complete successfully, a new **unix** is automatically linked.

   If there are also changes to the **CONFIGURATION** file, entering the **make** command will also cause the processing of the new configuration information. (See Section 7.10, ''Generating a Reconfigured Kernel.'')

   The distributed kernel source files contain defines that will create a 4Mb kernel. This default can be overridden on the **make** command line by specifying **twomeg** (to make a 2Mb kernel).

   NOTE: If either the **twomeg** or **fourmeg** arguments are specified, the default virtual kernel size changes to that argument and will be the default for all subsequent makes of the kernel until the alternative specification is made.

   To specify a 2Mb kernel, enter

   ```
   # make twomeg
   ```

   To specify a 4Mb kernel, enter

   ```
   # make fourmeg
   ```

   NOTE: The capability to reconfigure and make both 2Mb and 4Mb kernels is also available on binary distributions.

6.8 Recompiling and Installing Standalone Programs

To recompile the standalone diskboot programs (**tboot**, **dkboot**, and **sdboot**) you must be superuser. Enter these commands:

```
# cd /usr/src/src/sys/stand
# make
```

To install the standalone boot programs, follow these instructions.

1. **tboot** is the tape boot program. It is copied only to a tape device from which you plan to boot. To install it, mount the tape on the tape drive, make sure the tape drive is on line, then type

   ```
   # cd boot
   # dd if=tboot of=/dev/rmtn bs=8k
   ```

2. **dkboot** is the UDP diskboot program. It is placed on the superblock of the **a** partition of UDP disk from which you plan to boot. To install it, type

   ```
   # cd boot
   # newfs -B /dev/rdkna
   ```

3. **sdboot** is the SCSI diskboot program. It is placed on the superblock of the **a** partition of SCSI disk from which you plan to boot. To install it, type

   ```
   # cd boot
   # newfs -B /dev/rsdna
   ```

   NOTE: Do not remove the diskboot programs from the source directory. Some programs, such as **newfs**, look for these programs in the directory **/sys/stand/boot**, which will normally point to the source directory **/usr/src/src/sys/stand/boot**. See *dd*(1) for more information.

6.9 Installing the Kernel

Installing a new kernel involves placing the **unix** that was created in the /sys/**obj** directory into the root directory, /. You do not have to do this manually. The makefile in the /sys/**obj** directory takes care of it for you. Follow this procedure:

1. Make sure the system is in single-user mode.
2. Change into the /sys/**obj** directory.
3. Make a backup of **unix** so that the old kernel can be restored if necessary:

   ```
   # cp /unix /unix.bak
   ```

4. Enter the command

```
# make install
```

to copy **unix** into the destination directory, which is by default the root directory, /.

5. Check that the virtual memory link **/vmunix** points to **/unix**. Do this by typing

   ```
   # ls -l /vmunix
   ```

   If the link is missing, it can be remade by typing

   ```
   # cd /
   # ln -s /unix /vmunix
   ```

   NOTE: While it is possible to install a new kernel in multi-user mode, this practice is not recommended. During the time the old kernel is executing but the new object image is in place, utilities (such as **ps**) that look into the kernel for data will fail because of the mismatch.

To boot a newly installed kernel, see Section 7.14, "Booting the New Kernel."

6.10 Managing Source Code

If you intend to modify source, it is recommended that you use a source management scheme that will enable you to revert easily to a previous version of code if your current version fails. To this end, it is suggested that you use the Source Code Control System (SCCS) supplied with the UTX/32 distribution. See *Source Code Control System User's Guide [SV]* and *An Introduction to the Source Code Control System [BSD]* included in the UTX/32 supplementary documents that accompany your distribution. Another source control package, the Revision Control System (RCS), is available on the user library tape.

The following approach is recommended for use with SCCS:

- Be sure that the file access modes on all source files are set to read-only for everyone. This will help prevent inadvertent modifications to a source file. (It cannot prevent purposeful modifications, however.)
- When you have decided to modify a particular source file, be sure to make an SCCS storage file from the original source file before making any modifications. Use the SCCS **admin** command (see *admin*(1)).
- Use the SCCS **get** command, with the **–e** option, to check out an editable version of the file. After making your modifications, be sure to check the file back in with the **delta** command. See *get*(1) and *delta*(1) for more information.
- Extract a noneditable version of the changed source file, using **get** with no options, before recompiling.

You may want to modify the makefile associated with the modified source code so that it will automatically extract the source file from the SCCS storage file, when necessary.

7 Reconfiguring the System

Reconfiguring the system is broadly defined as the process of building, installing, and booting a kernel that is modified to meet a site's hardware configuration and performance needs. Devices and certain system parameters can be tailored to the needs of your particular installation by configuring your system at system generation time. System configuration allows you to configure your system without recompiling the entire UTX/32 kernel.

As delivered, the UTX/32 kernel is configured with the default values for system parameters and a default device configuration, and can be booted to multi-user mode if the hardware is configured to match the default configuration.

After the system has been booted to single-user mode for the first time, the kernel should be modified to match your hardware configuration.

The kernel can be reconfigured at system installation time or any time the system configuration changes. Administrators can

- Add devices to the kernel
- Modify device addresses
- Alter tunable system parameters
- Add or delete memory extents and physical memory regions

by editing the system configuration file and making a new kernel. After making such changes, the new kernel must be installed and booted before the changes are known to the system.

7.1 Guidelines for Formatting the System Configuration File

The entries in the input configuration file must be in a specific format. This section presents general guidelines to remember when editing the input configuration file.

The following section, Section 7.2, contains a sample configuration of a UTX/32 system, including the location of device controllers on both the SelBUS and a multipurpose bus (also called an IOP bus).

Four minimum device configurations are suitable for systems on which UTX/32 Release 2.1 is to be booted. Each includes the following:

- A Concept Product Line (CPL) machine capable of running UTX/32, such as a PN60xx, PN90xx, or MICROSel™ with 4 megabytes (Mb) of memory
- At least 160 Mb of disk space: two 80Mb disks or one 160Mb, 300Mb, 340Mb, or 675Mb disk configured with 1 KB per sector

- One Input/Output Processor (IOP) or one Multi-Function Processor (MFP) configured at SelBUS™ address 0x7e00
- One magnetic tape drive
- One console device

The four minimum configurations are distinguished by the device used as the magnetic tape drive, the device used as the disk drive, and the inclusion of an IOP or MFP. A configuration that includes an IOP must include a disk drive that is attached to either a Disk Processor II (DPII), a Universal Disk Processor (UDP), or a High-Speed Disk Processor (HSDP) and either a high-speed tape processor (HSTP), low-speed tape processor (LSTP), or Buffered tape processor (BTP). A configuration that includes an MFP may include a Small Computer Systems Interface (SCSI) bus, configured at an MFP subchannel, to which SCSI disks and tapes may be attached.

Table 7-1 summarizes the minimum configurations and the required controller locations for each.

7.1.1 Minimum Configuration

There are four minimum device configurations available to boot a UTX/32 system.

The first configuration includes

- One disk controller with a disk drive
- One tape controller with a tape drive
- One Input/Output Processor (IOP) or one Multi-Function Processor (MFP) with a console

The second configuration includes

- One MFP with a console, SCSI tape, and SCSI disk

The third configuration includes

- One MFP with a console and SCSI tape
- One disk controller with a disk drive

The fourth configuration includes

- One MFP with a console and SCSI disk
- One tape controller with a tape drive

The required controller locations for the minimum device configuration are listed in Table 7-1.

Table 7-1
Required Controller Locations

CONTROLLER	ADDRESS	BUS
Minimal Configuration #1:		
IOP or MFP	0x7e00	Sel
UDP or DPII or HSDP	0x0800 or 0x0c00 or 0x0400	Sel
LSTP or BTP or HSTP	0x1000 or 0x1800	Sel
Minimal Configuration #2:		
MFP	0x7e00	Sel
SCSI Disk	0x7e00 or 0x7e08	SCSI
SCSI Tape	0x7e10 (fiji)	SCSI
	or 0x7e18 (cipher)	SCSI
	or 0x7e40 or 0x7e48	SCSI
Minimal Configuration #3:		
MFP	0x7e00	Sel
SCSI Tape	0x7e10 (fuji)	SCSI
	or 0x7e18 (cipher)	SCSI
	or 0x7e40 or 0x7e48	SCSI
UDP or DPII or HSDP	0x0800 or 0x0c00 or 0x0400	Sel
Minimal Configuration #4:		
MFP	0x7e00	Sel
SCSI Disk	0x7e00 or 0x7e08	SCSI
LSTP or BTP or HSTP	0x1000 or 0x1800	Sel

You receive a UTX/32 system that is already configured for a usable set of devices. The values of all tunable system parameters are preassigned. The configured devices on the delivered system include the minimum required devices, plus devices that allow the system to be usable without necessarily having to reconfigure the system immediately. They are as follows:

- 1 MFP (mfp0) or IOP (iop0) at SelBUS address 0x7e00 with a console at subchannel 0xfc.
- 1 8-Line Asynchronous Communications Multiplexor (as0) configured at IOP or MFP subchannel 0x0e providing terminals numbered tty00 through tty07 for IOP or tty01 through tty07 for MFP. (On the MFP, tty00 is the console.)
- 1 lineprinter controller configured at IOP or MFP subchannel 0xf8 with 1 lineprinter device lp0.
- 1 Small Computer System Interface (SCSI) bus configured at MFP subchannel 0x00 with two SCSI disks: unit zero (sd0) at 0x00 and unit one (sd1) at 0x08. (Resulting disk addresses are: (sd0) 0x7e00 and (sd1) 0x7e08.)
- 1 SCSI bus configured at MFP subchannel 0x40 with two SCSI tapes: unit zero (st0) at 0x00 and unit one (st1) at 0x08. (Resulting tape addresses are: (st0) 0x7e40 and (st1) 0x7e48.)

- 3 SelBUS disk controllers (UDP, DPII, or HSDP) configured at SelBUS addresses 0x0800, 0x0c00, and 0x0400 provide disk units dk0 through dk3, dk4 through dk7, and dk8 through dk11 respectively.
- 2 SelBUS tape controllers configured at SelBUS addresses 0x1000 and 0x1800 provide tape units mt0 and mt1 respectively.
- 1 SelBUS ethernet controller configured at SelBUS address 0x0e00 provides ethernet unit en0.

7.1.2 Maximum Configuration

The maximum configuration for a UTX/32 system is

- A CONCEPT Product Line (CPL) machine capable of running UTX/32, such as a MICROSel, a PN60xx or a PN90xx with a CPU/IPU pair.
- 16 Mb of memory.
- System support processors configured from a mix of up to 4 IOPs or 16 MFPs.
- 1 console device.
- 8 Disk Processor IIs (DPII), Universal Disk Processors (UDP), or High-Speed Disk Processors (HSDP) (any combination).
- 64 disk drives (8 disk processors with 4 drives each plus 2 to 16 MFPs with up to 16 drives each but a maximum of 32 drives total).
- 4 SelBUS Ethernet™ processors.
- 8 Synchronous Communications Multiplexors (SCM), each with 1 to 4 ports, for a maximum of 16 links for high-speed communication to other CPUs.
- 4 high-speed, low-speed, or buffered tape processors.
- 80 tape drives (4 tape processors with 4 drives each, plus 16 MFPs with 4 drives each).
- 4 floppy disk drives (2 lineprinter/floppy-disk processors with 2 drives each).
- 4 lineprinters (2 lineprinter/floppy-disk processors with 2 lineprinters each or 4 MFPs with 1 lineprinter each).
- 128 terminals or other serial devices (for example, letter-quality printers) configured as 16 8-Line Asynchronous Communication Multiplexers (8LAS) on 4 IOPs or configured as 1 8LAS on 16 MFPs.
- 4 high-speed data (HSD) interfaces for use with custom devices.
- 6 Real-Time Option Modules (RTOMs).

NOTE: Depending on the use of IOPs and MFPs, the maximum configuration for certain peripherals may vary (i.e. serial, disk, and tape devices, line printers, floppy drives). Also, some of the above configurations are mutually exclusive.

7.2 Input Configuration File Format Rules

This section specifies the format of each type of entry that may occur in the configuration file. The following conventions are used:

- Characters that must be entered exactly as shown are printed in **boldface**.
- Entries that are variable in content are printed in *italics*.
- Optional entries are enclosed in square brackets, [ ].
- Ellipses (...) after an entry indicate that more than one value may be specified.
- The ordering of the information is critical:
  a. Tunable system parameters (if any) and memory extents
  b. Device configuration and physical memory regions (if any)
- White space (spaces, tabs, and newlines) is used only to separate words and is not otherwise required. Appropriate formatting is recommended to make the contents of the file readable.
- When a required entry is a number, the following conventions must be followed:
  - A hexadecimal value must be preceded by **0x** (zero-x). For example, the hexadecimal value 40 is expressed as

    ```
    0x40
    ```

  - A number beginning with a zero that is not followed by an **x** is interpreted as an octal value. Thus, 040 is an octal number.
  - All other numeric entries are interpreted as decimal values.
- Certain words are reserved. In addition to **at**, these include the following:

  Tunable system parameter names
  : **alternate-root**, **cpu**, **timezone**, **dst**, **hz**, **maxusers**, **memratio**, **nproc**, **nbuf**, **ntext**, **ninode**, **nfile**, **ncallout**, **nclist**, **nport**, **dumplo**, **maxuprc**, **dmmin**, **dmmax**, **npty**, **gateway**, **usrptpages**, **svchown**, **nccbs**, **niccbs**, **cistacksize**, **real-memory**, and **svdevt-size**

  Names identifying controller type
  : **disk**, **tape**, **ether**, **hsd**, **iop**, **lpr**, **floppy**, **asynch**, **custsel**, **custipi**, **custiop**, **memdisk**, **scsi-bus1**, **scsi-bus2**, **scsi-disk**, **scsi-tape**, and **mfp**

Names for standard controllers
dc, **tc**, **en**, **hs**, **iop**, **lc**, **fc**, **as**, **mc**, and **sc**

Unit declaration identifier
unit

Physical memory extent and region identifiers
memory, **region**, **length**, **in**, **shadow**, **reflective**, **vsa**, **cml**, **general**, **msms**, **nocache**, **noncontiguous**, **clear**, **type**, **probe**, **absent**, **mode**, **uid**, **gid**, and **initial-on**

Names for standard devices
dk, **lp**, **md**, **mt**, **fl**, **df**, **sd**, and **st**

Identifiers for lines on an **as** controller using modem controls
modems, **dtrflow**, and **rtsflow**

Identifier for custom controller mode type
char, **block**, or both; **uni** or **quad**; **length**

Identifiers for SCSI-type devices
fuji, **cipher**, **caliper**, **kenedy**, **patriot**, **m4**, **thorn**.

- The order of unit declarations made in the configuration file reflect the physical architectural hierarchy. For example,

```
tape    tc0
        unit    mt0     at 0x00

iop     iop0
        lpr     lc0     at 0xc0
                unit    lp0     at 0xc0
                unit    lp1     at 0xc1
```

implies a tape controller and an IOP on the SelBUS, with a lineprinter controller plus two lineprinters attached to the IOP bus.

NOTE: **iop** and **mfp** numbers cannot overlap. It is illegal to configure **iop0** and **mfp0** at the same time. The **iop**s and **mfp**s must be specified in order of increasing number; **iop0** or **mfp0** must be specified before **iop1** or **mfp1** and so on. The processor to be used as the console device must be specified first.

- The position of a device declaration in the configuration file indicates the controller to which that device is attached. The device is understood as being attached to the most recent controller declared. For example,

```
disk    dc0
        unit    dk0     at 0x00

disk    dc1
        unit    dk1     at 0x02
        unit    dk2     at 0x06
```

  implies that device **dk0** is attached to controller **dc0** and both devices **dk1** and **dk2** are attached to controller **dc1**; whereas

```
disk    dc0
        unit    dk0     at 0x00
        unit    dk1     at 0x02

disk    dc1
        unit    dk2     at 0x06
```

  implies that devices **dk0** and **dk1** are attached to controller **dc0** and device **dk2** is attached to controller **dc1**.

- You can include comments in the configuration file. They do not affect the processing of the configuration information in the file. They require the following format:

  # *Comment*

  The symbol # indicates that all of the text which follows, up to the next newline, is a comment. A comment may be entered anywhere in the configuration file. It may be a complete line or may occur at the end of any line.

7.3 Configuration Overview

The configuration file specifies

- Tunable system parameters
- Special directives
- Device configuration
- Physical memory extents and regions
- Comments

7.4 Tunable System Parameters

7.4.1 Guidelines for Setting the Tunable Parameters

The format for specifying tunable parameters in the configuration file is

parameter_name parameter_value

where *parameter_name* is the name of a tunable parameter and *parameter_value* is either the word **automatic** or is within the accepted range of values for that parameter. If **automatic** is used, a parameter is not specified, or *parameter_value* is not set, the kernel or the **/etc/config** program automatically assigns a default value. See Appendix C for a complete list of tunable parameter names and their accepted and default values.

7.4.2 Detailed List of Tunable Parameters

timezone	This parameter should be set to reflect the correct timezone at your site's location. Values correspond to minutes, with the value 0 representing Greenwich Mean Time. Values increase by 60 for each timezone west of Greenwich. Thus, 300 corresponds to Eastern Standard Time, and 360 corresponds to Central Standard Time. The default value is 300. See Section 4.2, ''Setting Timezone Daylight Savings Time Information'' for more information.
dst	The setting of this parameter tells your system if it should correct for Daylight Savings Time when reporting dates. The default value, 1, indicates that it should.
cistacksize	This parameter specifies the size, in words, of the stack that is allocated for indirect interrupts to run on. This stack is allocated from kernel space. The stack should be large enough to allow all routines that are called from an indirect interrupt to have stack frames. The current value of 384 is sufficient for this purpose.
nccbs	This parameter specifies the number of connections that can be made to interrupt levels using connected interrupt functionality. Users are able to make connections to **nccbs** different interrupt levels whether they are direct or indirect connections. Set **nccbs** to be the maximum number of interrupt levels that will ever be used. Do not include interrupt levels used by other configured devices. These do not subtract from the number of **nccbs** available for use. **nccbs** may be absent or zero in which case no direct or indirect connections can be made. Note, **nccbs** cannot be larger than 112.

niccbs This parameter specifies the total number of processes that can make indirect connections to interrupt levels. Every process that indirectly connects to an interrupt level subtracts from the number of **niccbs** available for use. **niccbs** is *not* the number of processes that can make indirect connections to *each* interrupt level; it is the total sum. **niccbs** may be absent or zero, in which case no indirect connections can be made. Note that setting **nccbs** to zero also has this effect. **niccbs** should be set to the maximum number of processes that will be allowed to make indirect connections to all interrupt levels. The maximum value for **niccbs** is 1024, but prudence dictates limiting **niccbs** to a modest number, such as 20.

nbuf This is the number of buffers for disk cache. One kilobyte of memory is reserved for each buffer specified. If this parameter is absent, approximately 10 percent of the physical memory will be used and the number of buffers will be set accordingly. If this number is too small, the disk throughput is going to suffer drastically on heavily loaded systems. If set too high, too much memory is wasted, causing the system to page or swap and thus degrading system performance.

Other tunable parameters can be adjusted to save memory; use **/etc/pstat –T** to see current usage.

See Chapter 14, "Monitoring System Performance," for information on how to determine whether or not your system could benefit by tuning these values.

npty This parameter should be set to reflect the desired number of pseudoterminals. The default value for this parameter is 32, but it may be set to any multiple of 16 from 0 to 256.

> NOTE: It is recommended that at least 16 PTYs are configured.

cpu	CPU type (67 or 97 only — currently unused)
hz	Power line frequency in hertz. The default is 60 Hz.
ninode	Number of in-core inodes
nfile	Number of in-core file structures
ncallout	Maximum number of pending kernel timeouts
nproc	Maximum number of processes
nclist	Total number of character I/O buffers

ntext	Maximum number of simultaneously active pure process texts
maxusers	Maximum number of allowed users
dmmin	Smallest contiguous allocation of swap space (in disk blocks)
dmmax	Largest contiguous allocation of swap space (in disk blocks)
dmtext	Swap allocation for text
dumplo	First disk block of the root swap space to be used for crash dumps
maxuprc	Maximum number of processes that a user may have running at any one time
usrptpages	Number of 8Kbyte pages of user page table space allocated (referred to as map image descriptors, or MIDs, in the hardware manuals)
gateway	(0/1) True (1) if machine is acting as a gateway between two or more directly connected networks
swdevt-size	The maximum number of block devices that can be used for swapping. The default is 16.

Additional parameters for System V compatibility include the following:

msgmap	Number of entries in the message map
msgmax	Maximum message size
msgmnb	Maximum number of bytes on message queue
msgmni	Number of message queue identifiers
msgssz	Message segment size (should be word-size multiple)
msgtql	Number of system message headers
msgseg	Number of message segments (must be less than 32768)
semmap	Number of entries in semaphore map
semmni	Number of semaphore identifiers
semmns	Number of semaphores in the system
semmnu	Number of undo structures in the system
semmsl	Maximum number of semaphores per semaphore ID
semopm	Maximum number of operations per **semop** call
semume	Maximum number of undo entries per process
semvmx	Semaphore maximum value
semaem	Adjust on exit maximum value

shmmax	Maximum shared memory segment size
shmmin	Minimum shared memory segment size
shmmni	Number of shared memory identifiers
shmseg	Maximum attached shared segments per process
shmbrk	Number of clicks between data and shared memory
shmall	Maximum total shared memory clicks in system
shmatt	Maximum number of attachs system wide
svchown	(0/1) Flag controlling how the System V **chown** system call behaves. By default, only **root** (the superuser) can change ownership of a file in the System V environment. This behavior is consistent with the BSD environment. If this logic flag is set (that is, given a value of 1), users other than root can change the ownership of a file, as would be expected on a System V system. The default is zero (off).

7.5 Special Directives

alternate–root	This directive is used to specify the name of the device that the kernel will use as the root device regardless of which device UTX/32 was IPLed from.
real–memory	This directive disables all swapping and paging activity. All processes must fit into physical memory.

7.6 Device Configuration

Declare each of the following in the configuration file:

- Controllers on the SelBUS
- Controllers on the IOP bus
- Devices on an MFP
- Devices on certain controllers
- Special regions of memory

The formats for specifying this information follows.

7.6.1 Declaration Format for Controllers on the SelBUS

The declaration format for a controller on the SelBUS is

> *controller_type controller_name* [**at** *address*]
> [**priority** *p*] [*mode*] [**length** *l*] [**subchannels** *n*]

The accepted values for the variable information are as follows:

controller_type
: May be **disk**, **tape**, **ether**, **hsd**, **iop**, **mfp**, or **custsel** for a custom device.

controller_name
: Must be of the form *namenumber*.

  name for a standard SelBUS controller is **dc** for a disk, **tc** for a tape, **mfp** for an MFP, or **iop** for an IOP. For a custom controller, *name* is any unique, two-character, lower case alphabetic combination. The *name* given for a custom device must be the same as that used in the customer written device driver.

  number must be a whole number and must be unique for each controller of the same type. Although not required, these numbers should be assigned starting at 0 (zero) and increase sequentially. In the absence of a specified overriding address, this number is used to obtain the controller's default SelBUS address.

at *address*
: Specifies the address for a standard or custom controller.

  This entry is optional for most standard SelBUS controllers, but it is required for Ethernet, HSD, IPI controllers, and any custom device attached to the SelBUS. *address* is the address of the controller on the SelBUS.

  If an address is not specified for a standard controller, the system assigns the default address. If an address is specified, an attempt is made to override the system default. If the override is successful, a warning message is written to standard output stating that the controller's address is not the system default address. This fact is also noted in the configuration report. If the specified address has already been assigned, then the override is not successful. This condition causes an error message on standard output and stops the system configuration process.

  If an address is not specified for a custom controller, it is an error. An error message is written to standard output and processing halts.

priority *p*
: Specifies the interrupt priority level, *p*, assigned to the controller.

This entry is optional for standard SelBUS controllers, except Ethernet, HSD, IPI and custom controllers. The interrupt priority level range for the Class E SelBUS controllers is between 4 and 19 and between 4 and 111 for Class F SelBUS controllers. These values determine the order in which the various controllers can interrupt the processor, with 4 being the highest priority.

If the priority is not specified for a standard SelBUS controller, the system default is assigned. If the priority is specified but does not match the default priority, the specified value overrides the default value. A warning is written to standard output, and the configuration report notes that the priority is nonstandard.

The default values for disk, tape, and IOP and MFP controllers are as follows:

Controller Name	Interrupt Priority Level
dc0	6
dc1	7
dc2	5
dc3	4
tc0	8
iop0 or mfp0	13
iop1	12
iop2	11
iop3	10
mfp1	15
mfp2	16
mfp3	17

Contention for the SelBUS is arbitrated by the hardware priorities jumpered on the devices. In general, choose the interrupt priority for a controller that is at or near the hardware priority of the device.

NOTE: The interrupt levels chosen for the various SelBUS controllers must be distinct.

mode

Applicable only to custom controllers.

mode is **char** for a character device, **block** for a block device, or **both** for a device that can be accessed in either block or character mode.

length *l*

Applicable only to custom controllers.

l is the length in bytes of the sense buffer associated with the custom device.

subchannels *n*

Applicable only to custom controllers.

n specifies the number of subchannels associated with each device attached to the controller.

7.6.2 Declaration Format for Controllers on an IOP Bus

The declaration format for a controller on an IOP bus is

> *controller_type controller_name* **at** *address*
> [*mode*] [*protocol*] [**length** *l*] [**subchannels** *n*]
> [**modems** *port* ...] [**dtrflow** *port* ...] [**rtsflow** *port* ...]

The priority for a controller on an IOP cannot be specified.

controller_type

Can be **lpr**, **floppy**, **scm**, **gbb**, **asynch**, or **custiop**.

controller_name

Must be of the form *namenumber*.

For standard IOP controllers, *name* is **lc** for a lineprinter, **fc** for a floppy disk, **sc** for an SCM, or **as** for an 8-line asynch. For custom controllers on an IOP, *name* is the unique, two-character, lowercase alphabetic combination used in the customer written device driver.

number must be a number and must be unique for each controller of the same type. Although not required, these numbers should be assigned starting at 0 (zero) and increase sequentially.

at *address*

Required for all IOP bus controllers.

If the IOP address is not present, an error is generated. This error sends a message to standard output and halts the system configuration process.

mode

Required for custom controllers, SCMs, and GBBs.

(It is not applicable to other IOP controller types.) For custom controllers, *mode* may be **char** for a character device, **block** for a block device, or **both** for a device that has features of both. For SCMs and GBBs, *mode* must be either **uni** or **quad**.

protocol

Applicable only to the SCM controllers.

Permissible values for *protocol* are

sc	(raw SCM interface)
km	(MTX)

If no protocol is specified, the protocol for networking software is used.

length *l*
Applicable only to custom controllers.

l is the length in bytes of the sense buffer associated with the custom device.

subchannels *n*
Applicable only to custom controllers.

n specifies the number of subchannels associated with each device attached to the controller.

modems *port ...*
Applicable only to 8-line asynch controllers.

To configure modems attached to an 8-line asynch, specify the location of each modem. Each *port* must be a digit between 0 and 7.

dtrflow port ...
Uses the read with flow control (DTR only) command of the 8-line asynch to allow the hardware to handle flow control.

NOTE: **flow** is a synonym for **dtrflow**.

rtsflow port ...
Uses the read with flow control (RTS only) command of the 8-line asynch to allow the hardware to handle flow control.

NOTE: This option is available only on 8-line asych hardware with revision 3G firmware (or higher). If the firmware is not at a high enough revision level, the option is ignored.

7.6.3 Declaration Format for Devices on an MFP

SCSI Devices

The syntax for a SCSI bus is

scsi-bus*n* **at** *subaddress*

n may be 1 or 2, and *subaddress* is the subaddress of the SCSI bus on the MFP. The subaddress specified for **scsi-bus1** is typicaly 0x00 and for **scsi-bus2**, 0x40. These addresses should be configured to match the hardware configuration. The first SCSI bus, **scsi-bus1**, has higher priority than the second bus. High throughput devices, such as disks, should be attached to **scsi-bus1**.

Lineprinter Devices

The line printer device on an MFP resides only at subaddress 0xf8. The syntax for a lineprinter is

> **lpr lp***n*

Asynchronous Devices

The asynchronous device on an MFP resides only at subaddress 0x0e. The syntax for an asynchronous device is

> **async as***n* [**modems** *port* ...] [**dtrflow** *port* ...] [**rtsflow** *port...*]

modems, **dtrflow**, and **rtsflow** are used in the same manner as 8LAS on IOPs.

Disk Devices

The disk devices are configured on the **scsi-bus***n*. The syntax for a disk drive is

> **scsi-disk sd***n* **at** *address*

address is the SCSI unit number 0 to 7 multiplied by 8 then converted to hexadecimal.

Floppy-Disk Devices

The disk devices are configured on the **scsi-bus***n*. The syntax for a disk drive is

> **scsi-floppy sf***n* **at** *address*

address is the SCSI unit number 0 to 7 multiplied by 8 then converted to hexadecimal.

Tape Devices

The tape devices are configured on the **scsi-bus***n*. The syntax for a tape device is

> **scsi-tape st***n* **at** *address* [*manufacturer*]

The optional manufacturer field determines the type of tape drive, (i.e., reel-to-reel, cartridge, etc). The default is reel-to-reel. The possible manufacturers are: **fuji**, **cipher**, **caliper**, **kenedy**, **patriot**, **m4**, and **thorn**.

> NOTE: The address for a disk or tape device on the MFP is the sum of the channel address of the MFP plus the subaddress of the **scsi-bus** plus the device address.

7.6.4 Unit Declaration Format

Certain controllers require that the attached device units be explicitly declared in the configuration file. Controller types for which attached units may *not* be specified are **iop**, **mfp**, **hsd**, **ether**, **scm**, **gbb**, and **asynch**.

To declare a device attached to a controller, the following format is required:

unit *unit_name* **at** *subaddress*

unit *unit_name*
: Specifies the unit name. This declaration must appear under the controller declarations for disk, tape, lineprinter, floppy disk, and all custom devices.

  unit_name takes the form *namenumber*. This must be unique for each individual device of the same type, regardless of the controller to which it is attached.

  name must be **dk** for a disk, **mt** for a magnetic tape, **lp** for a lineprinter, **fl** for a floppy disk, **md** for a memory disk, or any unique, two-character, lower case alphabetic combination for a custom controller.

  number specifies minor device numbers and consequently must not be duplicated within a controller type. Although not required, these numbers should be assigned starting at 0 (zero) and increase sequentially. These numbers are used to build various tables, specifically, the Subchannel Routing Table (SRT) and Minor Device to DCB Mapping Table (MTD); skipping a number results in null entries in these tables.

at *subaddress*
: is a number indicating the primary subchannel of the device. This subaddress is required for each unit.

7.7 Memory Declarations

To use the memory classes features of UTX/32, portions of physical memory must be declared in the **CONFIGURATION** file. For information on the nature and use of memory classes, consult the *UTX/32 Real-Time User's Guide*.

7.7.1 Format for Memory Extents

A *memory extent* is a range of addresses in physical memory. A memory extent declaration associates a name with a particular memory extent. Generally, all the pages in the extent will belong to a particular class of memory, such as shadow (fast) memory. Memory extents cannot overlap.

A memory extent declaration has the following form.

memory *name* **type** *type* **at** *addr* **length** *size*
[**control** *register_address*] [**key** *key*] [*attribute* ...]

memory *name*
The name of the memory extent. This name must not conflict with the name of any other memory extent or region. The memory *name* may include any alphanumeric character and dashes, but must have at least one alphabetic character.

type *type*
The class of memory within the extent. It may be any of the following: **shadow**, **reflective**, **vsa**, **cml**, **general**, or **msms**. **general** refers to general-purpose memory set aside for some special purpose; all the other types are special-purpose memory.

at *addr*
The address of the beginning of the memory extent. It must be an integral multiple of the page size which is *NBPG* in **/sys/sel/machparam.h**.

length *size*
The size of the memory extent in bytes. It must be an integral multiple of the page size which is *NBPG* in **/sys/sel/machparam.h**.

control *register_address*
The control register address. This option applies only to reflective memory. It is one of these hexadecimal numbers: 0x700 (the default), 0x708, 0x710, 0x718, 0x720, 0x728, 0x730, or 0x738. These are the physical addresses of the memory words that control reflective memory windows. See the *Gould Reflective Memory Port (RM Port) Model 7302A Technical Manual* for information on reflective memory.

key *key*
The shared memory key for the template region associated with this extent. (Refer to the *UTX/32 Real-Time User's Guide* for a discussion of template regions.) This option applies only to noncontiguous extents, which are defined in this list under the entry for *attribute*, following. The key must be unique; it must not conflict with the key of any other memory region. The default key is zero, meaning ``no key.''

attribute
Attributes give additional information about the extent. *attribute* may be any of the following:

nocache
Caching is turned off for addresses in this memory extent. This applies only to PN60xx and MICROSel machines. Caching is turned off for an area that is an integral multiple of 500 Kbytes and includes the extent. Since only one range of addresses at a time on a PN60xx or MICROSel machine may have cache disabled with the Shared Memory feature, it is recommended that extents requiring the **nocache** attribute be physically adjacent. The system startup code disables cache for the first extent requiring nocache through the last extent requiring nocache

and all inclusive memory addresses.

noncontiguous
: The extent is a *noncontiguous extent*, which means the extent acts as a pool of pages. When pages are allocated from the extent, they are not allocated in any particular order. In a *contiguous extent* (the default), a fixed number of pages are allocated that begin at a particular physical address and contain no gaps.

clear
: The memory in this extent is to be cleared at system boot time. **clear**, **probe**, and **absent** are mutually exclusive; more than one of them cannot be specified for the same extent.

probe
: The memory in this extent is to be probed at system boot time. Any parity errors discovered will be corrected by zeroing the guilty word. **clear**, **probe**, and **absent** are mutually exclusive; more than one of them cannot be specified for the same extent.

absent
: There is a gap in the machine's physical address space that corresponds to this extent. UTX/32 should not allocate or touch these locations. **clear**, **probe**, and **absent** are mutually exclusive; more than one of them cannot be specified for the same extent.

A memory extent unmodified by attributes is cached, contiguous, assumed to be present, and neither probed nor cleared at boot time.

7.7.2 Format for Memory Regions

A *memory region* is a subset of the pages that constitute a memory extent. The memory region declaration associates a name with a particular memory region. A memory region declaration has the following form.

region *name* **in** *memname* **at** *offset* **length** *size*
[**key** *key*] [**uid** *uid*] [**gid** *gid*] [**mode** *mode*] [*attribute* ...]

region *name*
: The name of the memory region. It must not conflict with the name of any other memory region. It must be no longer than **MAXREGNAME** characters. If you want to change this value, it is defined in **/sys/h/shm.h**.

in *memname*
: The name of the memory extent where the memory region is to be created. This extent must have already been declared.

at *offset*
: The offset, in bytes, of the first page to be allocated to the region. The offset is relevant only to regions within contiguous memory extents. If the region is within a noncontiguous memory extent, the offset must still be specified, but its value is irrelevant and should be zero. The offset must be

an integral multiple of the page size which is *NBPG* in **/sys/sel/machparam.h**.

length *size*
The size, in bytes, of the region. It must be an integral multiple of the page size which is *NBPG* in **/sys/sel/machparam.h**.

key *key*
The shared memory key. The key must be unique; it must not conflict with the key of any other memory region or extent. This number can be used to identify the region if, for some reason, the name is not used. The value of *key* must be a positive number. The default key is zero, meaning ''no key.''

uid *uid*, **gid** *gid*
The IDs of the owner and group, respectively, of the memory region. These are numbers, not string names like those found in **/etc/passwd** or **/etc/group**. The default value of each is zero.

mode *mode*
A number signifying, as for a file, the owner/group/other read/write/execute access permission modes for the region. A process may attach a shared memory segment as read-only or read-write; at that time, the process's effective user ID and group ID are checked against the mode. (See *intro*(2).) The default mode is 0666.

attribute
The attributes give additional information about the region. Currently only one attribute is available:

initial-on
This attribute is only applicable to memory extents of type **reflective**. It indicates that the reflective memory window for the owning memory extent is to be set to this region at boot time. If more than one region is given this attribute, only the last one declared will have the attribute.

7.8 Note about Configuring a Memory Disk

Two things are required when configuring a memory (ram) disk:

1. A **memdisk** directive, including the unit declaration(s). The default address for the memory disk is 0x7d00.
2. A **memory** directive specifying memory of type **general**, with a name of **memdisk***n*, where *n* is the minor number of the memory disk device. The physical address of the memory must start on a page boundary and be sized in page multiples. It is recommended that the **memory** declaration be declared at the top of physical memory. See Appendix C, Section C.2 ''Sample Configuration File'' for an example.

7.9 Note about Specifying Swap Partitions

Swap partition specification is no long necessary at configuration time. The UTX/32 kernel allows any valid block device to be dynamically added to the list of swap devices provided it is not already in use for swapping or file system use.

7.10 Generating a Reconfigured Kernel

Kernel reconfiguration is accomplished by changing to the kernel object directory and editing an input configuration file to add device specifications and alter the system parameters as desired. When the configuration file reflects the desired system, a modified UTX/32 kernel named **unix** is created by processing the edited input configuration file. An output report file and the system files needed to recreate **unix** are produced. The system files are then compiled, and the new kernel is loaded into a binary file called **unix**.

7.10.1 Editing the System Configuration File

To change the kernel device configuration or to modify the tunable system parameters, you must edit the system configuration file.

1. If the system is running multi-user, acquire superuser privileges. If the system is running single-user, you automatically have superuser privileges.
2. If the system is running single-user, ensure that all file systems are mounted by entering

   ```
   # cd /
   # /etc/mount -avt 4.3
   ```

   the **–a** option requests that all the file systems listed in the **fstab** file be mounted. This will give you access to the UTX/32 editing and processing tools that you are comfortable using. Mounting a selected subset could achieve the same result, but using **–a** is simpler.

   The **-v** (verbose) option specifies that the system notify you when each file has been mounted. The **-t** (type) option, along with the argument **4.3**, specifies that the files are Berkeley 4.3 as opposed to NFS files.

   NOTE: Always make sure that you are in the directory / when mounting or dismounting.

3. Change into the **/sys/obj** directory.

   NOTE: On source distributions only, **/sys** is a symbolic link to **/usr/src/src/sys**. If you change to the **/sys** directory and type the command **pwd**, the system responds with **/usr/src/src/sys**. Do not be confused by this.

4. Three configuration files exist in the object directory:
   - **GENERIC** is the configuration file describing the minimum bootable configuration for an IOP.
   - **CONFIGURATION** is the configuration file used to generate the kernel distributed with your system.
   - **CONF.DIST** is the configuration file used to generate the kernel used on the distribution tape.

   a. Make a backup of the **CONFIGURATION** file.

   b. Edit **CONFIGURATION** to set the tunable parameters to the values you desire and to reflect your actual hardware configuration. The contents and required format for the input configuration file are fully described in Appendix C

      NOTE: The system configuration information may actually be edited into a file of any name. Before making a new kernel, however, it must be in a file named **CONFIGURATION** for the makefile in /**sys** to recognize it.

5. If you have configured any custom devices, be sure to modify the makefile in /**sys** so that the custom device source files are compiled and loaded with the kernel.

   NOTE: The makefile in /**sys** is never run directly. It is always called indirectly through the makefile in /**sys**/**obj**.

7.10.2 Making the Kernel

When the **CONFIGURATION** file reflects the desired system and any necessary changes to the makefile are completed, create a modified **unix**. While in the /**sys**/**obj** directory, enter

```
# make
```

This command calls the makefile in /**sys**/**obj** which calls the makefile in /**sys**, which executes commands that result in the following:

1. The **CONFIGURATION** file is processed by /**etc**/**config**. If syntax or spelling errors exist in the file, the **make** halts. The errors must be corrected by re-editing the file. Then enter

```
# make
```

Repeat this edit/**make** sequence until no errors remain in the **CONFIGURATION** file.

When the file is correct, the required output source files are produced.

2. The output source files created by **/etc/config** are compiled, and certain configuration-dependent kernel source files are recompiled.
3. The resulting object files are loaded with the existing kernel object code to produce a new **unix** in the current object directory.

Check the **config_rpt** file to confirm that the desired devices have been configured correctly.

The kernel object files, as distributed, contain defines that will create a kernel of 4Mb virtual size. This parameter can be changed on the **make** command line by specifying **twomeg** (to make a 2Mb kernel). Note that if either the **twomeg** or **fourmeg** argument is specified, the default virtual kernel size changes to that argument and will be the default for all subsequent makes of the kernel until the alternative specification is made.

Making a 2Mb Kernel
To specify a 2Mb kernel, enter

```
# make twomeg
```

Making a 4Mb Kernel
To specify a 4Mb kernel, enter

```
# make fourmeg
```

NOTE: The capability to reconfigure and make both 2Mb and 4Mb kernels is also available on source distributions. See Section 6.7, ''Recompiling the Kernel.''

You should now have a new executable kernel contained in the file named **unix**. To boot this kernel, follow the instructions given in Section 7.12, ''Preparing to Boot the New Kernel.''

7.11 Error Handling

Errors can result from incorrect information being entered into the configuration file and from internal inconsistencies.

Unexpected entries in the configuration file may produce one of the following results:

- If the entry is a syntax error, an error message is produced on standard output and processing halts. An attempt is made to indicate the line number at which the syntax error occurred. (**yacc** is used to parse the input configuration file. It sometimes reports line numbers greater than the line number in which the error actually occurred.)
- If the entry contains information that is syntactically correct but erroneous in content, an error message indicating this is produced on standard output. Parsing of the input file continues if possible, but configuration processing halts and output files are not produced.
- If the entry implies a nonstandard request, a warning is written to standard output and processing continues. The default action taken is indicated. Warning messages should be examined carefully to verify that the system configuration process took the expected action.

In general, the flow of error-checking is as follows:

- Format errors edited into the configuration file are detected and reported as the file is parsed. If possible, parsing continues, without actually processing the data.
- As processing of the configured devices proceeds, addresses are verified and the number of devices that have been configured is compared to the minimum and the maximum number allowed.

All discovered errors must be corrected before the system configuration process can complete successfully.

7.12 Preparing to Boot the New Kernel

Once a new executable **unix** has been successfully created, the system must be rebooted with the new kernel. In preparation for this, the correct device entries must be made and the kernel must be installed.

If not already in single-user mode, bring the system down to single-user mode. See "Multi- to Single-user Mode" in Section 3.2.3, ''Switching between Modes.''

7.12.1 Creating Device Entries in /dev

If you have reconfigured the device information, you must make certain that the special files corresponding to those devices exist in **/dev**. These special files are the means by which programs access the physical devices.

Standard device entries can be built using some automated tools, but certain device entries must be built manually: custom device entries (see the section, "Custom Device Entries" in this chapter), device entries for additional pseudoterminals (see Section 7.16, "Adding Pseudoterminal (PTY) Devices"), and device entries for dialins and dialouts that have been configured (see Section 9.5, ''**tip** and **cu**'').

Standard Device Entries

One of the output files created by the system configuration program is **makedev.sh**, a shell script that aids in building the required /**dev** entries. **makedev.sh** invokes the shell script **MAKEDEV** with the appropriate arguments for producing device special files for each of the standard devices configured. Its use is illustrated below.

WARNING: The following commands must be carried out only while the system is in single-user mode.

1. Change into the /sys/**obj** directory.
2. Enter these commands:

```
# mkdir /newdev
# chmod 777 /newdev
# cp makedev.sh /newdev
# cp /dev/MAKEDEV /newdev
# cp /dev/MAKEDEV.local /newdev
# cd /newdev
# makedev.sh
# rm -fr /olddev
# mv /dev /olddev
# mv /newdev /dev
```

WARNING: If the new /**dev** entries are not created properly, you may not be able to reboot your system. If this should happen, you will need to reinstall the root image from the boot tape. If possible, reinstall the root image on a different disk drive and pack; this may enable you to salvage the existing system by mounting the root partition with the broken /**dev** entries and correcting them.

NOTE: If the kernel configuration has only MFP type tape devices, then the **makedev.sh** script will cause the tape device entries to be generated as /**dev**/***mt***. If only tape processor tape drives are configured, the device entries will be generated as /**dev**/***mt*** as well. If both MFP tape drives and tape processor tape drives are configured, then both /**dev**/***mt*** and /**dev**/***st*** entries will be generated.

Custom Device Entries

If custom devices have been configured into the system, the device special files for these custom devices must be built by hand, using /**etc**/**mknod**. (You must be superuser to use this command; see *mknod*(8).) This should be done after completing the creation of the /**dev** entries for standard devices, as described in the previous section.

WARNING: The system must be in single-user mode while building device entries.

1. Study the **config_rpt** file in /**sys**/**obj**. Note the device names for all custom devices, whether they are character or block devices, and their corresponding major and minor device numbers.
2. Change directory to /**dev**.
3. Enter a **mknod** command for each custom device.
4. Set the group, ownership, and access modes of the device entry.

Example

Suppose you have configured two custom block SelBUS devices named **xx0** and **xx1**, whose controller name is **xc0**. Looking at the **config_rpt**, you find that the major device number for this device type is 26 and the minor device numbers are 0 and 1, respectively. The **mknod** commands you would enter are

```
# mknod xx0 b 26 0
# mknod xx1 b 26 1
```

In this example, if these devices had been configured as both block and character devices, you would need to make a total of four device nodes: the two as shown above, plus

```
# mknod rxx0 c 26 0
# mknod rxx1 c 26 1
```

Notice that these entries differ in that the letter **r** is prepended to the device names and that the devices are **c** for character, rather than **b** for block. The letter **r** preceding the device name indicates raw I/O (character-at-a-time). This naming convention is used to make the name unique (since the system will not allow the creation of two different nodes with the same name) and to help the user distinguish which entry refers to which mode of device access.

7.13 Installing the New Kernel

Installing a new kernel involves placing the **unix** created in the /**sys**/**obj** directory into the root directory, /.

7.14 Booting the New Kernel

Once the new kernel is installed and all required device entries have been built, you are ready to boot the new kernel. In single-user mode,

1. Change directory to the system root, /:

   ```
   # cd /
   ```

2. Dismount all file systems:

   ```
   # /etc/umount -a
   ```

3. Halt the CPU and reboot. See Section 3.2.2, ''Halting from Multi-user Mode,'' or Section 3.2.1, ''Halting from Single-user Mode,'' and Section 3.2.4, ''Rebooting from Multi-user Mode''.

   WARNING: When rebooting a kernel, it is possible that the boot device has not been configured in the kernel. Should this occur, the system will halt during the boot process with no diagnostic messages displayed.

   Should an unexplained halt occur while booting a new kernel, reboot the default/backup system and inspect the output of the reconfiguration to insure that the boot device is configured in the kernel.

7.15 Adding Terminals

UTX/32 supports terminal lines that handle hard-wire terminals and dialin * and dialout modems. To set up lines to handle these types of devices requires two major steps, instructions for which are given in the following sections.

If you have modems or you wish to have an RS-232 line connected between machines for use by **tip** or **cu**, see *tip*(1), *cu*(1), and Section 9.5, ''**tip** and **cu**.''

7.15.1 Hard-wired Asynchronous Lines (Terminal Connections)

This section describes the steps necessary to make the hardware device known to the kernel.

After a hard-wired line is physically connected, the system must be modified to know about it. See Section 7.12.1 ''Creating Device Entries in /**dev**'' for instructions. To verify that the device entry exists and is correct, type:

* The term *dialin* is used to refer to all devices that may also be termed *autoanswer* or *dialup*; the term *dialout* is used to refer to all devices that may elsewhere be variously referred to as *automatic call*, *autodial*, or *callout*.

```
# ls -l /dev/tty*
```

to see the list of all hard-wired lines currently configured for the system.

A line's device entry is constructed from the line number. The line number is the number of the 8-Line Asynchronous Communications Multiplexor (or multifunction processor, if asynchronous operations are configured) to which it is connected, times 8, plus the line number (subchannel) of the line on the multiplexor. Thus, line 0 on multiplexor 0 has TTY number 00, line 0 on multiplexor 1 has TTY number 08, and line 7 on multiplexor 1 has TTY number 15.

To check whether the terminal **/dev/tty08** is known to the system, enter

```
# ls -l /dev/tty08
```

This should print a line similar to

```
crw------- 1 adm     7,8  Nov  7 12:07 /dev/tty08
```

Make sure the minor device number matches the terminal number. In this example

```
7,8
```

indicates major device 7, minor device 8.

If there is an error in any of the device entries for terminal lines, reexamine the system reconfiguration process to determine the source of the error. See the subsection "Standard Device Entries" in Section 7.12, "Preparing to Boot the New Kernel."

7.15.2 Editing /etc/ttys

The **/etc/ttys** file tells **init** what lines are known to the system and whether they are active terminal lines. The information in **/etc/ttys** also indicates the type of device attached to each system line. When a terminal is given a name, the system must be informed. This is done by editing a file called **/etc/ttys**. **/etc/ttys** is read when process 1, **init**, first executes. Normally, this occurs only when the system restarts. For changes to **/etc/ttys** to go into effect without restarting the system, enter

```
# kill -1 1
```

This signals **init** to reread the **/etc/ttys** file. See *tty*(7) for more information.

/etc/ttys contains one line for each terminal on the system. The first line describes the console, following lines describe normal terminals, and the remaining lines in the file describe pseudoterminals used for networking and for some commands. Here are two sample entries from the distributed file:

```
console     "/etc/getty std.9600"      vt100   on      secure      "Console"
tty00       none                       vt100   off                 "Regular tty"
```

The status of a line is either **off** or **on. on** means logins are to be allowed on that terminal, and **off** means they are not. The distributed system recognizes only the console. Change **off** to **on** where appropriate. If your system has more terminals than those mentioned in **/etc/ttys**, add new lines to the file.

Logins are not allowed on automatic call ports. If you have added automatic call ports, do not change the line in **/etc/ttys**. For example, if port 7 on multiplexor 0 has an automatic call port, the following line in **/etc/ttys** is left unchanged:

```
tty07       "/etc/getty std.19200"     T9      off      "Regular tty"
```

The second field in the file identifies the speed and initial parameter settings if logins are allowed on the terminal.

The commonly used choices are shown in Table 7-1.

Table 7-2
Terminal Speed Identifiers

String	Speed list
0 or d300	300-1200-150-110
std. 9600	9600
D1200	1200-300
D300	300-1200
std. 19200	19200

See *getty*(8), *tty*(7) and *gettytab*(4).

When a string identifier represents more than one speed, the first number is the speed initially assigned to the terminal. Whenever the user hits the break key while logging in or just carriage return for a login prompt, the next speed in the list is used. For example, many systems have 1200-baud dial-ups, which use the terminal speed identifier D1200.

If you have added dial-ups to your system, change the line in the **ttys** file to be the name of the dial-up. If the first dial-up is on port 0, multiplexor 1, change:

```
    tty08       none                       D5      off     "Unused port"
```

to

```
ttyd0      "/etc/getty D1200"    dialup  on      "Dialup Line #1"
```

7.16 Adding Pseudoterminal (PTY) Devices

A pseudoterminal (PTY) device is used by a software device driver that was originally designed for host-to-host communication instead of a terminal interface. Other utilities also rely on the presence of PTYs (for example, **script**).

Upon initially installing the UTX/32 system, the device entries for 32 PTYs should have been generated and entered in the **/dev** directory when the shell script **MAKEDEV** was executed.

A pseudoterminal is a pair of character devices, a *controlling* (*master*) *device* and a *slave device*. Each pseudoterminal consists of two files in **/dev**: the master pseudoterminal file is named **/dev/pty***xy* and the slave file is **/dev/tty***xy*. *x* is **p** for the first sixteen pseudoterminals, **q** for the next sixteen, and so on, and *y* varies from **0** to **f**.

The slave device provides an interface identical to that described in *tty*(7). However, whereas all other devices which provide the interface described in *tty*(7) have a hardware device of some sort behind them, the slave device has, instead, another process manipulating it through the master. That is, anything written on the master device is given to the slave device as input, and anything written on the slave device is presented as input on the master device. The slave side of each pseudoterminal must appear in the **/etc/ttys** file and must have a *none* in the second column so **getty** is not started for it.

The UTX/32 kernel as delivered allows a maximum of 32 PTY devices. The number of PTYs is a configurable system parameter. Thus, if more pseudoterminals are required, it is possible to configure up to 256 PTYs in multiples of 16.

7.17 Adding Users

For a new user to have access to system resources, the system administrator must add the new user to the password file and the group file. The password file, **/etc/passwd**, contains the list of users known to the system (see *passwd*(4)). The group file, **/etc/group**, contains the list of groups known to the system and the names of the users in each group (see *group*(4)).

To make the user's initial environment friendlier, the administrator should create a home directory for the user and copy startup files for the user's shell (templates of **/.cshrc**, **/.login**, or **/.profile**—see *csh*(1) and *sh*(1)) into this directory. For example, type

```
# cp /.login /mnt/username
```

NOTE: As distributed, a dot (.) is normally included in root's PATH. Because this could create a security breach, it is recommended that the dot be placed at the end of the PATH.

The administrator should also add the user to the system's mail alias list.

To add a user, perform the following steps as superuser:

1. Create a new entry for the user in the password file. This is done by typing

   ```
   # vipw
   ```

   and editing the password file. The filename **/etc/passwd** is taken as the default argument and need not be specified. See *vipw*(8).

   > NOTE: Always use **vipw** to edit the password file. This command locks the password file while it is being edited to prevent accidental concurrent modifications and ensures that some consistency checks are made on the file after it has been edited.

   vipw will execute the editor specified in the invoker's environment (default is **vi**) to edit the password file. This file contains lines of this form:

   username : *passwd* : *userID* : *def_groupID* : *finger_info* : *homedir* : *shell*

   The administrator must create a new entry of this form for each new user. The fields in the password entries are as follows:

   username — The user name. All user names must have the following characteristics:

   - They must be unique through the first eight characters. Because of this restriction, it is suggested that user names be no more than eight characters long.
   - The first character must be alphabetic or an underscore.
   - Characters other than the first must be either alphanumeric or underscores.
   - The login name must not be all uppercase.

   passwd — The encrypted password associated with the user name. This field may be * to prevent a user from logging into the system. New user entries initially should have a * within the password field.

   > NOTE: It is recommended that a new user's password field be initialized to *. This will disable the new user

	name until **passwd** is used to properly initialize the password. See *passwd*(1).
userID	The user identification number. A user ID is any number ranging from 0 to 65535. The user ID number may be arbitrarily chosen, except that all user IDs must be unique; otherwise, unpredictable results may occur during system operation. NOTE: The assigned username and user ID must be unique to retain unique identification of users throughout the life of the system. No two user entries in the password file for different users should ever have the same user name or user ID. Furthermore, user IDs should never be reused unless all files within the system, including those on backup tapes, are checked to ensure that no files have the reused user ID associated with them. Otherwise, users may inadvertently gain access to those files. It is also sound practice never to reuse a user name.
def_groupID	The default group ID for the user. The user's name must also appear in the default group's group member list in the group file.
finger_info	Used by the **finger** utility. Four comma-separated fields indicate the user's name, office, office phone extension, and home phone number. For historical reasons, this is sometimes called the gecos field.
homedir	The pathname of the directory into which the user is placed at login time. The home directory traditionally contains **username** as the last component of its pathname.
shell	The shell that is invoked for the user at login time. If the field is null, **/bin/sh** is executed as the default shell. See *passwd*(4).

Example

To create a new user called **jjones** in group **staff**, with **/mnt/jjones** as his home directory, **/bin/csh** as his login shell, execute **vipw**. Using editor commands, create a new line of the form

```
jjones:*:nn:10:Jim Jones,B36,523,5551212:/mnt/jjones:/bin/csh
```

The number *nn* is a user ID not already in use on the system. The number 10 is the ID number for the group **staff**. Group IDs can be found by looking in the group file, **/etc/group**.

2. After the addition is made, write and quit the editor. The user has been created within the password file.
3. Edit the group file to place the user in the group member list of any groups in which the user is permitted. (See Section 7.18, "Adding Groups," for more information.) The user must be included in the default group specified in the password file. A user may be put into a maximum of eight groups.
4. Create a home directory for the new user with **mkdir** (see *mkdir*(1)). This is the home directory path specified within the password file. To create a home directory for user **jjones** of the example above, enter the command

   ```
   # mkdir /mnt/jjones
   ```

5. Place a skeleton **.cshrc** file and **.login** file (if **csh** was specified within the password file) or a skeleton **.profile** file (if **sh** was specified) in the new user's home directory. Templates of these files may be copied from those residing in the root directory and modified as necessary.
6. Use **chown** to change the ownerships of the new user's home directory and the skeleton files to the new user. Change the group associated with the new user's home directory and the skeleton files to the user's default group with **chgrp**. Change the modes of the home directory and skeleton files with **chmod** to protect the new user's files. Recommended modes are 750 for the home directory and 740 for the skeleton files. See *chown*(8), *chgrp*(1), and *chmod*(1) for more information.

   For example, to change the ownership, group, and modes of the home directory and skeleton files for the **jjones**, enter

   ```
   # /etc/chown jjones.staff /mnt/jjones
   # /bin/chmod 750 /mnt/jjones
   # cd /mnt/jjones
   # /etc/chown jjones.staff .cshrc .login
   # /bin/chmod 740 .cshrc .login
   ```

7. Finally, assign the user a password using the **passwd** command.

   NOTE: The new user should be encouraged to change the password immediately after logging in for the first time.

8. Optionally, the user may be added to the mail aliases file (see *aliases*(4)). The mail aliases to which the user might be added include appropriate project mail aliases, an alias for the user's first name (if not the login name), an alias for the user's last name (if not the login name), an alias of initials, and so on. (See Section 9.2, "The Mail System," for more information on establishing mail aliases.) As always, execute **newaliases** after the aliases file has been changed (see *newaliases*(8)).
9. It is a good practice to save revisions of **/etc/passwd** in an SCCS file.

7.18 Adding Groups

Groups are named sets of users, used by UTX/32 to partition classes of users for discretionary access control. Groups should be set up to

- Partition users on different projects or work-related areas
- Partition users inside and outside a company
- Form special-purpose categories

The names of groups should be related to the function of group members. Examples of typical group names are **guests**, **sales**, **staff**, **operations**, **secretaries**, and **training**.

The list of groups is contained within the group file, **/etc/group** (see *group*(4)). To add a group, as superuser, use your favorite editor to edit the **/etc/group** file. Entries in the **/etc/group** file should be in the following format:

groupname : *group_password* : *groupID* : *username* , *username* , . . .

groupname
: The group name. All group names must be unique in the first eight characters. Otherwise, unpredictable results may occur during system operation. Because of this restriction, it is suggested that group names be no more than eight characters long.

group_password
: An obsolete field left from an earlier version of UNIX. This field should be left empty to prevent confusion.

groupID
: The group identification number. Group IDs range from 0 to 65535. The group ID may be chosen arbitrarily, except that all group IDs must be unique. Otherwise, unpredictable results may occur during system operation.

username,...
: A comma-separated list of user names. This is the list of users who are in the group, that is, the *group members*. A maximum of 200 users may be in the list. The list may be empty.

The administrator must assign a group name and a group ID to the new group and include in the group member list the user names of the users who are to be in the group. Any unused group ID may be assigned to the new group. It is good practice to keep the user names in the group member list in alphabetical order.

NOTE: Group IDs and group names must be unique. Furthermore, group IDs should not be reused unless all files within the system, including those on backup tapes, are checked to ensure that no files have that group ID associated with them. Otherwise, users may inadvertently gain access to those files.

The number of groups to which a user may belong is limited to eight.

Groups must be kept up-to-date to properly control access to files. For example, users should be added and removed from project groups in accordance with staff changes.

It is a good practice to save revisions of **/etc/group** in an SCCS file.

Example

To create a group called **training** containing users **jjones**, **msmith**, and **dbrown**, edit **/etc/group**. Using editor commands, create a new line of the form

```
training::21:jjones,msmith,dbrown
```

where 21 is a group ID that is unused anywhere else in **/etc/group**. Write the file and quit the editor. The group has been created.

NOTE: The new group does not go into effect for users already logged in—they must log out and log in to enter the new group.

For further information about the group file and its maintenance, see *group*(4), *vipw*(8), and Section 13.9, "**/etc/group**."

NOTE: In the System V environment, the utility **grpck** verifies the format of the group file after changes have been made. That utility does not exist in the BSD environment. See *grpck*(1M) in the System V environment for more information.

8 Establishing Environments

Before Release 2.0, UTX/32 combined a BSD system with selected features from System V to make a new, hybrid system. See Figure 8-1.

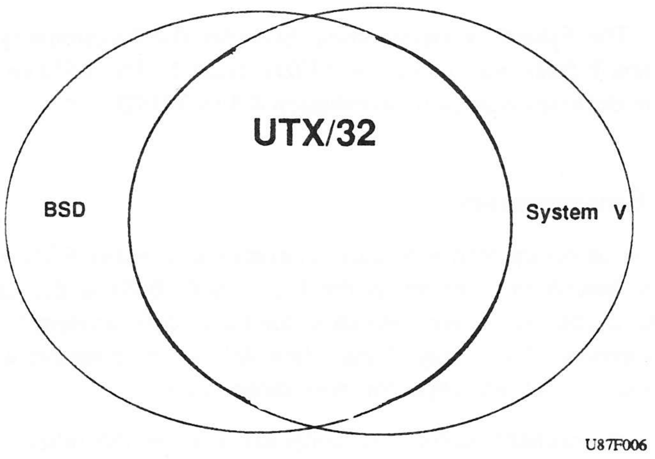


Figure 8-1. UTX/32 Environments Prior to Release 2.0

Release 2.0 and subsequent releases of UTX/32 provide a full BSD environment and a pure System V environment. Users can work in either environment. To switch environments, the user need only enter a simple command: **sv** to enter System V; **bsd** to enter BSD. See Figure 8-2.

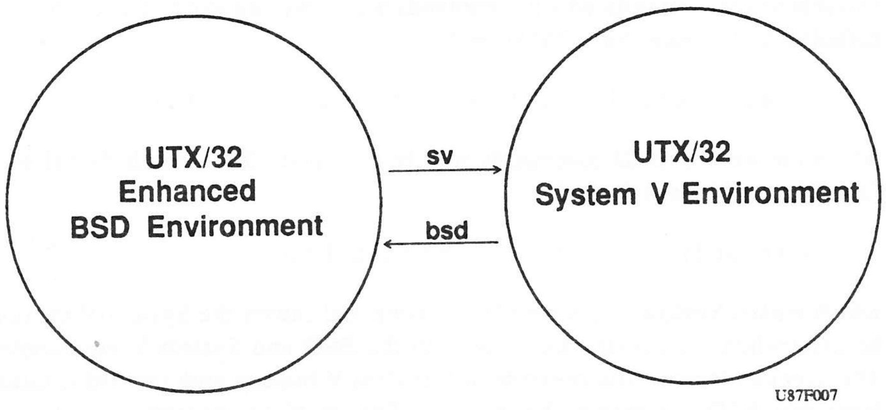


Figure 8-2. Current UTX/32 Environments

Within either environment, users can use **sv** or **bsd** to execute commands from the nonselected one, but the behavior of some system calls will be specific to the chosen environment. Commands will be found in one tree or the other depending on the user's search path. Most sites will choose the enhanced BSD environment as the default user environment.

After a brief section on how to move between environments, this chapter will focus primarily on how BSD users may access the System V environment.

> NOTE: The System V environment provides the functionality defined by the *System V Interface Definition (SVID)*, Issue 2. The BSD environment is based on Berkeley Standard Distribution 4.3 (4.3 BSD).

8.1 Moving Between Environments

New users can be set up with a default environment of either BSD or System V. If you select **/bin/sh** or **/bin/csh** as the login shell, BSD is the user's default environment. If you select **/usr/5bin/sh** as the login shell, System V is the user's default environment. Users may change their default environment with the **chsh** command. See *chsh*(1) and *sh*(1) for more information.

From either environment, users may temporarily enter the other environment. From the BSD environment, the **sv** command places the user in the System V environment with a default environment of **/usr/5bin/sh**. From the System V environment, the **bsd** command places the user in the BSD environment with a default environment of **/bin/sh**. Each temporary environment is exited when the user types ^D.

8.2 Command Execution

BSD commands are installed in the **/bin**, **/usr/bin**, and **/usr/ucb** directories. System V commands are installed in **/usr/5bin**. The user's PATH environment variable always controls which commands a user will execute. For example, the default PATH setting for a BSD user is

```
    .:/etc:/usr/local:/usr/ucb:/bin:/usr/bin
```

which ensure that BSD commands will be executed. The default PATH setting for a System V user is

```
    /usr/5bin:/usr/ucb:/bin:/usr/bin
```

which makes System V commands available and causes the System V version to be used when a command exists in both the BSD and System V environments. The user can temporarily override this System V biasing with the **bsd** command. Note that BSD commands that have no System V counterpart may always be executed.

Users may alter the default PATH by setting them in their **$HOME/.login** or **$HOME/.profile** (System V users).

8.3 System V Programming Environment

UTX/32 provides a System V compilation and execution environment that offers the user System V source compatibility. The **cc** command in **/usr/5bin** predefines a preprocessor variable called **sysv** so that code can be conditionally compiled in either the System V or BSD environments. This **cc** command uses the System V include files located in **/usr/5include** and **/usr/5include/sys**.

After preprocessing is complete, **cc** runs the compiler and assembler passes, which are the same for System V and BSD. The resulting objects are linked using a common linker, but the program is linked with the System V libraries found in **/usr/5lib**. These libraries contain the system calls necessary to provide a process with a System V system call and runtime interface, as well as standard System V library functions. When the resulting program is executed, it first notifies the operating system that it is a System V process. Thus, when the process makes a system call having disparate BSD and System V definitions, the correct behavior results.

> WARNING: Executing a command in one environment from the other environment sometimes gives unpredictable results. These cases occur when orders of search paths in each environment differ or when system calls (for example, **mount** and **pipe**) have unique, environment-dependent behaviors.

8.4 System V Interprocess Communication

As part of the System V environment, UTX/32 provides the Shared Memory (SHM) and Interprocess Communications (IPC) facilities defined in the SVID. The IPC facilities are a permanent part of UTX/32. There are parameters that control such things as message sizes, numbers of semaphores, maximum number of shared pages, and others, all of which are known to the **/etc/config** program. This arrangement allows these parameters to be modified at system configuration time. See Chapter 7, ``Reconfiguring the System''.

Two utilities, **ipcrm** and **ipcs**, are provided as a part of the IPC facility. The user can invoke these utilities from the shell to perform IPC and SHM maintenance operations. **ipcrm** can be used to remove message queues, semaphores, and shared memory ID's from the system. It will most likely be used in the course of application crash recovery or program debug. The **ipcs** command provides status about currently active IPC facilities. This information helps determine which processes currently have control of various resources.

Most of these IPC facilities are available in the BSD environment through the real-time library **/lib/librt.a**. See *ipcrm*(1) and *ipcs*(1) in the System V environment and the *UTX/32 Real-Time User's Guide* for more information.

8.5 System V Verification Suite (SVVS)

For users wanting to run AT&T's System V Verification Suite (SVVS) on the UTX/32 System V environment, the following procedures should be followed:

1. Change the value of **SV_chown** in the kernel from 0 to 1. As superuser do:

```
# adb -w /unix /dev/kmem
# SV_chown/W1        (for temp change)
# SV_chown?W1        (for perm change)
# ^D
```

2. Add **svvs** to **/etc/passwd**. See the *AT&T System V Verification Suite User's Guide* for more information.
3. Check that the login entry for **svvs** and **root** is in two different groups.
4. Recompile the entire SVVS including libraries.
5. Run the SVVS script **setaccess** before running the SVVS.
6. Delete all asterisks (*) from the password file (**/etc/passwd**).

NOTE: The value of **SV_chown** can also be changed as a tunable system parameter at configuration time. See Section 7.4, ''Tunable System Parameters'' for more information.

9 Communication Subsystems

This chapter describes how to

- Maintain and set up the lineprinter system
- Tailor the mail system
- Set up networking
- Use the Sun network extensions
- Configure the system, dialin lines, and modems to work with **tip** and **cu**
- Install and use UUCP (UNIX-to-UNIX copy)

9.1 The Lineprinter System

This section contains information about maintaining and setting up a lineprinter system. The lineprinter system, as distributed, is set up for a single lineprinter, but it supports multiple printers and multiple spooling queues.

Changes made to the lineprinter system are dynamic: as soon as the changes are in place, the newly configured printing device can generally be used without rebooting the system. (A reboot *is* required if you have modified the kernel.)

9.1.1 Maintaining the Lineprinter System

The lineprinter system consists of five elements:

Printing capabilities file

The file **/etc/printcap** lists the capabilities and attributes of each spooled printing device on the system. As distributed, it contains an entry for a single, standard lineprinter device:

```
lp|lineprinter:lp=/dev/lp0:sd=/usr/spool/lpd:if=/usr/lib/lpf:rf=/usr/lib/lrf
```

The fields in an **/etc/printcap** entry are separated by colons. The first field contains a list of alternate names for this lineprinter device. The remaining fields need not be in any particular order, since each is introduced by declaring the parameter that is being set. In the distributed file,

- **lp** specifies the full pathname of the special file for the lineprinter as **/dev/lp0**. There must be an actual special file with this name.
- **sd** specifies the full pathname of the directory for spooling print jobs for this device. A directory with this name must exist.
- **if** specifies the input filter to use with the device. The filter specified in this entry is the standard one distributed with UTX/32 systems.

- **rf** specifies the filter to use for printing FORTRAN-style text files.

  A full explanation of the fields in **printcap** entries can be found in *printcap*(4).

Spool directories
: Each printing device must have a designated directory for spooling its print jobs. Spooling directories are traditionally kept in **/usr/spool**. The distributed UTX/32 system includes a directory of the name **/usr/spool/lpd**.

Device special files
: Each local printing device must have a special file in **/dev** associated with it. A special file of the name **/dev/lp0** is created when you install the distributed UTX/32.

Kernel configuration
: The kernel, as distributed, is configured for a lineprinter attached to a lineprinter controller at 0x7EF8. You may need to modify the kernel configuration to match your hardware configuration.

/etc/ttys file
: In the **/etc/ttys** file, any tty line being used as a printer port should have login capabilities disabled.

9.1.2 Setting Up a New Lineprinter

If your lineprinter hardware matches the default configuration assumed in the distributed system, the lineprinter is usable with no further modifications to the system. Otherwise, you may need to modify elements of the distributed lineprinter system.

Five steps are required to set up a new lineprinter (steps 3 through 5 apply only to local printers):

1. Be sure that a correct entry appears in the file **/etc/printcap**.
2. Decide on an appropriate name for the spool directory of each printing device, and make sure that a directory of that name exists.
3. Be sure that appropriate special files exist in **/dev**. For adding **/dev/lp***n* entries, you can use the **MAKEDEV** command:

   ```
   cd /dev
   MAKEDEV lpn
   ```

   where *n* is a digit between 0 and 7.
4. Be sure that the device configuration in the kernel matches your hardware configuration. If you are using a lineprinter attached to a standard lineprinter controller, be sure that both the controller and the lineprinter device are configured in the kernel. If you intend to connect a printer device to a tty line, that tty line must be configured in the kernel.

5. Disable login capabilities in the **/etc/ttys** file for any tty line being used as a printer port.

Example

Assume that a printer is to be attached to the line **tty00**, that **tty00** is already reflected in the kernel configuration, and that the appropriate device entry already exists. It remains to modify the files **/etc/ttys** and **/etc/printcap** and create a spool directory. You must be superuser to perform these operations.

1. Modify **/etc/ttys**.

   Edit the file **/etc/ttys** and disable logins on the terminal line to which you intend to attach the printer. The following is an example of a modified entry:

```
tty00   none   vt100   off   "Line Printer #1"
```

   Effect the edits on **/etc/ttys** by typing

```
# kill -HUP 1
```

2. Make an entry in **/etc/printcap**.

   When you attach a printer device to a tty line, you must supply additional information indicating tty characteristics in **/etc/printcap**. An entry in **/etc/printcap** for a printer attached to line 0 on the 8-line multiplexor 0 (**as0**) might be

```
printek|dp:lp=/dev/tty00:sd=/usr/spool/lpd:br#9600:fc#43:fs#3280:sb:
```

   This entry indicates that

   - The name of the lineprinter is **printek** or **dp**.
   - The device entry for the lineprinter (**lp**) is **/dev/tty00**.
   - The spool directory (**sd**) is **/usr/spool/lpd**.
   - The baud rate (**br**) of the tty line is 9600.
   - Certain control bits are cleared (**fc**).
   - Other control bits are set (**fs**).
   - The short form of the banner is to be used on all listings printed on this device (**sb**). (**sb** is especially useful on slow printers.)

   To calculate the values for **fc** and **fs**:

   a. Select the required attributes from the list of valid values for the **sg_flags** field (referenced whenever an **ioctl** call is issued for that line) given in the "Basic Modes" section of *tty*(7). These values are given in octal notation. The attributes you choose depend upon the type of printing device; check the device reference manual for required settings.

b. Add the values for the attributes you require. (Be sure to use octal arithmetic.)

c. Convert the octal value from the previous step to decimal, or precede the octal value with a 0.

In this example, the **fc** value is obtained by adding the octal representations for CBREAK, TANDEM, ECHO, and RAW. The **fs** value is obtained in a similar manner. Table 9-1 shows the necessary calculations corresponding to the steps above.

Table 9-1
Calculating Values for **fc** and **fs**

Step	Flag Bits			
	Clear (**fc**)		Set (**fs**)	
1	CBREAK	0000002	CRMOD	0000020
	TANDEM	0000001	EVENP	0000200
	ECHO	0000010	ODDP	0000100
	RAW	0000040	TBDELAY	0006000
2	Octal	0000053	Octal	0006320
3	Decimal	43	Decimal	3280

3. Make a spool directory.

You must now set up a spool directory whose name matches the name indicated in the **sd** field of the **printcap** entry.

- Make the spool directory with the command

```
# mkdir /usr/spool/lpd
```

- Change the ownership of the directory to **adm**, change the group ownership of the directory to **adm**, and change the mode of the directory to read-write-execute permissions for owner, group, and public:

```
# /etc/chown adm.staff /usr/spool/lpd
# chgrp adm /usr/spool/lpd
# chmod 777 /usr/spool/lpd
```

4. Enable printing.

Use the lineprinter control utility, **lpc**, to enable printing on the new device:

```
# /etc/lpc start printek
```

You should now be able to use the newly configured printer.

9.2 The Mail System

UTX/32 is distributed with a mail system that handles mail sent between users on the same system or between users on systems linked by a UUCP or TCP/IP network. This mail system may be used without further modification, but it can also be tailored to your site's needs. Specifically, it can be set to recognize group names and alternate usernames, and its network access can be expanded.

9.2.1 Local Mail

/usr/lib/aliases is the mail alias file. The version of this file that is distributed with the system, reproduced in Section 13.39, ''**/usr/lib/aliases**,'' defines aliases for the mailer daemon and the postmaster. Do not change the mailer daemon alias; the postmaster daemon alias can be modified. Neither alias can be deleted.

You may tailor mail to your system by adding entries that allow the mail system to recognize users by names other than their login names or to recognize groups of names. See *aliases*(4) for more information.

Recognizing Multiple Names

To allow the mail system to recognize a user by various names, add each of those names to the alias file as an individual entry. For example, suppose Derek Smith's login name is **dks**. Without the mail alias file, only mail addressed to **dks** will actually be delivered to Derek Smith's mailbox. But if the following entries are added to **/usr/lib/aliases**

```
derek:dks
Smith:dks
```

then mail addressed to **derek** or **Smith** will also be delivered correctly to Derek Smith's mailbox. In addition, the mail system converts names to lowercase before comparing them with entries in the alias file, so mail addressed to **Derek**, **Smith**, or **DKS** will be delivered correctly.

Recognizing Groups

To allow a group of people to be recognized, add an entry defining the group. For example, suppose the members of the sales team have login names **dks**, **leo**, and **doug**. With the following entry in **/usr/lib/aliases**

```
sales: dks,leo,doug
```

any mail sent to **sales** will automatically be sent to all three people.

After initially setting up the **/usr/lib/aliases** file, execute the following commands (as superuser) to set up the mail alias database files:

```
# cp /dev/null /usr/lib/aliases.dir
# cp /dev/null /usr/lib/aliases.pag
# chmod 666 /usr/lib/aliases.*
# newaliases
```

The files **aliases.dir** and **aliases.pag** are the database files, maintained by **dbm**, containing information from the **aliases** file (see *dbm*(3X)). These are the files that the mail system uses when attempting to resolve mail aliases. It is therefore extremely important that you keep these database files up to date by *always* executing the utility **newaliases** after any modifications to the **/usr/lib/aliases** file:

```
# newaliases
```

The system will not know any new aliases added to **/usr/lib/aliases** until this command is executed.

For **sendmail** to work properly on UTX/32, start its daemon with the **–q** option to enable queuing of mail messages. To see the correct way to start the **sendmail** daemon, look at its invocation in **/etc/rc**.

NOTE: Because **mail** creates additional files, the **/usr** filesystem must not exceed 90% of capacity to run it.

9.2.2 Network Mail

Network mail requires the presence of the aliases file **/usr/lib/aliases** and the network configuration file **/usr/lib/sendmail.cf**. The distributed configuration file works for a small number of machines linked with either a UUCP net or TCP/IP net. You can use this file as is, or you can modify it to expand your network mail capacity.

The **–bz** option of **sendmail** creates the **/usr/lib/sendmail.fc** (frozen configuration) file. The **sendmail.fc** file must be recreated whenever the **sendmail.cf** (configuration file) file changes.

For more information about local and network mail, see *mail*(1), *aliases*(4), and *newaliases*(8). For detailed information, see *sendmail*(8), Section 9.2.3 ''Sendmail Mailer Setup,'' and *Sendmail Installation and Operation Guide* and *SENDMAIL–An Internetwork Mail Router* in the UTX/32 supplementary documents.

9.2.3 Sendmail Mailer Setup

Release 2.0 and subsequent release of UTX/32 use a system for setting up mail that is different from previous releases. If your machine needs to deliver mail only to its local users, you can skip this section. If you are connected to an Ethernet or UUCP connection or other network, this section will be of interest.

Picking Your Domain Name

The network mail program **sendmail** uses Internet standard mail addressing formats to enable any Internet system in the world to exchange mail with any other Internet system. For this to work, each system must have a unique name.

To make them easier to manage, the world of mail addresses is divided into *domains* and *subdomains*. Within each domain, each system must have a unique name, but another machine in another domain could have that system name.

Domain names are read from right to left. The full name **mach.COM** says that the machine **mach** is in the domain **COM**. The full name **mach.gould.COM** says that the machine **mach** is in the subdomain **gould** which is a member of the domain **COM**. The last domain is usually called the *root domain* or *top-level domain*. In both examples **COM** would be the top-level domain for the machine **mach**.

You will need to determine your top-level domain name. For example, if your machine is connected only to UUCP (Usenet), your top-level domain is probably UUCP. If your machine is connected to the MILNET or ARPANET, your top-level domain is probably **ARPA**, **COM**, **EDU**, or **GOV**. When you determine your top-level domain, remember that the domain does not need to be directly connected to your machine. If your system talks to machine X, and X has an ARPANET connection that your machine uses, then your top-level domain can be one of the ARPANET domains.

If your system has no way of reaching the outside world, use any domain name or use the name UUCP, which allows you to exchange mail via the telephone with other sites using UUCP. See *uucp*(1C) for more information.

Setting Mailer Type

You will need to determine whether your system is a *root mailer* or a *leaf mailer* in a network. Your system must be one or the other type of mailer.

If your machine is attached only to an Ethernet, then you are probably a leaf machine. This means that you will be sending mail for other domains across the Ethernet to a machine that will contact the other domains for you.

If your machine has a UUCP (Usenet) connection or ARPANET/MILNET connection, then you are probably a root machine. You will be exchanging mail with systems in other domains.

UTX/32 comes with a prototype **sendmail** configuration file for the leaf and root mailers. These files, **sendmail.leaf.cf** and **sendmail.root.cf**, are located in the directory **/usr/lib**. Copy the prototype configuration file of your choice to the file that **sendmail** uses, called **/usr/lib/sendmail.cf**.

> NOTE: These configuration files are provided so that most users will not need to write their own. If your site configuration is too complex to work with the mail configuration files provided, you will need to write your own. Use these files as a reference and read the *sendmail*(8) manual page.

Setting Your Domain Name

Once you have decided on a domain name, edit **sendmail.cf** to add it to your configuration. To set your top-level domain name, look for the lines in the configuration file similar to the following:

```
# domain
DDxxx
CDLOCAL xxx
```

Replace the name *xxx* on both lines with the name of your top-level domain. Save the edited configuration file.

If your system will be functioning as a root mailer, you have completed your **sendmail** setup. Make sure that any Ethernet hosts to which you want to send mail to are listed in your **/etc/hosts** file. Save the edited configuration file and either reboot your system or restart the **sendmail** daemon. See *sendmail*(8) for more information.

Setting Your Mail Relay

If your system will be functioning as a leaf mailer, you must tell it where to send outgoing mail for processing. This will usually be another host on a local area Ethernet. Add the machine's name to your **sendmail** configuration file. Edit the file **sendmail.cf** and look for the lines in the configuration file similar to the following:

```
# host on LAN with UUCP   (or other) connection
DRxxx
```

Replace the name *xxx* with the name of your outgoing mail relay. Save the edited configuration file. You should now be done. Reboot your system or restart the **sendmail** daemon for the changes to take affect.

9.3 Networking

UTX/32 provides support for a subset of the DARPA standard Internet protocols IP, TCP, UDP, and Xerox Network Services (XNS™) protocols. These protocols may be used on top of a variety of hardware devices ranging from the IMPs used in the ARPANET to local area network controllers for the Ethernet. Currently, only the Synchronous Communication Multiplexor (SCM), the Ethernet, and the NSC HYPERchannel® are supported.

Network services are split between the kernel (communication protocols) and user programs (user services such as TELNET and FTP). This section describes how to configure your system to use the networking support.

9.3.1 System Configuration of Networking

Many of the network utilities (for example, **rsh** and **rlogin**) use pseudoterminals. These utilities establish a connection to a pseudoterminal on another machine and thus become, effectively, a terminal on that machine. Pseudoterminals are also used by the **script** and **window** programs (see *script*(1) and *window*(1)).

As delivered, the system has 16 pseudoterminal pairs configured. That is, appropriate files exist in **/dev**, appropriate entries exist in **/etc/ttys**, and the value of the soft parameter **npty** is 16. If more pseudoterminal pairs are needed, see Section 7.16 ``Adding Pseudoterminal (PTY) Devices'', for instructions on how to add them.

9.3.2 Setting Up the IOP/SCM Controller

The SCM supports point-to-point communication paths between pairs of machines. Each SCM board can support one communications line at up to 56000 baud in **unimode** or up to four communication lines at 9600 baud in **quadmode**. For each SCM, a line must be added to the system **CONFIGURATION** file with the format

```
scm scn at     addr type
```

type is either **quad** or **uni**. See **/etc/scm_info** for additional SCM parameter information.

9.3.3 Setting Up the SelBUS Ethernet Controller

The Ethernet controller is a standard SelBUS device and therefore must be configured as such. The following entry should be made in your **CONFIGURATION** file after the last entry for **tape** and before the entry for **iop**:

```
ether enx at address priority n
```

where *x* specifies 0 to 3, *address* is the ethernet address, and *n* is the interrupt priority number.

The interrupt priority for the Ethernet board must be a number less than the priority for the IOP, allowing the Ethernet board a higher interrupt priority than the IOP. This prevents users at terminals from generating interrupts that would block Ethernet transmission. The recommended order of interrupt priority, highest to lowest, is disk, tape, Ethernet, and IOP.

The disk interrupt priority defaults to 6, tape defaults to 8, and IOP defaults to 13. Therefore, set the priority for the Ethernet board at 10. If the system has two tape drives and they are at interrupt priorities 8 and 10, set the Ethernet priority at 11. The IOP remains at 13.

In a system with several IOPs, it is necessary to set the SelBUS Ethernet at a priority lower than 10 (for example, 8). See *ec*(7) for more information.

For a detailed description of the Ethernet Controller, see the *Ethernet Controller Model 8516 Technical Manual*, not included in the UTX/32 documentation set. After the kernel is built, perform the following:

- Make entries in /dev
- Edit host files
- Edit /etc/rc.boot
- Initialize the network device
- Edit **/etc/rc.local**

Making Entries in /dev

For each Ethernet controller, create a **/dev/en***n* entry if one does not already exist. Do this using the **MAKEDEV** command:

```
# cd /dev
# MAKEDEV enn
```

For more information, see *makedev*(8).

Editing Host Files

Set up the Ethernet host names and addresses appropriately in the **/etc/hosts** and **/etc/hosts.equiv** files. See Sections 13.10, ''**/etc/hosts**,'' and Section 13.12, ''**/etc/hosts.equiv**,'' for details.

Editing /etc/rc.boot

The local host name is set in **/etc/rc.boot**. At installation time, you must edit this file, replacing **noname** in the line

```
/bin/hostname noname
```

with the name of your host.

Initializing the Network Device

Use **/etc/ifconfig** to initialize the device.

For Ethernet devices, use this command line:

```
# /etc/ifconfig enn protocol internet_address_or_hostname up_or_down arp –trailers_or_trailers broadcast_network_number
```

For SCM devices, use this command line:

```
# /etc/ifconfig scn[a,b,c,d]  protocol up_or_down internet_address_or_hostname remote_host_address_or_hostname
```

protocol
: Specifies **inet** or **ns** for Internet of XNS protocols.

internet_address
: Specifies the numerical four byte Internet address.

hostname
: Specifies the name of the Internet host.

up_or_down
: Marks the interface **up** or **down**.

arp Specifies the address resolution protocol (optional).

–trailers_or_trailers
: Specifies a VAX^{TM}-type machine. The current Ethernet driver supports the trailers protocol.

broadcast_network_number
: Sets broadcast flag on or off, specifying the address used to broadcast to the network.

[a,b,c,d]
: Indicates the optional four channels for SCM devices in **quadmode**.

After successfully installing the SelBUS Ethernet controller(s) or SCM, **/etc/ifconfig** may be added to the **/etc/rc.local** file for automatic execution at boot time.

Editing /etc/rc.local

The UTX/32 distributed **/etc/rc.local** file has some of the networking-related commands and daemons commented out. You should uncomment the **/etc/ifconfig** command, which is in this case the Ethernet network interface.

```
# Below is an example ifconfig for INTERNET (TCP/IP) protocols, uncomment
#            to turn on INTERNET protocols.
#     Warning: you may need to set your broadcast address to be 4.3
#                    compatible.  See the man page and docs!
# /etc/ifconfig en0 inet `hostname` up arp -trailers
```

Note that the local host name must have already been set using **hostname**. Normally this is done by **rc.boot** when the machine is booted. For more information about **ifconfig**, see *ifconfig*(8).

The following daemons are commented out in the distributed **/etc/rc.local**:

gated
An advanced routing daemon. This program supports Berkeley UNIX RIP protocol (see *routed*) along with ARPANET EGP and NSFnet HELLO protocols. It is the recommended routing program. See *gated*(8) for more information.

routed
The Berkeley network routing daemon. It is intended to keep the kernel's Internet routing tables up to date based on information supplied by other hosts within a cluster of networks. See *routed*(8) for more information. Note that this daemon is superseded by **gated** and may be dropped in future product releases.

timed
The time daemon. It attempts to synchronize the clocks of all hosts on the local network that also run **timed**. See *timed*(8) for more information.

ypserv
The yellow pages server process. It should be run only on YP server machines with complete YP databases. See *ypserv*(8) for more information.

ypbind
The yellow pages binder process. It should run on all machines using YP services. See *ypbind*(8) for more information.

Depending on your local network configuration, you may want to modify **/etc/rc.local** so that one or more of these daemons start up on each reboot.

9.3.4 Ethernet Modes

Several modes are now supported for the Ethernet. Refer to the *Ethernet Controller Technical Manual* for more information.

Mode 0 (transparent mode)

The standard Ethernet 1.0 packet is passed, as is, to the network and localhost.

The format of the packet is

destination	6-byte Ethernet address
source	6-byte Ethernet address
type	2-byte type field

Mode 1 (Ethernet 1.0) and Mode 2 (Ethernet 2.0)

The standard Ethernet 1.0 packet is passed to the network, but the controller supplies the source address. The packet passed to the local host from the controller has a 2-byte length field added to it. This field is stripped off by the Ethernet driver before being passed to ARP, IP, or XNS, but it is available to the user when reads are issued to the driver.

The only difference between mode 1 and 2 is electrical.

The format of the packet is

write

destination	6-byte Ethernet address
* * * * * *	Controller supplies source address
type	2-byte type field

read

destination	6-byte Ethernet address
length 2-byte length supplied by the controller to the driver	
source	6-byte Ethernet address
type	2-byte type field

Mode 3 (IEEE 802.3)

The standard IEEE 802.3 packet is passed to the network. The user supplies only the destination. The controller puts the source and length field in the packet. The packet passed to the driver from the controller is the standard IEEE 802.3 packet with a length field added. Currently, no IEEE 802.1 and IEEE 802.2 networking software is available above the Ethernet driver. See *enfunc*(8) for how to put the **en** card into IEEE 802.3 mode. Note that the mode must be set before an **ifconfig** of the device is executed.

The format of the packet is

write

destination	6-byte Ethernet address
* * *	Controller supplies source address
* * *	
* * *	Controller supplies length field

read

destination	6-byte Ethernet address
length	2-byte length supplied by the controller to the driver
source	6-byte Ethernet address
length	2-byte length field

Set up the Ethernet host names and addresses appropriately in the **/etc/hosts**, **/etc/hosts.local**, and **/etc/hosts.equiv** files. See *hosts*(4) for more information.

9.3.5 Network Databases

The data files used by the network library routines and server programs are listed in Table 9-2. Most of these files are host independent and rarely updated. This section briefly explains the contents of each network database. More information on each file may be obtained by reading the manual pages shown in the table.

Table 9-2
Network Databases

File	Manual Page	Use
/usr/spool/rwho	*rwho*(1)	Remote user information
/etc/hostdb_method	*hostdb_method*(4)	Host name resolver selection
/etc/hosts	*hosts*(4)	Host names
/etc/networks	*networks*(4)	Network names
/etc/services	*services*(4)	List of known services
/etc/protocols	*protocols*(4)	Protocol names
/etc/hosts.equiv	*rshd*(8)	List of trusted hosts
/etc/rc.local	*rc*(8)	Command script for starting server
/etc/ftpusers	*ftpd*(8)	List of unwelcome FTP users
/etc/gateways	*routed*(8)	List of neighboring gateways
/etc/named.*	*named*(8)	Internet domain name server
/etc/gated.conf	*gated*(8)	Advanced routing config. info.
/etc/scm_info	*scm_info*(4)	SCM configuration information

/usr/spool/rwho

rwho lists the users logged in on all machines on a local network. This directory must be created, if it does not exist, as a place for **rwho** to store remote user information on the local machine.

/etc/hostdb_method

/etc/hostdb_method tells the system what methods to use for host name and address lookup and in what order to use the methods. The file lists the methods, one per line. A line with a # at the beginning is a comment and is ignored. If a program needs to look up a host name or address, the listed methods will be tried in the order given.

The available choices are the DOD domain server (**nameserver**), Sun Yellow Pages (**yellowpages**), and use of the **/etc/hosts** file (**hosttables**). As distributed, only the **hosttables** method is used:

```
#nameserver
#yellowpages
hosttables
```

For more information, see *hostdb_method*(4).

NOTE: It is wise to list **hosttables** as a last resort and to keep a minimum set of information in the **/etc/hosts** file, even if you are using one or both of the other nameservers.

/etc/hosts

The **/etc/hosts** file identifies network hosts. For each host, it lists the network address, the local host name, and any other names (aliases) by which the local host is known. The distributed **/etc/hosts** file is

```
127.1           localhost
#
# Add your systems below - do not change
# or remove localhost definition above
#
1.2             sysname SYSNAME Sysname
```

The first line in the file defines the loopback connection. It is the same for every host and should not be removed or modified. The contents of the rest of the file reflect each site's configuration. An example host entry is

```
192.5.13.1      alberto al a
```

The numbers represent, in decimal, the digits that specify the host address in the Internet format.

/etc/networks

This file provides mapping between network numbers and network names. **/etc/networks**, as distributed, is

```
loopback        127
```

You will have to add an entry for each network you wish to access, including the local network. Here are some sample entries:

```
Ethernet     140.0     localnet
test-net     142.0
```

/etc/services

This file provides mapping between TCP port addresses and standard services provided by those addresses. This file does not normally need to be modified in the field. See *services*(4) for more information.

/etc/services as distributed is divided into two sections: Network services (Internet style) and UNIX specific services:

```
#
#                   @(#)services        1.16 (Berkeley) 86/04/20
#
# Network services, Internet style
#
echo                          7/tcp
echo                          7/udp
discard                       9/tcp           sink null
discard                       9/udp           sink null
systat                        11/tcp          users
daytime                       13/tcp
daytime                       13/udp
netstat                       15/tcp
qotd                          17/tcp          quote
chargen                       19/tcp          ttytst source
chargen                       19/udp          ttytst source
ftp                           21/tcp
telnet                        23/tcp
smtp                          25/tcp          mail
time                          37/tcp          timeserver
time                          37/udp          timeserver
rlp                           39/udp          resource# resource location
nameserver      42/tcp                        name# IEN 116
whois                         43/tcp          nicname
domain                        53/tcp          nameserver# name-domain server
domain                        53/udp          nameserver
mtp                           57/tcp          # deprecated
tftp                          69/udp
rje                           77/tcp          netrjs
finger                        79/tcp
link                          87/tcp          ttylink
supdup                        95/tcp
hostnames       101/tcp                       hostname# usually from sri-nic
#csnet-cs       105/?
pop                           109/tcp         postoffice
sunrpc                        111/tcp
sunrpc                        111/udp
auth                          113/tcp         authentication
sftp                          115/tcp
uucp-path       117/tcp
nntp                          119/tcp         readnews untp# USENET News Transfer Protocol
erpc                          121/udp         # Annex rpc listener
```

```
#
# UNIX specific services
#
exec                            512/tcp
biff                            512/udp         comsat
login                           513/tcp
who                             513/udp         whod
shell                           514/tcp         cmd# no passwords used
syslog                          514/udp
printer                         515/tcp         spooler# line printer spooler
talk                            517/udp         # old talk (bsd4.2, utx2.0)
ntalk                           518/udp         # new talk (bsd4.3, utx3.0)
efs                             520/tcp         # for LucasFilm
route                           520/udp         router routed
timed                           525/udp         timeserver
tempo                           526/tcp         newdate
courier                         530/tcp         rpc
conference      531/tcp                         chat
netnews                         532/tcp         readnews
netwall                         533/udp         # -for emergency broadcasts
uucp                            540/tcp         uucpd# uucp daemon
remotefs        556/tcp                         rfs_server rfs# Brunhoff remote filesystem

ingreslock      1524/tcp
```

/etc/protocols

This file provides mapping between protocol name and protocol number. It does not normally require modification in the field. **/etc/protocols**, as distributed, is

```
#
# Internet (IP) protocols
#
ip      0       IP      # internet protocol, pseudo protocol number
icmp    1       ICMP    # internet control message protocol
ggp     3       GGP     # gateway-gateway protocol
tcp     6       TCP     # transmission control protocol
egp     8       EGP     # exterior gateway protocol
pup     12      PUP     # PARC universal packet protocol
udp     17      UDP     # user datagram protocol
hmp     20      HMP     # host monitoring protocol
xns-idp 22      XNS-IDP # Xerox NS IDP
rdp     27      RDP     # "reliable datagram" protocol
```

For more information, see *protocols*(4).

/etc/hosts.equiv

The remote login and shell servers use an authentication scheme based on trusted hosts. The **hosts.equiv** file contains a list of hosts which are considered trusted and/or under a single administrative control. When you contact a remote login or shell server requesting service, the client process passes your name and the official name of the host on which the client is located. In the simple case, if the host's name is located in **hosts.equiv** and you have an account on the server's machine, service is rendered (i.e., you are allowed to login, or the command is executed). This equivalence of machines may be constrained by installing a

.rhosts file in the login directory. The *root* login is handled specially, bypassing the **hosts.equiv** file and using only the **/.rhosts** file.

To create a class of equivalent machines, the **hosts.equiv** file should contain the first name defined in **/etc/hosts** for those machines.

/etc/hosts.equiv, as distributed, is

```
localhost
sysname
SYSNAME
Sysname
```

NOTE: No space follows the name unless there is a user restriction. For example,

```
    localhost Rose Bill
```

allows only Rose and Bill on localhost.

/etc/rc.local

Most network servers can be automatically started up at boot time by the command file **/etc/rc.local** if they are in their presumed locations. The network servers include the following:

/etc/named	DOD domain name server
/etc/inetd	inet daemon
/etc/rwhod	system status daemon
/etc/routed	routing table management daemon
/etc/syslogd	error logging server
/etc/gated	advanced routing table management daemon

To have other network servers started up as well, commands of the following sort should be placed in the site dependent file **/etc/rc.local**:

```
if [ -f /etc/rwhod ]; then
/etc/rwhod ; echo -n 'rwhod'
```

Refer to the manual pages for details about their operation.

/etc/ftpusers

The **ftp** server may provide a loophole for interlopers if certain user accounts are allowed. The file **/etc/ftpusers** contains a list of users who are *not* permitted to connect to this host via **ftp**. The presence of this file prevents possibly dangerous interlopers from gaining access to the host, using generic user accounts.

/etc/ftpusers, as distributed, is

```
root
uucp
cshroot
```

There is another restriction on **ftp** connections: users with a * as the first character in their password field (in **/etc/passwd**) and users with no password (except the users **ftp** and **anonymous**, if allowed) cannot connect to the host via **ftp**. See *ftpd*(8) and Section 13.5, ''**/etc/ftpusers**,'' for more information.

/etc/gateways

If many local networks are interconnected by hosts running UTX/32 (forwarding packets), you may choose to run the routing table management daemon. **/etc/routed**, but see **/etc/gated.conf** below for an alternative. The routing daemon attempts to track changes in the network topology, keeping track of accessible networks and their associated gateways by using a variant of the XNS Routing Information Protocol. To connect routing daemon processes on detached networks or to propagate routing information about networks not running the routing daemon, the gateways file may be set up to reflect this information.

/etc/gateways consists of a series of lines of the form

`net` *network-name* `gateway` *gateway-name* `metric` *x* *routing-type*

The **net** keyword may also be **host** if the routing information is about a specific host (i.e., on a point-to-point style network like DECnet). The network and gateway names may be symbolic, in which case they are searched for in the host and network databases, or an Internet address of the form *a.b.c.d*. The metric specified is the number of hops to the destination network or host; it should be at least 2 (otherwise, presumably, the routing daemon would be able to recognize the network's presence by itself). If the **passive** keyword is specified for the routing type, it is presumed the gateway does not understand the routing table update protocol, and the routing daemon assumes the gateway is always up. If the routing type is specified as **active**, the routing daemon attempts to converse with it periodically, providing routing information and marking the gateway as down if it fails to elicit a response from it over a period of time. A sample **gateways** file follows. See *route*(8), *routed*(8), and *gateways*(4), if you intend to run the routing daemon.

```
net   bbn-net   gateway   10.3.0.72     metric  2  passive
net   sat-net   gateway   10.1.0.20     metric  2  passive
net   mit       gateway   10.0.0.77     metric  2  passive
net   cmu-net   gateway   cmu-gateway   metric  2  passive
net   128.10    gateway   10.2.0.37     metric  2  passive
net   wisc      gateway   10.0.0.94     metric  2  passive
net   lbl       gateway   lbl-csam      metric  2  active
net   sri       gateway   srijoyce      metric  2  active
```

/etc/gated.conf

gated is a routing daemon that handles multiple routing protocols and is meant to replace **routed**, **egpup**, and any routing daemon that speaks the HELLO routing protocol. See *gated*(8) for more information.

/etc/named.*

These files contain configuration information for the DOD domain name server, **named** (see *named*(8)). For more information about these files, see Section 13.18, ''**/etc/named.boot**'' and the *Name Server Operations Guide for BIND* in the UTX/32 supplementary documents.

/etc/scm_info

Each SCM requires entries in two files. The first file is the system **CONFIGURATION** file. The second file is **/etc/scm_info**, containing all the parameters for each SCM board, and for each link supported by each SCM board. The parameters in **/etc/scm_info** can be changed without regenerating the system.

The system is distributed with a sample **/etc/scm_info** file which can be used as distributed, with the exception that the **cmdaddr** field at one end of each SCM link (i.e., on one of the two machines involved in a link) must be changed. The **cmdaddr** field on one machine must be changed from 1 to 3.

The SCM, like the Ethernet interface, must be configured with **ifconfig** before being used. See *ifconfig*(8) for more information.

The command address, **cmdaddr**, as specified in **/etc/scm_info** is the X.25 address for the SCM port. The command address field must be unique with respect to the remote end of a link. The SCM only supports command addresses of 1 and 3.

The mode, while distributed as **unimode**, may be either **quadmode** or **unimode**. See *scm_info*(4) for more information.

The **/etc/scm_info** file, as distributed, is

```
scm sc0 at 0x7e70 unimode enabled
link 0 enabled

speed=56000
cmdaddr=1                  / change to 3 for machine 2
retrans=4                  / number of retrans allowed
t1=40                      / timeout in 1/4 sec quadts
t3=40                      / timeout in 1/4 sec quadts
mtu=1027
```

9.3.6 FTP Anonymous Accounts

The FTP server included in the system provides support for an anonymous FTP account. Due to the inherent security problems with such a facility, read this section carefully when considering such a service.

An anonymous account is enabled by creating a user **ftp.** When an anonymous account is used, a **chroot** system call is performed by the server. This restricts the client from moving outside the part of the file system where the user's **ftp** home directory is located. Because a **chroot** call is used, certain programs and files must be supplied to the server proce s. All directories and executable images must be unwritable.

The following directory setup is recommended:

```
# mkdir ~ftp
# cd ~ftp
# chmod 555 .; chown ftp .; chgrp ftp .
# mkdir bin etc pub
# chmod 555 bin etc
# chown ftp pub
# chmod 777 pub
# cd bin
# cp /bin/sh /bin/ls .
# chmod 111 sh ls
# cd ../etc
# cp /etc/passwd /etc/group .
# chmod 444 passwd group
```

When local users wish to place files in the anonymous area, the files must be placed in a subdirectory. In the setup here, the directory **~ftp/pub** is used.

9.3.7 Network Troubleshooting

Occasionally there are problems in complex network configurations. The following steps can be taken to locate the problem:

1. Check the network connections. On networks such as the Ethernet, a loose cable tap or misplaced power cable can result in severely deteriorated service. The **netstat** program is an aid in tracking down hardware malfunctions. In particular, look at the **–i** and **–s** options. See *netstat*(1) for more information.

2. Check the Ethernet connections. Several debugging tools are available for the SelBUS-based Ethernet. For example,

   ```
   # enfunc en? trace
   ```

   can be used to toggle packet tracing on and off. Packets are dumped to the console before and after they go out on the cable. You must be superuser to use **enfunc**.

   ```
   # enfunc en? stats
   ```

   obtains statistical information, such as packets coming in on the cable to the controller and being sent to the host.

   ```
   # enfunc en? sense
   ```

   obtains sense information such as collision counts. See *enfunc*(8) for more information.

   NOTE: The **enfunc** command produces a lot of output on a busy network.

3. Check for communication protocol problems. If a communication protocol problem exists, consult the protocol specifications and attempt to isolate the problem in a packet trace. The **SO_DEBUG** option may be supplied before establishing a connection on a socket, in which case the system will trace all traffic and internal actions (such as timers expiring) in a circular trace buffer. This buffer may then be printed with the **trpt** program. Almost all the servers distributed with the system accept a **–d** option forcing all sockets to be created with debugging turned on. See *trpt*(8) for more information.

4. Check for routing daemon malfunctions. If you believe the routing daemon is malfunctioning, its actions and all the packets sent and received may be printed out. To create a log file of routing daemon actions, supply a file name when the daemon is started, as in this example:

```
/etc/routed /etc/routerlog
```

Packets sent and received will be printed in the log file. To force full packet tracing, the **-t** option may be specified when the daemon is started up. See *routed*(8) for more information.

WARNING: On a busy network, checking for routing daemon malfunctions will generate almost constant output.

9.4 Sun Network Extensions

UTX/32 supports most of the features provided in the Sun networking extensions to BSD, including the Network File System (NFS™) and the Yellow Pages distributed database.

To use these features, you must have a network and a network interface for your machine (such as Ethernet). This section assumes that you have already completed the general networking setup described in Section 9.3, ``Networking.''

9.4.1 General Setup

To enable the Sun networking extensions described in the following sections, you will need to do the following:

1. Make sure that all remote hosts are listed in your **/etc/hosts** database.
2. Edit the **/etc/rc.local** file so that the correct daemons are started at boot time.
   a. So that the **portmap** and **sund** daemons will always to be started at multi-user boot time, make sure the following lines appear in the **rc.local** file:

```
if [ -f /etc/portmap ]; then
   /etc/portmap; echo -n ' portmap'
fi
if [ -f /etc/sund ]; then
   /etc/sund; echo -n ' sund'
fi
```

   The default configuration, as shipped on the distribution tape, starts these daemons.

   See *portmap*(8) and *sund*(8) for more information.

   b. If you want to run the Network File System, you should remove the pound signs (#) preceding the following lines in **rc.local**:

```
if [ -f /etc/nfsd ]; then
   /etc/nfsd 4; echo -n ' nfsd'
fi
if [ -f /etc/biod ]; then
   /etc/biod 4; echo -n ' biod'
fi
```

You will also need to set up the Network File System. See Section 9.4.2, "The Network File System (NFS)."

c. If you want to run Yellow Pages, you will need to remove the pound signs (#) preceding the following lines in **rc.local**:

```
if [ -f /usr/etc/ypserv -a -d /usr/etc/yp/'domainname' ]; then
   /usr/etc/ypserv; echo -n ' ypserv'
fi
if [ -f /etc/ypbind ]; then
   /etc/ypbind; echo -n ' ypbind'
fi
```

You will also need to set up the Yellow Pages database. See Section 9.4.3, "The Yellow Pages Database."

9.4.2 The Network File System (NFS)

The Network File System allows for the importing and exporting of file systems. Imported file systems are on remote machines, and exported file systems are local disk file systems that are to be shared with other machines.

Exporting

To export a file system, add its mount point name to the file **/etc/exports**. Note that this name must correspond to a mounted file system on the local machine. A remote machine can mount all or any part of a file system listed. See the examples below:

```
/tmp
/mnt
/utx2.1/ref
/usr
/usr.SUN68020
```

For more information, see *exports*(4).

Importing

To import a remote file system, follow these instructions:

1. Add an entry in **/etc/fstab** like one of those shown below:

   ```
   goat:/tmp/rabbit /tmp nfs rw,soft,noquota 0 0
   bear:/mnt /mnt nfs rw,soft,quota 0 0
   bear:/usr/src /usr/src nfs rw,soft,noquota 0 0
   mach:/utx2.1/ref /utx2.1/ref nfs ro,soft,noquota 0 0
   ```

   The rationale for entering the first example in **fstab** might be this: The machine, **rabbit**, needs more disk space, **goat** can provide it. A **/tmp** directory is created on **rabbit**. A **/tmp/rabbit** directory, dedicated to **rabbit**'s need for disk space, is created on **goat**. NFS enables the two directories to work in concert.

   The first field in each entry is the name of the remote machine, followed by the path name of the file system or directory on the remote machine that you want to mount. Note that you may mount a subdirectory of a remotely mounted file system.

   The second field is the name of the local directory to be given to the file system on the remote machine.

   The third field identifies to **mount** that this entry in **/etc/fstab** is for a networked file system.

   The fourth field lists **mount** options separated by commas (no spaces). The first can be **rw** or **ro** for read/write or read/only, the second should always be **soft**, and the third enables or disables remote file system quotas. Other options exist. See the *UTX/32 Network File System (NFS) Guide* for more information.

   The last two fields are dump and file system check flags. These should normally be set to zero (off).

2. Ensure that the directory you plan to mount exists on the local machine. If it does not, make it. In the **goat** example above, **goat**'s **/tmp/rabbit** directory is to become **/tmp** on the local machine. If **/tmp** does not exist on the local machine, create it by typing

   ```
   # mkdir /tmp
   ```

3. Mount the networked file system by typing

```
# cd /
# mount -a
```

9.4.3 The Yellow Pages Database

The Yellow Pages database provides a way to set up a shared set of administrative system files across multiple machines. If you use yellow pages, files such as **/etc/passwd**, **/etc/group**, **/etc/hosts**, and **/etc/networks** can be shared across a set of machines.

When running yellow pages, one machine acts as the master and other machines as slaves. Whenever shared files are modified, the master provides copies of the updated versions to the slaves. The files are kept in **dbm** format for fast access.

To set up yellow pages, follow these instructions:

1. Set your yellow pages domain name using **domainname** (see *domainname*(1)).
2. Edit the **/etc/rc.boot** file so that your domain name is set automatically each time the system boots.
3. On the system that will act as the master, ensure that the files to be shared exist and are in the proper format.
4. Type

   ```
   # /etc/yp/ypinit
   ```

   on the machine designated as the master and

   ```
   # /etc/yp/ypinit -s
   ```

   on machines designated as slaves.

   The **ypinit** command will ask a series of questions. When **ypinit** completes, the setup procedure is over, and yellow pages will go into operation after the next reboot. See *ypinit*(8). For more information, see the *UTX/32 Network File System (NFS) Guide*.

9.5 tip and cu

This section tells how to configure your system, dialin lines, and modems to work properly with the communications programs called **tip** and **cu**.

tip and **cu** are different names for the same communications program, but they have slightly different user interfaces. This program allows users to establish a connection with another system through a phone line or a serial communications line. Using the command **tip**, a system name or phone number can be specified and called; using the command **cu**, a phone number can be specified and called.

See *tip*(1) for more information. All further references to **tip** in the manual page and in this document also refer to **cu**.

To use **tip** to communicate with another machine, your system must have either a dialout modem connected to a phone line or a serial communications line connection to a remote machine. The dialout modem or serial communications line must be connected to a TTY line on your system. You must prepare the TTY line so that the software utilities will handle the communications correctly. Also, you must have an account on the remote system.

To configure a dialin or dialout modem, use the information in Section 9.5.1, ''Dialin Asynchronous Lines (Dialin Modems),'' Section 9.5.2, ''Dialout Asynchronous Lines (Dialout Modems),'' and Section 7.15.2, ''Editing **/etc/ttys**.'' You may also need to modify these files:

/etc/phones
/etc/remote
/etc/modcap

See the subsections with those titles in Chapter 13, ''Site-dependent System Files.'' If you are configuring a serial communications line between machines, see Section 7.15.2, ''Editing **/etc/ttys**.''

9.5.1 Dialin Asynchronous Lines (Dialin Modems)

This section gives information about configuring a dialin modem. You should follow the steps given in the following list. Additional information about editing the **/etc/ttys** file can be found in Section 7.15.2, ''Editing **/etc/ttys**.''

1. Wire the dialin lines to use the modem only when the phone line is dialed. Configure modem devices to use the Data Terminal Ready (DTR) signal from the RS-232 line instead of assuming the DTR setting is high.

2. Set the DTR to variable. This setting allows the modem to hang up when the system closes the line.

   Other modem switch settings vary according to the type of modem. These settings are usually controlled by switches on or inside the modem. Look at the **/etc/modcap** file for suggested modem switch settings. See Section 13.15, ''**/etc/modcap**,'' and your modem's user manual for additional information on modem settings.

3. Specify dialin lines as TTY lines with modem control in the system **CONFIGURATION** file.

   Dialin lines are typically named **/dev/ttyd***n*, where *n* is a decimal digit. Conventionally, the first dialin line is **/dev/ttyd0**, and the numbers increase sequentially. If a line is used as a dialin line, the TTY device entry for that line should not appear in the **/dev** directory. For example, if line 0 on multiplexor 1 is the first dialin line, it is named **/dev/ttyd0**, and there is no line named **/dev/tty08**. If line 7 on that multiplexor is assigned to the next dialin line, it is named **/dev/ttyd1**, and there is no **/dev/tty15**.

See Section 7.10, "Generating a Reconfigured Kernel," for more information on configuring modem control lines.

4. Add a line to the file **/dev/MAKEDEV.local** so the device entry will automatically be made any time the system is reconfigured. If you have source, make these changes in the source directory too. For example, to add a dialin line to 8-Line Asynchronous Communications Multiplexor 1, line 4, enter the command:

   ```
   mv /dev/tty12 /dev/ttyd0
   ```

   Every hard-wired line name should be wired to a modem and listed in **/etc/ttys**. However, the system is distributed without any dialin or dialout line names.

5. Check whether the dialin line **/dev/ttyd0** is available by entering:

   ```
   # ls -l /dev/ttyd0
   ```

   The response should be similar in form to

   ```
   crw-rw-rw- 1 adm      19,12  Nov  7 12:07 /dev/ttyd0
   ```

   In this example,

   ```
   19,12
   ```

   represents major device number 19, minor device number 12. Be sure the minor device number corresponds to the correct TTY line number.

To add more dialins, repeat the above process. After adding more lines, always re-edit **/etc/ttys** to record your new dialin lines.

9.5.2 Dialout Asynchronous Lines (Dialout Modems)

This section gives information about configuring a dialout modem. You should follow the steps given in the following list. Additional information about editing the **/etc/ttys** file can be found in Section 7.14.2, "Editing **/etc/ttys**."

1. Wire the dialout lines to use the modem only when the phone line connection is made. Set the Data Terminal Ready (DTR) to variable. This setting lets the modem hang up when the system closes the line.

2. Specify dialout lines as TTY lines with modem control in the system **CONFIGURATION** file.

   Dialout lines require two device entry names for each line. Each dialout line has a **/dev/cul***n* and **/dev/cua***n* device entry, where *n* is a decimal digit. **/dev/cul***n* is a character device with major and minor device numbers for the TTY line on which the dialout modem is located. **/dev/cua***n* is a hard link to the corresponding **/dev/cul***n*. (This means that **/dev/cua***n* and **/dev/cul***n* are two names for the same device.)

Conventionally, the first dialout line is named **/dev/cul0** and **/dev/cua0**, and the numbers increase sequentially for the following dialout lines. If a line is used as a dialout line, the TTY device entry for that line should not appear in the **/dev** directory. Thus, if line 0 on multiplexor 1 is the first dialout device, it would be named **/dev/cul0**. **/dev/cua0** would be linked to **/dev/cul0**, and there would be no **/dev/tty08**. If line 7 on that multiplexor is assigned to the next dialout line, it would be named **/dev/cul1**. **/dev/cua1** would be linked to **/dev/cul1**, and there would be no **/dev/tty15**.

See Section 7.10, ''Generating a Reconfigured Kernel,'' for more information on configuring modem control lines.

3. Since UTX/32 is distributed without dialout lines, add dialout lines to your system as superuser. For example, to add the first dialout line to 8-Line Asynchronous Communications Multiplexor (ACM) 2, line 0 (zero), change directory to **/dev** and enter the following commands:

```
# mv tty16 cul0
# ln cul0 cua0
# chmod 666 cul0
```

4. To check whether the first dialout line is available, enter

```
# ls -l /dev/cun0
```

The system should respond with lines similar in form to

```
crw-rw-rw- 2 adm      19,16  Nov  7 12:07 /dev/cul0
crw-rw-rw- 2 adm      19,16  Nov  7 12:07 /dev/cua0
```

In this example,

```
19,16
```

represents major device number 19, minor device number 16. Be sure the minor device number of **/dev/cul***n*, 16 in this example, corresponds to the correct TTY line number.

5. Edit the file **/etc/ttys** to remove the **/dev/tty***n* entry, again, 16 in this example, and replace it with

```
tty16 "/etc/getty D1200" dialup off "Dialup line #1"
```

To add more dialout lines, repeat the above process. After adding more lines, always re-edit **/etc/ttys**.

9.6 UUCP

This section describes how to install and use UUCP. In particular, the procedures for setting up **uucp** files are described in detail with examples. Install **uucp** when the system is in single-user mode and when all file systems are mounted. Only a system administrator logged in as **root** should modify the **uucp** directories and files.

9.6.1 Checking the Hardware Requirements

uucp is a set of programs allowing data transfer between two UNIX machines over modem lines or hardwire-connected TTY lines. Over modem lines, at least one of the machines must have an dialout port; the other must have a dialin port. It is better if both sites have both. Connections to local computers can be hardwired simply by connecting two TTY ports with the proper RS232C cable.

9.6.2 Checking the UUCP Directories

The **uucp** command set uses two main directories: **/usr/spool/uucp** and **/usr/lib/uucp**. Subdirectories under the **/usr/spool/uucp** directory must have read, write, and execute permissions 700 for **uucp** to function. To check the permissions, enter

```
# ls -ldg /usr/spool/uucp /usr/lib/uucp
```

Check the permissions in the output lines. The lines should be similar to these:

```
drwx------ 2 uucp uucp 3424 Dec 14 17:59 /usr/spool/uucp
drwx------ 3 uucp uucp  368 Dec  6 11:54 /usr/lib/uucp
```

Change the permissions if necessary, and create **/usr/spool/uucppublic** with mode 777.

9.6.3 Modifying the /etc/passwd and /etc/group files

Examine the **/etc/passwd** file. It should contain a user named **uucp** with a home directory of **/usr/spool/uucppublic**. Additionally, **/etc/passwd** should contain a user named **Uuucp** with a group 66, home directory of **/usr/spool/uucppublic**, and a shell program of **/usr/lib/uucp/uucico**. Each **uucp** site that calls can be given a separate account with a similar entry.

Note the following **/etc/passwd** sample:

```
uucp::66:66:Unix to Unix Copy:/usr/spool/uucppublic:
Uuucp::67:66:Unix to Unix Copy:/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

The entry in **/etc/group** should appear as

```
uucp::66:uucp,network,root,Uuucp
```

Enter passwords for users **uucp**, **Uuucp**, and any other **uucp** site login that you add (see *passwd*(1)). Other sites that communicate with yours must know the password of the **uucp** login to poll you.

9.6.4 Entering a Local Host Name for UUCP

uucp gets the local host name from **/bin/hostname**. Ensure that an entry for **/bin/hostname** is in your **/etc/rc.boot** file.

9.6.5 Checking Files in the /usr/lib/uucp directory

uucp requires the following files in the **/usr/lib/uucp** directory:

- **L–aliases**
- **L–devices**
- **L–dialcodes**
- **L.cmds**
- **L.sys**
- **USERFILE**

Example files are available in **/etc/system_setup/data**. The files with all uppercase names in **/etc/system_setup/data**, like **L-devices**, are provided as examples for corresponding files represented in lowercase in **/usr/lib/uucp**.

For additional information on **L–aliases**, refer to **uucp_tip** in **/etc/system_setup**. Also, see **/etc/modcap** file and the *modcap*(4) manual page for more information.

9.6.6 Describing the Devices in the L-devices File

The example file **L-devices** lists the call unit devices and hard-wired connections used by **uucp**. Use line entries in the **L-devices** example file as models for the entries you make in the **L-devices** file.

For example, line entries may be of this form, with five or six fields:

```
DIR tty07 0 1200 direct protocol
ACU cul0 cua0 1200 modem_type protocol
```

The first field is **DIR** or **ACU**. **DIR** is a direct-connect, hardwired TTY line; **ACU** is an automatic call unit.

The second field is the device entry name.

The third field is the call unit entry. In processes that open only either **cua0** or **cul0 (tip)**, correct locking of the device for exclusive use may not be ensured because other processes using both **cua0** and **cul0** (**uucp**) may have already locked the device.

The fourth field is the baud rate.

The fifth field is either **direct** or the *modem_type*. **direct** is the direct-connect lines. The *modem_type* entry may be one of the supplied drivers in the current release of BSD (VA212, Hayes, ...) or it may appear as **generic/***entry*, using a specific entry from the **modcap** file.

The sixth field is the optional **uucp** protocol to use (**g**, **t**, or **f**). In this release of UTX/32, **uucp** supports **g** (standard **uucp**), **f** (X.25 PAD 7-bit path), and **t** (TCP/IP) protocols.

9.6.7 Checking Device Modes

Verify that the modes of the device(s) used in **L–devices** are 666. For example,

```
# chmod 666 /dev/cul0
```

9.6.8 Checking Information in L-dialcodes and L.sys

The **L-dialcodes** file contains the location abbreviations used in the **L.sys** file. It may also contain the dial sequence needed to make the connection. A line entry in **L-dialcodes** has the form

```
xxx        5551000
```

The second field must not contain non-numeric characters unless the modems are configured to use them.

The **L.sys** file contains the information needed to make a call to another system:

- Host name
- When the host system may be called
- Device to call on
- Baud rate
- Phone number
- Expected login sequence

During system configuration, you must enable modem control for the TTY line attached to the dialout and auto-answer modem.

9.6.9 Examples of L.sys Entries

The following examples show line entries in the file **L.sys**. These examples use the following typographical conventions:

- Each entry is contained in one record. Lines are not broken as shown in the Hayes Smartmodem examples that follows.

- **\r** may be entered into the **/etc/modcap** file to represent <CR>. In the **L.sys** file, you must also use a **\r**.

Example 1

A general line entry in **L.sys** may look like this:

```
ccselc Any tty07 1200 tty07 "" CR ogin:-CR-ogin: Uuucp ssword:  ccselcuu
```

or like this:

```
ccselc Any DIR  1200 0 "" CR ogin:-CR-ogin: Uuucp ssword:  word
```

Example 2

If the Hayes Smartmodem or an equivalent is used, the entry line may look like this:

```
    ccselc Any ACU 1200 5550000  ogin:-CR-ogin: Uuucp
                    ssword: xxxxxxxx
```

In this example,

- **ccselc** is the name of the site to be called.
- **Any** means you can call any time.
- **ACU** is the automatic call unit at **1200** Baud.
- The use of a nonzero number in the phone field (**5550000**) turns on the table-driven modem software.
- After successful call completion by the table-driven modem software, the remaining fields in the line are processed.
- If the call was not completed, a second attempt is made.
- Assuming successful completion, **uucp** expects to receive the response **ogin:**.
- If it does not receive the response, a timer will go off and a <CR> will be generated.
- It again expects the **ogin:** response.
- When it receives the **ogin:** response, it sends **Uuucp** and expects **ssword:** as a response.
- When it receives **ssword:**, it sends out the password **xxxxxxxx**.

At this point a connection is made between the sites.

Example 3

Another Hayes Smartmodem example might read

```
ccselc Any ACU 1200 0 "" AT^M OK
                    ATDT5550000\r CONNECT \r ogin:-BREAK-ogin:
                     Uuucp ssword: xxxxxxxx
```

In this example,

- **ccselc** is the name of the site being called.
- **Any** means you can call any time.
- **ACU** is the automatic call unit at **1200** Baud.
- The **AT\r** gets the attention of the modem and responds with **OK.**
- The **ATDT5550000\r** tells the modem to dial the number when the connection is made.
- The modem responds with the **CONNECT** message.
- The next **\r** is a carriage return to the remote system.
- **uucp** then expects to receive the response **ogin:**.
- If it does not, a timer will go off and a BREAK will be generated.
- It again expects the **ogin:** response.
- When it gets the **ogin:** response, it sends the user name **Uuucp** and expects **ssword:** as a response.
- When it receives **ssword:**, it sends out the password **xxxxxxxx**.

At this point a connection is made between the sites.

You may wish to refer to the **system_setup** files for more examples.

9.6.10 Setting Up the Files L.cmds and USERFILE

The **L.cmds** file indicates which commands may be executed on the local system. Entries might look like this:

```
rmail
cp
```

The **USERFILE** contains a list of authorized users and the directory from which transfers can be made. A sample file might contain

```
dan, /usr/dan
, /
```

The entry format is:

username, *system_name directory_name*

The word before the comma is the user's name. If there is no word before the comma, anyone can use the system. The directory after the comma is the directory from which the transfer can be made. In the example, **dan** is allowed to transfer from the directory **/usr/dan**, and anyone else is allowed to transfer from /. See *system_setup*(8) for examples of these files and other **uucp** configuration files.

9.6.11 Using uucp

Once the necessary files are set up correctly, both on the calling system and the called system, files can be transferred between systems. The calling sequence is

`uucp` *source file destination file*

To send the file to a remote system, the filenames must be either in the form of pathnames on the local system or in the form:

system name!*path name*

Note that the ! is a special character to **csh**. When using **csh**, you must escape this character with a backslash (\!).

The user of **uucp** must own the file being sent and must have write permission in the destination directory. If the user does not have a login on the remote machine, the destination path name must be omitted. In this case, the file will be sent to the directory **/usr/spool/uucppublic** and will be owned by the user **uucp**.

uucp gives the user no indication of its status while the file transfer is being attempted. See *uucp*(1C) for information about **uulog**, a logging facility built into **uucp**.

10 The Accounting System

This section describes how to implement, modify, and use the UTX/32 system accounting. These topics are covered:

- The purpose and structure of the accounting system
- Setting up the system
- Automated daily accounting procedures
- The main daily accounting shell procedure, **runacct**
- How to detect and recover from accounting errors
- How to fix damaged accounting files
- The accounting directories and files
- Keys to accounting tables and reports

10.1 Overview

The UTX/32 accounting system is based on the utilities from both BSD and System V accounting utilities. This system

- Collects per-process utilization data
- Records connect sessions
- Monitors disk utilization
- Charges fees to specific logins

Its design permits customized installation for your site's demands and contains many options that may be modified. (Modification requires root privileges.) A set of shell scripts and programs reduce this accounting data to summary files and reports.

The accounting system is complex. Its maintenance requires a thorough knowledge of the constituent programs and scripts. Administrators should examine these programs and scripts, study the manual pages for accounting utilities, and keep a printed copy of the shell procedures handy.

The accounting system has a high operating overhead. It uses large amounts of system time and storage space, including at least 500 blocks on the **/usr** file system. Before implementing accounting, you must determine whether the information it provides will be of use to your site.

The following sections describe System V's accounting utilities and implementation. The BSD accounting utilities are described in *accton*(8), *sa*(8), and *lastcomm*(1).

The following list summarizes how the accounting system collects and records data:

- When a UTX/32 process terminates, the accounting system writes a record in **/usr/adm/pacct** in a **struct acct** (see **/usr/include/sys/acct.h**). These records provide data for reports about system usage. of all input/output files.)
- The **login** and **init** programs record user login sessions by writing records in the file **/usr/adm/wtmp**. This file also records date changes, reboots, and shutdowns. Records in **/usr/adm/wtmp** provide data for reports about user connect time.
- The disk utilization program **/usr/5lib/acct/acctdusg** records disk usage by login name.
- The **/usr/5lib/acct/chargefee** shell script calculates fees for file restores and other services performed for specific logins.

The following list summarizes the production of accounting reports:

- Each day, **cron** executes the **/usr/5lib/acct/runacct** shell script to reduce accounting data and produce summary files and reports.
- **crontab** invokes a simple shell that regularly prints both daily and weekly reports.
- You may execute the **/usr/5lib/acct/monacct** procedure on a monthly or fiscal basis. This saves and restarts summary files, generates a report, and cleans up the **/usr/5lib/acct/sum** directory. You may use these saved summary files to charge users for system usage.

If accounting does not have enough space to run, if a general user attempts to run the accounting program, or if some other error occurs, the users named **root** and **adm** receive mail indicating what went wrong. System administrators should check mail for **root** and **adm** regularly to ensure that the accounting system is running smoothly.

10.2 Setting Up Accounting

If you are unfamiliar with the directory structure and file functions of the accounting components, read a description in Section 10.3, ``Accounting System Components,'' before setting up your accounting system.

You must modify four files to automate the operation of the UTX/32 accounting system. Perform the following as superuser:

1. Insert the following lines in the **/etc/rc.local** file:

```
echo "Starting accounting." > /dev/console
/usr/5lib/acct/startup &
```

This enables accounting at system boot time and cleans up obsolete accounting files. Check the free space in **/usr**; remember that accounting needs 500 free blocks to function correctly.

2. Add this line to **/etc/shutdown** to turn off accounting before the system shuts down:

   ```
   /usr/5lib/acct/shutacct
   ```

   NOTE: Customers with only binary distribution cannot turn off accounting.

3. Make entries in **/usr/lib/crontab** so that **cron** will automatically execute various accounting shell procedures. For more information on **/usr/lib/crontab** and **cron**, see Section 13.41, ``**/usr/lib/crontab**,'' and *cron*(8).

   Examples of such entries are

   ```
   2 4 * * *         root   /usr/adm/dailyacct.sh
   2 12 * * *        root   /usr/adm/noonacct.sh
   5 6 1 * *         root   /usr/adm/monthlyacct.sh
   3 * * * *         root   /usr/adm/hourlyacct.sh
   ```

   These entries would execute the following shell procedures:

dailyacct.sh	**/usr/5lib/acct/dodisk**
runacct.sh	**/usr/5lib/acct/runacct**
noonacct.sh	**/usr/5lib/acct/acctdusg**
monthlyacct.sh	**/usr/5lib/acct/monacct**
hourlyacct.sh	**/usr/5lib/acct/ckpacct**

   Not all of these entries are required. Choose the subset that best meets the accounting demands and space limitations of your system.

   For more information on these shell procedures, see Section 10.3, ``Accounting System Components.''

4. Enter the following line in **crontab** to facilitate monthly merging of accounting data:

```
15 5 1 * * /bin/su -adm -c /usr/5lib/acct/monacct
```

This entry allows **monacct** to clean up all daily reports and one monthly total accounting file in the **fiscal** directory. It takes advantage of the default action of **monacct** that uses the current month's date as the suffix for the file names.

Notice that the entry executes at a time that allows **runacct** time to complete. On the first day of each month, **monacct** will create monthly accounting files with the entire month's data.

5. Set the PATH shell variable in **/usr/adm/.profile** to

```
PATH=/usr/5lib/acct:/bin:/usr/bin
```

10.3 Accounting System Components

The following subsections discuss the files and directories used for system accounting. You may implement a subset of these accounting components to provide an accounting system tailored to your computing environment. System programmers may modify the code in these files to further customize the system.

10.3.1 Directories

/usr/5lib/acct This directory contains all the C language programs and shell procedures necessary to run the accounting system. The accounting system uses the login name **adm**.

/usr/adm/acct This directory contains the active data collection files. The **nite** subdirectory contains files that are used daily by the **runacct** procedure. The **sum** subdirectory contains the cumulative summary files updated by **runacct**. The **fiscal** subdirectory contains periodic summary files created by **monacct.**

10.3.2 Input and Output Files

/usr/adm/fee **chargefee** creates this file to assign a number of units to each login name for special services, such as file restore and tape manipulations. **chargefee** writes a record to **/usr/adm/fee** each time it is invoked. This record is later merged with other accounting records by **runacct**. Since sites often do not charge fees, the default configuration does not charge fees. See *acctsh*(8).

/usr/5lib/acct/holidays
This ASCII input file, tailors prime and nonprime connect time to your company's standards. You must edit this file to show current year, start of prime time (usually 8:00 AM [0800]), start of nonprime time (usually 5:00 PM [1700]), and holiday

information. Additional information about the **holidays** input file is given in Section 10.8, ''Prime Time and Holidays.''

/usr/adm/pacct

This file contains records written by the kernel. It is created when accounting is turned on. The output data is in the **tacct** format. Many of the accounting utilities use **/usr/adm/pacct** as input.

/usr/src/5src/cmd/acct/tacct.h

The accounting header file **tacct.h** defines the format **tacct** (total accounting record). Most of the accounting programs take their input from standard input or from files with the string **tacct** in their names. These files contain data in the **tacct** format. See *acct*(4) in the *System V Programmer's Reference Manual* for a description of **struct tacct**.

/usr/adm/wtmp

The **login** and **init** programs record connect sessions by writing records into this file. **/usr/adm/wtmp** also records date changes, reboots, and shutdowns. Records in **/usr/adm/wtmp** provide data for reports about user connect time. The file provides input to the accounting programs **acctcon1**, **acctwtmp**, and **wtmpfix**. See *wtmp*(4) for the structure of this file. **shutacct**, which should be run prior to system shutdown, outputs messages to **/usr/adm/wtmp**.

10.3.3 Accounting Scripts and Programs

acctdusg The **acctdusg** shell procedure performs disk resource consumption by the user ID. To process the disk-total accounting records, you must run **dodisk** and **acctdusg** prior to executing **runacct**. See *acct*(8) and *acctsh*(8) for more information.

chargefee The optional **chargefee** program (see *acctsh*(8)) bills users for file restores and other services. It adds records to **/usr/adm/fee**; these records are picked up and processed by the next execution of **runacct** and merged into the total accounting records.

NOTE: The **chargefee** program, if used, must be run before **runacct**.

Executing the command

```
# /usr/5lib/acct/chargefee login_name fee-unit
```

creates the following **/usr/adm/fee** file.

```
3  login_name   0 0 0 0 0 0 0 0 0 0   fee_unit
```

The user ID will appear in the file. In this example, the user ID is 3. Also, administrators must decide on the number of *fee_units* to charge each user and what each unit represents. The fee-unit value is inserted into the file.

ckpacct The **ckpacct** (see *acctsh*(8)) program checks that the accumulation of data in the file **/usr/adm/pacct** does not get too large. If the file grows past the default of 1000 blocks, **turnacct switch**, a routine within **ckpacct**, executes and turns off accounting until a cleanup is done.

dodisk The **dodisk** shell script (see *acctsh*(8)) lists disk accounting. It calls **acctdisk** (see *acct*(8)) to get the number of disk blocks in use by each UID. Note that **dodisk** executes with superuser privileges so that directory searching is not roadblocked.

monacct The **monacct** script (see *acctsh*(8)) generates a monthly accounting report, typically on the first day of each month. It generates summary files in the directory **/usr/adm/acct/fiscal** and cleans up all prior daily reports in **/usr/adm/acct/sum**.

Although the **monacct** program will clean up some of the report files, you should occasionally check the accounting subdirectories in **/usr/adm** and **/usr/adm/acct** as well as other directories in **/usr**, such as spool directories. The file **/usr/adm/wtmp** becomes especially large over time since it accumulates connect session records generated by **init** and **login**. Reboots, halts, and shutdowns also append messages to this file.

10.3.4 Manual Pages for the Accounting System

The *acctsh*(8) manual page describes the following accounting scripts and programs:

/usr/5lib/acct/chargefee	**/usr/5lib/acct/prdaily**
/usr/5lib/acct/ckpacct	**/usr/5lib/acct/prtacct**
/usr/5lib/acct/dodisk	**/usr/5lib/acct/runacct**
/usr/5lib/acct/lastlogin	**/usr/5lib/acct/shutacct**
/usr/5lib/acct/monacct	**/usr/5lib/acct/startup**
/usr/5lib/acct/nulladm	**/usr/5lib/acct/turnacct**
/usr/5lib/acct/prctmp	

The *acct*(8) manual page describes the following commands:

acctdisk	**accton**
acctdusg	**acctwtmp**

Each of the following commands has its own manual pages in the *UTX/32 System Administrator's Reference Manual*:

acctcms	**acctcon**	**acctprc**
acctcom	**acctmerg**	

All commands perform specific tasks as necessary. See the appropriate manual pages for complete descriptions of each program and its connection with other accounting components.

10.4 Daily Operation

Once you set up an accounting system, its automated components generate reports about various aspects of system usage. You should print out the reports and remove old on-line files on a regular basis.

Each morning, you should run **/usr/5lib/acct/prdaily** to print the previous day's accounting report.

> NOTE: Clean up the report files regularly to prevent the **/usr/adm/acct/sum** directory from using all the space on the **/usr** file system.

Except for printing reports and cleanup, all aspects of accounting are performed automatically and are usually initiated by **cron**. The following list describes the automated components for daily operation of accounting.

- The program **cron** runs the **ckpacct** procedure hourly to check the size of **/usr/adm/pacct**. If the file grows past the default size of 1000 blocks, the routine **turnacct switch** within **ckpacct** turns off accounting. Running **ckpacct** is not absolutely necessary. However, **ckpacct** produces several small **pacct** files, which simplifies restarting **runacct** after a failure and checks the number of free blocks in **/usr**.
- The **chargefee** program, if used, bills users for file restores and other services.
- **cron** executes **runacct** each night and processes the following active accounting files:

  /usr/adm/pacct

  /usr/adm/wtmp

  /usr/adm/acct/nite/diskacct

/usr/adm/fee

- The **runacct** shell procedure produces command summaries and usage summaries organized by login name.
- When the system shuts down using the **shutdown** command, the **shutacct** shell procedure writes a shutdown record into **/usr/adm/wtmp** and turns off process accounting.

10.5 The runacct Shell Procedure

This section describes **runacct**, the main daily accounting shell procedure, and the files it produces.

runacct performs all process, connect, and command accounting functions at an indicated time, merges all total accounting records, generates cumulative summary files, and produces a daily report. See *runacct*(8) for more information.

The following files produced by **runacct** in the directory **/usr/adm/acct** collect data about particular aspects of the system:

nite/lineuse
User statistics for each terminal line on the system. **lineuse** is the output of **acctcon**, which reads the **wtmp** file.

nite/dayacct
Total accounting file for the previous day, in **tacct** format.

sum/tacct
Accumulation of each day's **nite/daytacct** which can be used for billing purposes. The **monacct** procedure, described in Section 10.3, ``Accounting System Components,'' restarts **sum/tacct** each month or fiscal period.

sum/daycms
Summary of daily commands. **daycms** is the output of the **acctcms** program. The ASCII version of this file is **nite/daycms**.

sum/cms
Accumulation of each day's command summaries. **cms** is restarted by **monacct**. The ASCII version is **nite/cms**.

sum/loginlog
Record of the last time each login was used. **loginlog** is the output of the **lastlogin** shell procedure.

sum/rprt.MMDD
Copy of **prdaily** output saved on each execution of **runacct**.

10.6 Damage Control

The following subsection discusses recovering from failure and restarting the accounting system.

10.6.1 Safeguards

In the event of errors, **runacct** takes care not to damage files. A series of protection mechanisms attempt to detect errors file, provide intelligent diagnostics, and terminate processing in such a way that **runacct** can be restarted with minimal intervention. **runacct** records its progress by writing descriptive messages into the **active** file. It also writes all diagnostics output generated during its execution into **fd2log**. (Files used by **runacct** are assumed to be in the **/usr/adm/acct/nite** directory unless otherwise noted.)

Multiple invocations of accounting (for example, if **cron** starts twice) will corrupt accounting records. To prevent multiple invocations, **runacct** opens the files **lock** and **lock1**. When **runacct** is improperly invoked, these lock files exist and prevent a second invocation. Also, the **lastdate** file contains the month and day **runacct** was last invoked and prevents more than one execution per day. If **runacct** detects an improper invocation, it writes a message to **/dev/console**, sends mail to users named **root** and **adm**, removes the locks, saves the diagnostic files, and terminates execution.

NOTE: If system failure occurs while running **runacct**, you must manually remove the lock files, **lastdate** file, and **statefile** with the **rm** command.

10.6.2 Executing runacct

To start **runacct** for the first time each day, enter the command **runacct** without arguments. For example,

```
# nohup runacct >& /usr/adm/acct/nite/fd2log &
```

To restart **runacct**, the argument *mmdd* is necessary. This argument specifies the month and day for which **runacct** will rerun the accounting. For example, 1225 is December 25. To restart **runacct** on June 1, enter:

```
# nohup runacct 0601 >& /usr/adm/acct/nite/fd2log &
```

The entry point for processing is based on the contents of **statefile**. To override **statefile**, include the desired state on the command line. See Section 10.5.4 ''Restarting **runacct**'' for more information. For example, to restart **runacct** on June 1 at the specific state named FEES, enter:

```
# nohup runacct 0601 FEES >& /usr/adm/acct/nite/fd2log &
```

NOTE: Normally, do not restart **runacct** in the SETUP state. Instead, run SETUP manually and restart with the command

```
# runacct mmdd WTMPFIX
```

Also, if **runacct** failed in the PROCESS state, remove the last **ptacct** file, because it will be incomplete.

10.6.3 Recovering from Failure

The **runacct** procedure can fail for a variety of reasons, such as a system crash, lack of /**usr** space, or a corrupted **wtmp** file. In the event of a **runacct** failure, first, check the **active.MMDD** file for error messages. Next, if the **active** file and lock files exist, check **fd2log** for error messages.

Following are error messages produced by **runacct** and the recommended recovery actions:

`ERROR: locks found, run aborted`
: The files **lock** and **lock1** were found. These files must be removed before **runacct** can restart.

`ERROR: acctg already run for <date>: check /usr/adm/acct/nite/lastdate`
: The date in **lastdate** and today's date are the same. Remove **lastdate**.

`ERROR: turnacct switch returned rc=<x>`
: Check the integrity of **turnacct** and **accton**. The **accton** program must be owned by **root** and have the **setuid** bit set.

`ERROR: Spacct<x>.MMDD already exists`
: The file setups are probably already run. Check the status of files, then run setups manually.

`ERROR: /usr/adm/acct/nite/wtmp.MMDD already exists, run setup manually`
: Self-explanatory.

`ERROR: wtmpfix errors see /usr/adm/acct/nite/wtmperror`
: **wtmpfix** detected a corrupted **wtmp** file. Use **fwtmp** to correct the corrupted file.

`ERROR: connect acctg failed: check /usr/adm/acct/nite/log`
: The **acctcon1** program encountered a bad **wtmp** file. Use **fwtmp** to correct the bad file.

`ERROR: Invalid state, check /usr/adm/acct/nite/active`
: The file **/usr/adm/acct/nite/statefile** is probably corrupted. Check **statefile** and read the active state before restarting.

10.6.4 Restarting runacct

To allow **runacct** to restart, processing breaks down into separate reentrant states by using a **case** statement inside an endless **while** loop. Each state is one case of the **case** statement. As each state completes, it updates the **/usr/adm/acct/nite/statefile** to reflect the next state. In the next pass through the **while** loop, **statefile** is read and the **case** falls through to the next state. When **runacct** reaches the CLEANUP state, it removes the locks and terminates. **statefile** then contains the word COMPLETE.

The **runacct** states execute in the following order:

SETUP

Executes the routine **turnacct switch** within **ckpacct**.

In the directory **/usr/adm**, the process accounting file **pacct***x* moves to **acct/nite/pacct.MMDD**. The **wtmp** file of login information, with the current time appended it, moves to **acct/nite/wtmp.MMDD**.

WTMPFIX

The user checks the **wtmp** file in the **nite** directory with the **wtmpfix** program.

Some date changes may cause **acctcon1** to fail, so **wtmpfix** corrects the time stamps in the **wtmp** file as a date change record appears.

CONNECT1

Writes connect session records to **ctmp** file in the form of **ctmp.h**, creates the **lineuse** file, and creates the **reboots** file that shows all the boot records found in the **wtmp** file.

CONNECT2

Converts **ctmp** to **ctacct.MMDD**, which contains accounting records in **tacct** format.

PROCESS

Uses the **acctprc1** and **acctprc2** programs to convert **/usr/adm/Spacct***x***.MMDD**, the process accounting files, into **ptacct***x***.MMDD** accounting records.

The **Spacct** and **ptacct** files are correlated by number so that if **runacct** fails, **Spacct** files will not be processed unnecessarily. One precaution should be noted—when restarting **runacct** in this state, remove the last **pacct** file, because it will be incomplete.

MERGE

Merges process accounting records and connects accounting records to form **daytacct**.

FEES

Merges any ASCII **tacct** records from the file **fee** into **dayacct**.

DISK
: Merges **disktacct** files with **daytacct** the day after the **sdisk** procedure runs.

MERGETACCT
: Merges **daytacct** with **sum/tacct** and produces a cumulative accounting file.

  Each day, **dayacct** is saved in **sum/tacctMMDD** so that **sum/tacct** can be recreated if it becomes corrupted or lost.

CMS
: Merges today's command summary with the cumulative command summary file **sum/cms** and produces ASCII and internal format command summary files.

USEREXIT
: Runs any installation-dependent (local) accounting programs.

CLEANUP
: Removes temporary files, runs **prdaily**, saves its output in **sum/rprtMMDD**, and removes locks.

COMPLETE
: Exits **runacct**.

10.7 Fixing Corrupted Files

Unfortunately, the accounting system is not fool-proof. Occasionally, a file will become corrupted or lost. Some of the files can simply be ignored or restored from a system backup tape. However, certain files must be fixed to maintain the integrity of the accounting system.

10.7.1 Fixing wtmp Errors

The **wtmp** files seem to cause the most problems in the day-to-day operation of the accounting system. When the date is changed and the UTX/32 system is in multi-user mode, a set of date change records is written into **/usr/adm/wtmp**. The **wtmpfix** program adjusts the time stamps in the **wtmp** records when a date-change is encountered. Some combinations of date changes and reboots, however, will slip through **wtmpfix** and cause **acctcon1** to fail. When WTMPFIX is in **statefile** and repair is necessary, the following steps show how to patch the **wtmp** file using the **fwtmp** command:

```
# cd /usr/adm/acct/nite
# fwtmp < wtmp.MMDD > xwtmp
# vi xwtmp
```

In the edit session, delete corrupted records or delete all records from the beginning of the file to the date change. Exit the edit session and enter this command line:

```
# fwtmp -ic < xwtmp > wtmp.MMDD
```

If the **wtmp** file is severely damaged, create a null **wtmp** file to prevent any incorrect charging of connect time during repairs. Since **acctprc1** will not be able to determine which login owned a particular process, it will record any incorrect charge in the null **wtmp** file.

10.7.2 Fixing tacct Errors

If the installation uses the accounting system to charge users for system resources, the integrity of **sum/tacct** in **/usr/adm/acct** is important. Occasionally, unusual **tacct** records will appear with negative numbers, duplicate user IDs, or an impossible user ID.

If unusual **tacct** records appear, first check **sum/tacctprev** with **prtacct**. If the file appears correct, patch the latest **sum/tacct.MMDD** and then recreate **sum/tacct**.

```
# cd /usr/adm/acct/sum
# acctmerg -v < tacct.MMDD > xtacct
# vi xtacct
```

While in the edit session, remove the bad records and write duplicate UID records to another file. Then, exit the edit session and enter these commands:

```
# acctmerg -i < xtacct > tacct.MMDD
# acctmerg tacctprev < tacct.MMDD > tacct
```

Remember that the **monacct** procedure removes all the **tacct.MMDD** files; therefore, **sum/tacct** can be recreated by merging these files.

10.8 Prime Time and Holidays

acctcon1 and **acctprc1** use the **pnpsplit** subroutine to determine the difference between prime and nonprime time. By default, prime time is defined as 9:00 AM to 5:00 PM, Monday through Friday. Nonprime time is defined as all other hours and the entire day for those days listed in the **holidays** structure in **pnpsplit.c.** The holidays listed are accurate for Gould, Inc., for the year of this release of UTX/32.

Every year on the day after the last holiday of the calendar year, the following message will be printed on the system console terminal and appear in **log**:

```
***UPDATE pnpsplit WITH NEW HOLIDAYS***
```

This message will continue to be sent each time the accounting runs until **pnpsplit**, **acctcon1**, and **acctprc1** are recompiled. Take the following steps before you recompile these programs:

1. Edit **pnpsplit.c** in the directory **/usr/src/5src/cmd/acct/lib** to change the **thisyear** variable to the new year.
2. Update the **holidays** table in **/usr/5lib/acct** to reflect the new holidays. The numeric entry in the structure is the day of the year. For example, New Year's Day (January 1) is entered as 1. The following is an excerpt from the **/usr/5lib/acct/holidays** file:

```
*   Curr      Prime       Nonprime
*   Year      Start       Start
    1987      0800        1700
*   Day of    Calendar    Company
*   Year      Date        Holiday
*
       1      Jan 1       New Year's
       2      Jan 2       day after New Year's
     147      May 27      Memorial Day
```

10.9 Accounting Reports

With a fully implemented accounting system, each invocation of **runacct** generates four basic reports.

- Daily Report
- Daily Usage Report
- Daily and Monthly Command Summaries
- Last Login Report

These reports summarize connect accounting usage by person on a daily basis, command usage reported by daily and monthly totals, and a report of the last time each user was logged in. Samples of these reports are shown at the end of this section.

The following subsections describe the reports and their data. Occasionally, tips are presented on what actions to take due to certain report results.

10.9.1 Daily Report

The *daily report* consists of two parts. The first part of the report is a **from/to** banner. The information on this banner include the time the last accounting report was generated and the time the current accounting report was generated. The second part of the report is a log of system reboots, shutdowns, power fail recoveries, and all other records dumped into **/usr/adm/wtmp** by the **acctwtmp** program.

The second part of the report presents information about line utilization. The TOTAL DURATION tells how long the system was in multi-user mode. The columns are:

LINE	Terminal line or access port
MINUTES	Total number of minutes the line was in use during the accounting period
PERCENT	Total number of MINUTES that the line was in use divided by TOTAL DURATION
# SESS	Number of times the port was accessed for a login session
# ON	Number of logins
# OFF	Number of logoffs

The following is a sample portion of a daily report:

```
Jan  5 00:14 1987  DAILY REPORT FOR unix Page 1

from Sun Jan  4 01:01:21 1987
to   Mon Jan  5 00:13:08 1987

TOTAL DURATION IS 1392 MINUTES

LINE         MINUTES      PERCENT      # SESS       # ON         # OFF
ttyp3        0            0            3            2            3
ttyp7        45           3            3            2            3
ttyp8        597          43           6            3            6
ttyp1        215          15           4            3            4
 ...          ...          ...         ...          ...          ...
 TOTALS      1529         --           45           28           45
```

10.9.2 Daily Usage Report

The *daily usage report* gives a user-by-user breakdown of system resource utilization. Its data is tagged as follows:

UID
User ID number

LOGIN NAME
User login name

Shared accounts can exist for certain projects, and more than one login name can exist for a single user ID. LOGIN NAME identifies specific users.

CPU(MINS)
Amount of time the user's process used the CPU

This category breaks down into PRIME and NPRIME (nonprime) time. The accounting system defines this breakdown in the **holidays** array of the accounting library function **pnpsplit**.

KCORE-MINS
: Number of kilobyte segments of memory used per minute, divided into PRIME and NPRIME amounts

CONNECT(MINS)
: Amount of time the user was logged into the system, divided into PRIME and NPRIME amounts

  If this time is high and the column # OF PROCS is low, then this user may be termed a *line hog*. This person logs in but hardly uses the terminal.

DISK BLOCKS
: Output of the disk-accounting program in number of blocks used

  This number is merged into the total accounting record. The **acctdusg** program handles this type of disk accounting.

# OF PROCS
: Number of processes invoked by the user

  Watch this column for large numbers, which could indicate that a user has a shell procedure in an infinite loop. For example, a **crontab** entry could try to execute a user's **.profile** via **su**. Unfortunately, this prompts for a terminal type and sits in an endless loop trying to read from the terminal. Since there is no terminal when **cron** is executing a process, encourage preventive coding in each user's **.profile** at your site.

# OF SESS
: Number of times the user logged in to the system

# DISK SAMPLES
: Number of times disk accounting was run to obtain the average number of DISK BLOCKS listed earlier

FEE
: Accumulation of units charged for special services by the **chargefee** shell procedure

  The **chargefee** procedure charges a user for special services performed by the operations staff. Many sites do not use this field in their total accounting record.

The following is a sample daily usage report.

```
Jan  5 00:14 1987  DAILY USAGE REPORT FOR unix Page 1
        LOGIN    CPU(MINS)        KCORE-MINS     CONNECT (MINS) DISK      # OF     # OF      # DISK  FEE
UID     NAME     PRIME   NPRIME  PRIME    NPRIME PRIME    NPRIME BLOCKS  PROCS    SESS      SAMPLES
0       TOTAL       0    274     6        70959  0        1529   0       21483    17        0
0       root        0    8       0        958    0        0      0       2120     0         0
1       daemon      0    0       0        66     0        0      0       44       0         0
4       dave        0    35      0        9698   0        1      0       980      1         0
25      raxs        0    29      0        11356  0        323    0       1988     1         0
255     notes       0    10      0        1597   0        0      0       1881     0         0
```

10.9.3 Daily and Monthly Command Summaries

The *daily command summary* and *monthly total command summary* are basically the same except for the time interval reported. The daily command summary reports on the current accounting period, while the Monthly Total Command Summary shows the data accumulated since the last invocation of **monacct.**

These reports show the most frequently used commands.

The reports are sorted by TOTAL KCOREMIN described below, This is an arbitrary yardstick, but often a good one for calculating the drain on a system.

COMMAND NAME
: Command (program) names

  All shell scripts are grouped under the name **sh**, because only object modules are reported by the process accounting system. You should monitor the frequency of programs called **a.out**, **core**, or any name that does not seem quite right. Use **acctcom** as a tool to determine who executed a suspiciously named command. Such occurrences are especially suspect if superuser privileges were used.

NUMBER CMDS
: Number of invocations of the program

TOTAL KCOREMIN
: Number of kilobyte segments of memory used by a process per minute of runtime

TOTAL CPU-MIN
: Total minutes of CPU processing time accumulated by the program

TOTAL REAL-MIN
: Total wall-clock minutes accumulated by the program

MEAN SIZE-K
: The mean derived from TOTAL KCOREMIN divided by NUMBER CMDS

MEAN CPU-MIN
: The mean derived from NUMBER CMDS divided by TOTAL CPU-MIN

HOG FACTOR
: The TOTAL CPU-MIN divided by TOTAL REAL-MIN. This gives a relative measure of the total available CPU time consumed by the process during its execution.

CHARS TRNSFD

Number (positive or negative) of characters accessed by **read** and **write** system calls.

BLOCKS READ

Number of physical block reads and writes performed by a process

The following is a sample portion from a daily command summary.

```
Jan  5 00:14 1987  DAILY COMMAND SUMMARY Page 1

                                        TOTAL COMMAND SUMMARY
COMMAND    NUMBER       TOTAL       TOTAL      TOTAL     MEAN     MEAN     HOG           CHARS       BLOCKS
NAME         CMDS    KCOREMIN     CPU-MIN   REAL-MIN   SIZE-K  CPU-MIN  FACTOR          TRNSFD         READ

TOTALS      21483    69850.69      273.68   32863.05   255.23     0.01    0.01       981745152       218991

ccom          985    22947.60       66.81      93.54   343.48     0.07    0.71        52926912         7604
cpp          1003     6828.30       32.16      53.87   212.35     0.03    0.60       160399376        34352
xemacs          6     6498.92        6.28     540.62  1034.36     1.05    0.01         9926656         3147
npas          840     5166.63       23.50      31.21   219.87     0.03    0.75        39727104         9197
c2            872     4047.09       23.96      27.93   168.88     0.03    0.86        34093136         5700
csh          2332     3725.47       18.42   13137.12   202.26     0.01    0.00         8481644        19007
make           85     2658.07        7.29     240.52   364.54     0.09    0.03         7211873         9308
vi            131     2107.51        7.34    3547.76   287.17     0.06    0.00        20450176         7202
```

The following is a sample portion from a monthly total command summary.

```
Jan  5 00:14 1987  MONTHLY TOTAL COMMAND SUMMARY Page 1

                                        TOTAL COMMAND SUMMARY
COMMAND    NUMBER       TOTAL       TOTAL      TOTAL     MEAN     MEAN     HOG           CHARS       BLOCKS
NAME         CMDS    KCOREMIN     CPU-MIN   REAL-MIN   SIZE-K  CPU-MIN  FACTOR          TRNSFD         READ

TOTALS    3783258 57853056.00    67310.63  7451756.00   859.49     0.02    0.01 253819879424      70485104

se_adb       3432 43747008.00  12407.31    63174.03 3525.91     3.62     0.20     3083666176       518872
ccom       128326  1963363.00   6062.20     8418.16  323.87     0.05     0.72     4150761472      2197137
find         2494  1491838.00   5076.13    17871.88  293.89     2.04     0.28      997972480      5693749
xemacs       1929   701229.88    607.90   100953.25 1153.52     0.32     0.01      365112832       309308
cpp        152796   632011.81   3129.21     9882.49  201.97     0.02     0.32    13075611648      5523618
c2         114341   615171.06   2906.96     4268.68  211.62     0.03     0.68     3406757632      1358243
csh        191166   516593.38   2199.30  2339170.00  234.89     0.01     0.00      824606464      3928121
```

10.9.4 Last Login

This report gives the date when a particular login was last used. It helps in finding unused login names or directories.

The following is a sample portion from a last login report.

```
Jan  5 00:14 1987  LAST LOGIN Page 1

86-04-05  raxs                 87-01-05  root                 87-01-05  ras
87-01-05  smith                87-01-05  wilson
87-01-05  johnson              87-01-05  jones
```

11 Administering Disk Quotas

The *disk quota system* controls the disk usage of individual users. After the administrator sets soft and hard quotas, the quota system monitors usage. It warns users when they exceed their *soft limit* but denies resource requests after the *hard limit* is reached. A user who remains over quota will be denied further allocation of the resources associated with the exceeded quota.

Disk space and file quotas may be set for individual users on any or all file systems, including NFS file systems.

11.1 Overview of Disk Quotas

Quotas are set for each user on a per-file-system basis. For each file system, two types of quotas are normally imposed:

Disk space quota
: A quota on the amount of disk space, measured in 1Kb blocks, that the user may occupy on the file system.

File quota
: A quota on the number of real files the user may own on the file system.

The following information is associated with each quota type:

Current usage
: The user's current usage of file system resources.

Soft limit
: The resource limit below which the user is expected to remain. This limit can be exceeded temporarily. When a user goes beyond the soft limit, a warning message is generated and a one-week timer starts.

Hard limit
: The user's absolute resource limit. If resource usage reaches this limit, one error message is generated and further resource requests fail.

Remaining warning time
: The time remaining for the user to correct soft quota violations before allocation of further resources is denied. (the time counts down from one week).

Only the first attempt to exceed the hard limit will cause an error message. This prevents programs that ignore write errors from generating a large number of error messages. The printing of error messages is reenabled when resource usage is reduced below the hard limit.

If quotas are enabled when a user logs in, and if the current usage exceeds the soft or hard quota for that user, then a warning message is displayed. The message states the number of blocks or files that must be removed and the time remaining to correct over-usage. If the remaining warning time has expired, the user is not allowed to allocate disk resources for the affected filesystem(s) until disk usage falls below the soft quota for that user and filesystem(s).

11.2 Disk Quota Command Summary

The following is a summary of quota commands:

quota	Displays a user's disk resource usage and quotas
edquota	Edits a user's disk quotas
quotacheck	Builds and updates the **quotas** files and the internal, file system quota files
quotaon	Enables the quota system
quotaoff	Disables the quota system
repquota	Displays, for specified file systems, a summary of disk resource usage and quotas

See also *quota*(1), *quota*(2), *setquota*(2), *edquota*(8), *quotacheck*(8), *quotaon*(8), and *repquota*(8).

11.3 Administering the Quota System

Perform the following steps to set up the disk quota system:

1. Decide which file systems need quotas. Usually file systems containing home directories or other user files are the only ones that require quotas, although it may also prove useful to include /**usr**. If possible, /**tmp** should be free of quotas.
2. Edit /**etc**/**fstab** to enable quotas for each file system that requires quotas. Specifically, change the file system options field (the fourth field) of /**etc**/**fstab** from **noquota** to **quota**. See *fstab*(4) for more information.
3. Disk quota information is stored in a file called **quotas** that resides in the root directory of each file system that has quotas. Create each **quotas** file with the **touch** command. For example,

```
# touch /mnt/quota
```

4. To initialize the quota system, type

```
# quotacheck -v -a
```

5. At this point, the quota system is ready for operation. Request that the system enforce quotas on the desired file systems by using the **quotaon** command. **quotaon** either enables quotas for a particular file system or, with the **–a** option, enables quotas on each file system in **/etc/fstab** designated as using quotas.

   Most sites that want to use the quota system will include the line

   ```
   /etc/quotaon -a
   ```

   in **/etc/rc.local**, so that quotas are enabled when the system is rebooted.

6. After establishing the **quotas** file and enabling quotas, use the **edquota** command to set the quota limits for each user. **edquota** enters the editor specified in your environment (default is **vi**) to edit a file for the specified user(s). This file appears in the form

   ```
   fs  /filsy  block (soft = 0, hard = 0) files (soft = 0, hard = 0)
   ```

   Terms in this line are

/filsys	Name of the mounted file system
soft	Soft limit
hard	Hard limit

   Leaving the soft and hard limits at zero for a user indicates that the user has no quotas and therefore has unlimited resource usage.

   As an example, assume you want to set quotas in the **/mnt** file system.

   a. To limit user **jsmith** to 500 blocks of disk space and 100 files and to issue a warning when those limits are in danger of being reached (at 475 blocks and 90 files, for example), enter

   ```
   edquota jsmith
   ```

   b. Using editor commands, change the line to read:

   ```
   fs /mnt blocks (soft = 475, hard = 500) files (soft = 90, hard = 100)
   ```

   c. After the changes are made, write and quit the editor.

   The **quotas** file for **jsmith** is now updated so that user **jsmith** has the specified usage limits.

After performing this procedure for all users, the quota system is established.

NOTE: Use the **–p** option to give several users the same quotas.

7. Use **quotaoff** to disable quotas. **umount** also disables quotas before a file system is dismounted, but only if the **umount** is successful. See Section 5.11, ''Dismounting File Systems with **umount**,'' for more information.
8. After each reboot, or when quotas are being established on a file system, use the **quotacheck** command to check the records within the **quotas** file for consistency with the number of blocks and files allocated to each user.

It is not necessary to dismount the file system or disable the quota system before running **quotacheck**, although on active file systems you may get inaccurate results. This does no real harm in most cases, and another run of **quotacheck** when the file system is idle will correct any inaccuracies.

Superusers can use the **quota** command to examine the resource usage and quotas of any user, and they can use the **repquota** command to check the usages and limits of all users on a file system. General users can use the **quota** command to check their individual resource usages and quota limits.

11.4 Recovering after Exceeding a Quota

In most cases, the only way to recover from over-quota conditions is to

1. Abort the activity in progress on the file system.
2. Remove enough files to bring the resource usage back below quota, and allow room for the needs of the failed program.
3. Retry the failed program.

If a write fails during an editor session because of an over-quota condition, it is likely that an initial attempt to write the file will truncate its previous contents. Furthermore, if the editor is aborted without correctly writing the file, it is likely that recent changes will be lost, possibly along with much or all of the previously existing data.

If you are caught in this situation, try one of the following safe exits:

- Use the editor's **!** shell escape command to examine resource usage and remove surplus files. Make sure the autowrite flag is not set if the editor is **vi**.
- If **csh** is being used, suspend the editor, remove some files, and then resume the editor. The **vi** editor is suspended by entering **`suspend`** or **`^Z`**, and resumed by entering **`fg`**.

- Write the file to another file system (perhaps **/tmp**) where the user's quota has not been exceeded. After rectifying the over-quota condition, move the file back to the file system on which it belongs.

11.5 How the Quota System Works

The **quotas** file contains an array of structures, with one structure for each user on the system. The structures are indexed by user ID. Data exists for each user whether or not that user has a quota on the file system. Note that the **quotas** file is not an ASCII file.

The **setquota** system call informs the system of the existence of the **quotas** file. First, **setquota** reads the quota entries for each active user, then it reads the quota entries for any open files owned by users who are not currently active. Each subsequent opening of a file on the file system is associated with the owner's quota information. In most cases this information is retained in core, either because the user who owns the file is running some process, because other open files are owned by the same user, or because some file was recently accessed.

In memory, the quota information is hashed by user ID and the file system name. This information is retained in an LRU (Least Recently Used) chain so that recently released data can be easily reclaimed. Information about users whose last processes have recently terminated is also stored in this way.

Each time a block is accessed or released and each time an inode is allocated or freed, the quota system is informed.

Measurements have shown that the quota code uses a very small percentage of the system CPU time consumed in writing a new block to disk.

12 Using the Auditing System

This section provides information about the UTX/32 security auditing facility. This facility allows administrators to audit security-related events occurring on the system, events that may jeopardize system integrity.

12.1 Types of Events

The kernel detects the occurrence of certain events and records them in an audit trail file that is specified when auditing is enabled. Audited events are divided into two classes: *required events* whose recording is required whenever auditing is turned on, and *optional events* whose recording is optional.

12.1.1 Required Events

Required events are those that either provide information necessary for the proper functioning of the audit trail analyzer or provide information about who is using the system or manipulating the system's security-related features. Required events include

- Successful logins
- Changes of current working directory
- Administrator requests for a privileged shell
- Creation and termination of processes
- Alteration of a process's user ID, logging ID, controlling TTY, or environment ID
- Changes of device ownership
- Creation of special files
- Mounting and dismounting of file systems
- System reboots
- Starting or stopping of accounting
- Changes to the events
- Starting or stopping of the audit system itself

12.1.2 Optional Events

Optional events are recorded at the discretion of the administrator. These events are those that can provide information concerning possible attempts at unauthorized use of the system. Optional events include unsuccessful login attempts, password changes, and removal of files.

The auditing system includes an option for recording all auditable events. A complete list of auditable events can be found in *setslogdetail*(8).

12.2 Starting and Stopping Auditing

The command **setslogfile** enables and disables auditing. **setslogdetail** specifies which, if any, optional events should be recorded. You must have superuser privileges to use either command.

12.2.1 Starting

If the optional **setslogdetail** command is used, it must be run before **setslogfile**. **setslogdetail** enables the recording of optional events. To specify auditing on an event, type

```
# setslogdetail   event ...
```

See *setslogdetail*(8) for a complete list of optionally auditable events.

setslogdetail can also be included in the /**etc**/**rc** file, but it must follow the **setslogfile** line.

To enable auditing, execute the command

```
# setslogfile   filename
```

This initializes the audit mechanism and begins recording, in the specified file, the messages generated by the required events.

You may enable auditing at any time, but it is recommended that you enable it before the system is brought up multi-user. This is because several important events occur at system start-up. To ensure that you do not miss important events, include the command to enable auditing in the /**etc**/**rc** file. This file is automatically executed when the system comes up multi-user. Add the following line to the /**etc**/**rc** file:

```
setslogfile filename
```

Keep your audit trail files in a directory that

- Is owned by root
- Has its file modes set so that only root can access the files

These precautions will help prevent unauthorized users from gaining access to the audit trail files.

12.2.2 Stopping

Stop the collecting of audit trail records by using the **setslogfile** command with the **–s** option:

```
# setslogfile –s
```

Although auditing can be stopped at any time, it is recommended that you wait until the system is in single-user mode; otherwise, important events could be overlooked. In particular, if auditing is stopped in multi-user mode, information about users who are logged in and the processes they own will be lost.

> NOTE: Auditing is *not* automatically stopped when the system is brought from multi- to single-user mode.

12.3 Analyzing Audit Trail Files

Use the **sanalyze** command to print a formatted version of the audit trail file. The general format of the command is

```
# sanalyze  options filename ...
```

options
: These specify which events from the audit trail file to report. You may specify events of interest by giving a user's name or a specific time frame. The absence of options causes every event in the audit trail file to be reported.

filename
: This is the name of the audit trail file to be analyzed. More than one audit trail file may be analyzed with a single invocation of **sanalyze**; however, the files must be named in chronological order starting with the oldest.

sanalyze extracts events specified by *options* from the named audit trail file(s) and formats the events into a readable report. The output is sent to the invoker's standard output, which may be redirected to another file or to the lineprinter.

You may read the audit trail file directly, but this is difficult because the information is stored in compact form to reduce the amount of disk space required for on-line storage.

For a complete description of **sanalyze** and options, see *sanalyze*(8).

12.4 Maintaining Audit Trail Files

You should devise maintenance procedures for collecting and archiving audit trail files, based on the rate at which the file grows.

Here are a few suggestions for controlling audit trail growth:

- Set aside a directory, **/usr/audit**, to be used only for the purpose of collecting and analyzing audit trail files.
- If possible, mount the audit directory on its own disk partition.
- When auditing is enabled with **setslogfile**, locate the specified log file in the audit directory. For example, include this command in the file **/etc/rc**.

```
    setslogfile /usr/audit/current
```

- Periodically transfer the **current** audit trail information to a permanent file with an appropriate name. (See the following section.)
- Periodically produce hardcopy reports from the permanent files using **sanalyze**.
- Periodically archive old audit trail files to tape. (See Section 12.4.2, "Archiving Audit Trail Files to Tape.")

The size of the security audit trail file can increase very quickly, particularly if all events are being recorded and the system is heavily used. It is therefore crucial to carefully monitor the rate at which the file expands. This rate will be different for every facility.

12.4.1 Periodic Renaming of the Current Audit Trail File

If you enable the audit mechanism from the **/etc/rc** file, you will want to rename the **current** file periodically to a name that indicates its contents.

Here is a suggested procedure. Collect each month's audit trail information in a file named **current**. At the beginning of a new month, bring the system down to single-user mode, disable auditing, and rename **current** to **audit.***month* where *month* is a three letter abbreviation for the month during which audit information was recorded (for example, **audit.feb**).

If the audit information accumulates quickly at your site, you may wish to rename the audit files more frequently than once a month. You may also need to increase the frequency of archiving the audit files to tape (see the following section). Choose time intervals that suit your system.

> NOTE: You must rename the audit trail file while in single-user mode after auditing is disabled to ensure that the information in the file is complete. Otherwise, information regarding users logged in and the processes they own may be lost.

12.4.2 Archiving Audit Trail Files to Tape

When the partition containing the audit information approaches saturation, you must archive older audit trail files onto tape and delete them from the audit directory. Save these tapes long enough to satisfy the security requirements of your site. Some sites may wish to retain these audit tapes permanently.

The frequency of archiving depends upon several factors:

- Number and frequency of optional events being recorded
- Number of users of the system
- Size of the partition used to collect the audit information

If you regularly collect a substantial amount of audit information, it might be reasonable to archive audit trail files once a month.

Site preferences determine the tape utility used to archive audit trail files. If your site has no preferences, **tar** is recommended. It is reasonably fast when a small number of files are archived, it is easy to use, and it is compatible with other UNIX systems. See *tar*(1) for more information.

When archiving audit files, write the files to tape and then delete them to conserve disk space online. Do not delete the **current** file.

For example, assume the following:

- Your monthly archive takes place on the first Monday of each month.
- Today is Monday, 4 November.
- You have already changed the audit trail file so that **audit.oct** contains the previous month's auditing information.
- You have mounted a blank tape.

A typical command sequence to use for archiving audit trail files would be

```
# cd /usr/audit
# tar c audit.oct
# tar tv
# rm audit.oct
```

These commands

- Change to the audit directory
- Create a **tar** image on tape of the audit file for October
- Verify that the **tar** images were really created by listing the file name from the tape
- Remove the online version of the file

12.4.3 Audit Trail Overflow

The audit mechanism has a built-in self-preservation scheme. When the audit mechanism detects that only 5% of the space on the partition remains, it begins sending warning messages to the console. If you receive such a message, immediately archive and remove old audit files, or move old audit files to another partition until they can be archived. These warning messages are sent each time the amount of remaining space decreases by 1%. When the amount of remaining space increases above 5%, the messages cease.

If space on the partition is not freed up in time (that is, the available space becomes 0%), then the audit trail system automatically stops. You can restart it manually once space has been freed.

13 Site-dependent System Files

This chapter describes site-dependent system files distributed with UTX/32. These files are needed for the proper functioning of various utilities and system processes. Some must be edited at installation time to reflect the configuration of each particular system and again whenever the relevant configuration information changes. Others are maintained by system utilities. The files listed in this section are distributed with reasonable defaults. All of the files can be modified unless otherwise stated. Additional information on each file may be obtained by reading the suggested manual pages and other documentation.

For the location of command source, manual page source, binaries, and site-dependent system files, see *hier*(5). For more information about NFS-related files, see the *UTX/32 Network File System (NFS) Guide*.

13.1 /etc/dffstab

The file **/etc/dffstab** describes the filesystems used by the direct file system. It is read by **dfmount** to determine the device address. The system administrator can modify this file with a text editor.

/etc/dffstab consists of a number of lines of the form:

l_dev_name:*special_file*:*device_addr*

For example

```
rtdevice:/dev/rdk0a:800
```

The *l_dev_name* field is the logical device name used to reference the volume. The *special_file* field is the full pathname of the special file associated with the device. The *device_addr* field is the physical hex address of the device (e.g., 800 or 0x800 or 0800). A # as the first non-whitespace character indicates a comment. All blank lines are ignored.

/etc/dffstab is read-only by programs; the system administrator must maintain it manually. The order of entries in **/etc/dffstab** is important because **/etc/dfmount** processes the file sequentially. The filesystem cannot be mounted at all if the path prefix contains a symbolic link.

/etc/dffstab is not distributed with the system. If direct files are to be used, then this file must be set up by the system administrator manually. See the *UTX/32 Real-Time User's Guide* for more information on direct files and direct file systems.

13.2 /etc/dumpdates

/etc/dumpdates is used by **dump** to record the date each file system is dumped and the level of the dump.

The distributed **/etc/dumpdates** file is empty. It is provided because **dump** does not automatically create the file if it does not exist. If the file is not there when **dump** is run, dump information will be lost.

It should not be necessary to edit this file manually. It is maintained by the **dump** program.

See *dump*(8), Section 15.2 ''**dump**'' and Chapter 15, ''Performing System Backups.''

13.3 /etc/exports

/etc/exports contains a list of file systems that can be exported to (remote mounted by) NFS clients. Each line of the file contains first a file system name, then a list of host names or net group names (see the file **/etc/netgroup** and *netgroup*(4)) that are allowed to remotely mount the file system. If the list is empty, the file system is exported to any host that requests it. **/etc/exports**, as distributed, is

```
/usr.POWERNODE
/tmp
/mnt
```

This file must be maintained by an administrator who edits it at system installation time according to the NFS requirements of the local network and whenever these requirements change. See *exports*(4) for more information about the entries in this file.

13.4 /etc/fstab

The file system table, **/etc/fstab**, is a list of file systems and the disk partitions on which they are usually mounted. The information in this file is used by **mount –a** (mount all file systems) and **preen**, which are generally called from **/etc/rc**. **/etc/fstab**, as distributed, is

```
/dev/dk0a / 4.3 rw,noquota 1 1
# /dev/dk0b is the default swap partition
/dev/dk0d /usr 4.3 rw,noquota 1 2
```

The distributed file serves only as an example of the information the **/etc/fstab** file should contain. This file must be maintained by an administrator who edits it at system installation time according to the current layout of the system and at any time the file system arrangement changes.

NOTE: **/etc/fstab** does not necessarily describe the current mount table. For the current mount table, see **/etc/mtab.**

See *fstab*(4) for more information about the entries in this file; see also Section 5.8, ''Setting Up **/etc/fstab.**''

13.5 /etc/ftpusers

The file **/etc/ftpusers** lists users who are *not* permitted to connect to this host via **ftp**. This file prevents possibly dangerous interlopers using generic user accounts from gaining access to the host. **/etc/ftpusers**, as distributed, contains these entries:

```
root
uucp
cshroot
```

There is another restriction on **ftp** connections: if a user has no password or has * as the first character in the password field, that user cannot connect to the host via **ftp**. (There can be an exception for the users **ftp** and **anonymous**. See *ftp*(1C), *ftpd*(8) and Section 9.3.6 ''FTP Anonymous Accounts,'' for details.)

13.6 /etc/gated.conf

/etc/gated.conf contains information for **gated**, the routing daemon that handles multiple routing protocols. It is meant as a replacement for **routed**. See *gated*(8) for more information.

13.7 /etc/gateways

/etc/gateways contains information for **routed** (the routing daemon) about gateways on detached networks or networks not running the routing daemon. See ''**/etc/gateways**'' in Section 9.3.5, ''Network Databases,'' for details.

13.8 /etc/gettytab

/etc/gettytab is used at login time to set up initial terminal characteristics and print the text of the login herald. You may wish to edit the login herald. Because the distributed **/etc/gettytab** file is large, its contents are not reproduced here.

If you have a source distribution, the source directory **/usr/src/src/etc/getty** contains the source file for **gettytab**. When a **make install** is done in the source directory, the **gettytab** file there is installed in **/etc**. For this reason, it is recommended that you either modify **/usr/src/src/etc/gettytab** and then copy it to **/etc/gettytab**, being careful that the correct permissions are preserved or that you run **make install** to install **gettytab** and the **getty** program. Binary distribution customers do not receive this file. For more information, see *gettytab*(4).

13.9 /etc/group

The group membership file, **/etc/group**, lists the groups defined on the system and the members of each group.

The group file, as distributed, is

```
zero:*:0:who,rwho,finger,w
nobody:*:-2:nobody
other:*:1:
daemon:*:2:daemon
adm:*:3:adm
bin:*:4:bin,rje
kmem:*:5:root,cshroot,daemon,adm,bin
tty:*:6:root,cshroot
operator:*:7:root,cshroot,adm
guest:*:9:network
staff:*:10:root,cshroot
uucp:*:66:root,cshroot,daemon,uucp,news
news:*:70:news,usenet
rje:*:14:rje
+::0:
```

Each line defines one group and consists of the group name, group password, group number, and a comma-separated list of user names. In general, group passwords in UNIX are obsolete and are not used.

Group **zero** lists users allowed to **su** to root (assuming they give the root password correctly). The groups **daemon**, **adm**, **kmem**, **operator**, **uucp** and **news** exist for the benefit of daemons and system utilities that run **set-group-id** to the appropriate group to gain access to files and resources they need. The group **tty** is assigned to terminal and pseudoterminal devices on which users are logged in. The **talk**, **write**, and **wall** programs run **set-group-id** to group **tty**. The group **nobody** is assigned (for local purposes) to remotely mounted NFS files owned by **root** on the remote host. It is also used as the group ID for programs that are set to read-only for publicly readable files. The group + causes the contents of the yellow pages group file to be included. See the *UTX/32 Network File System (NFS) Guide* for information on yellow pages.

Administrators are responsible for maintaining the accuracy and consistency of the group file. When your system is installed, you should edit the group file to add the appropriate users to group **zero** and to set up appropriate groups for your site. The System V command **grpck** may be used to check the consistency of the group file.

WARNING: You must not delete any of the group entries contained in the distributed group file, but you may delete a user of a group.

For more information, see *group*(4) and *grpck*(8).

13.10 /etc/hosts

The **hosts** file identifies network hosts. It lists the network address, the local host name, and any other names (aliases) by which the local host is known.

/etc/hosts, as distributed, is

```
127.1               localhost
#
# Add your systems below - do not change
# or remove localhost definition above
#
1.2                 sysname SYSNAME Sysname
```

You should modify this file, changing **1.2** to the host's internet address, changing **sysname** to the host's name, and replacing the last two names with a list of aliases for the host's name. The host's name or one of its aliases must be the same as that specified in **/etc/rc.boot**.

For more information, see *hosts*(4), *hostdb_method*(4), and Section 13.26 "**/etc/rc.boot**".

13.11 /etc/hostdb_method

/etc/hostdb_method tells the system what methods to use for host name and address lookup and in what order to use the methods. The file lists the methods, one per line. A line with a # at the beginning is a comment and is ignored. If a program needs to look up a host name or address, the listed methods will be tried in the order given.

The available choices are the DOD domain server (**nameserver**), Sun Yellow Pages (**yellowpages**), and the /etc/hosts file (**hosttables**). As distributed, only the **hosttables** method is used:

```
#nameserver
#yellowpages
hosttables
```

For more information, see *hostdb_method*(4).

> NOTE: It is probably wise to list **hosttables** as a last resort and to always keep a minimum set of information in the **hosts** file even if you are using one or both of the other name servers.

13.12 /etc/hosts.equiv

/etc/hosts.equiv contains a list of trusted hosts. When a user does an **rlogin**, **rsh**, or **rcmd** from a host listed in the **hosts.equiv** file, the local host considers the user to have the same user ID as the local user of that name and does not ask for a password. It is possible to specify network group names (see *netgroup*(4)) and to limit trusted access to particular users on a host. The distributed **/etc/hosts.equiv** file is

```
localhost
sysname
SYSNAME
Sysname
```

You should replace **localhost** with the name of your machine, and replace the rest of the names with a list of trusted hosts, if any.

Access to particular users on remote hosts may also be granted via **.rhost** files. See *hosts.equiv*(4) and *rcmd*(3X) for more information.

13.13 /inetd.conf

/etc/inetd.conf is the Internet server configuration database that relates network programs and protocols. **/etc/inetd.conf** is used by **inetd**. See *inetd*(8) for more information.

13.14 /etc/mkproto.data.pn

/etc/mkproto.data.pn is a template file that can be used with **mkproto**. **mkproto** is a command used to bootstrap a new file system. See *mkproto*(8) for more information.

13.15 /etc/modcap

This file is used by **tip** and **uucp** to set modem control characteristics. It contains entries for the most common modem devices.

The **/etc/modcap** file as distributed is quite large, so its contents are not reproduced here. The entries in this file should not need to be modified, but you may need to add entries for devices not already included. It is recommended for efficiency that you determine which modem devices are most commonly used at your site and move those entries to the beginning of the file. (All searches of this file are sequential, starting at the beginning.) For more information, see *modcap*(4).

13.16 /etc/motd

/etc/motd is an ASCII file that contains a message printed on each user's screen as part of the logging in process.

The **/etc/motd** file, as distributed, is:

```
Welcome to UTX/32 release-number
```

This file is commonly used to announce events of public interest, such as scheduled system down time.

13.17 /etc/mtab

/etc/mtab contains information about mounted file systems. Entries are added by **mount** and removed by **umount**. This file should not be edited manually. See *mount*(8) and *umount*(8) for more information.

13.18 /etc/named.boot

/etc/named.boot is the boot file for the Internet domain name server that contains information about where the name server is to get its initial data. See *named*(8) for more information.

13.19 /etc/netgroup

/etc/netgroup defines network-wide groups, used for permission checking when doing remote mounts, remote logins, and remote shells. For remote mounts, the information in **netgroup** is used to classify machines; for remote logins and remote shells, it is used to classify users. For the format of the file, see *netgroup*(4).

13.20 /etc/networks

/etc/networks contains information about the networks that make up the DARPA Internet.

The **/etc/networks** file, as distributed, is

```
loopback          127
```

For more information, see *networks*(4).

13.21 /etc/passwd

The password file **/etc/passwd** contains user identification information and encrypted passwords, as well as information needed to establish a user's environment at login. This file must be edited and maintained by an administrator.

The password file, as distributed, is

```
root::0:10:C Super User:/:
cshroot::0:10:C Super User:/:/bin/csh
nobody:*:-2:-2:NFS no privileges user:/:
daemon:*:1:2:The devil himself:/:
adm:*:3:3:Admin:/usr/adm:
bin:*:3:4:THE BIN:/:
uucp:*:66:66:Uucp Administrator:/usr/spool/uucppublic:
usenet:*:71:70::/usr/spool/news:
netnews:*:72:70:Network News Administrator:/usr/spool/news:/bin/csh
rje::162:14:RJE login:/usr/rje:/bin/csh
who::93:0:Who command:/tmp:/bin/who
rwho::94:0:Rwho command:/tmp:/usr/ucb/rwho
finger::94:0:Finger command:/tmp:/usr/ucb/finger
w::95:0:W command:/tmp:/usr/ucb/w
games:*:96:0:games/adventure owner:/tmp:
```

The colon-separated fields on each line are user name, encrypted password, user ID, group ID, ASCII text identifying the user, home directory, and program to execute on login. If no program is given, the default is **/bin/sh**. If the password field is empty, the user is logged in without giving any password (and is not prompted for one). If the first character in the password field is an asterisk, logins under the user name are not permitted at all. The fifth field (identifying ASCII text) usually contains information in the format expected by **finger** and is maintained by **chfn**.

root and **cshroot** are the superuser user names. Note that they both have user ID 0. **root** uses **/bin/sh** as the login shell and **cshroot** uses **/bin/csh**. These user names should be assigned passwords during system installation. The user **nobody** is assigned (for local purposes) to remotely mounted NFS files owned by **root** on the remote host; this is used as the user ID for programs that are set to read-only for publicly readable files. The users **daemon** and **uucp** exist for the benefit of daemons and UUCP utilities that set user ID to the appropriate user name to gain access to files and resources they need.

If a user logs in under the name **who**, **rwho**, **finger** or **w**, the corresponding command is executed instead of a shell, and the user is logged out when the command completes. As distributed, these user names have no passwords. You may want to assign passwords to these user names or remove them if you do not wish to provide these services.

> WARNING: Other than **who**, **rwho**, **finger** and **w**, do not delete any of the user entries contained in the distributed password file.

The command **vipw** should always be used when modifying the password file. This command requires that the invoker have superuser privileges and protects against multiple simultaneous editing sessions. Always be very careful when adding or deleting users. Recommended procedures for adding users are presented in Section 7.17, ''Adding Users.'' To delete a user, it is best to leave the user's entry in the password file and replace the encrypted password with an asterisk. In this way, it is easier to avoid reusing user ID numbers. It is also recommended that the password file be sorted in order of increasing user ID number.

The System V command **pwck** can be used to perform consistency checks on the password file. These checks include checking the number of fields; checking the validity of the name, and the user ID and group ID fields; and checking the existence of the home directory and login shell.

For more information, see *chfn*(1), *finger*(1), *passwd*(4), *vipw*(8), and *pwck*(8).

13.22 /etc/phones

/etc/phones contains a list of phone numbers used by the **tip** program. **/etc/phones**, as distributed, is

```
bert                 004156427750-<
ernie                004156427652
lbl                  004154864979-<
ames                 009699129
decvax               00603-884-1241
decvax               006038841230
case                 002163683923
casevax              002163683240
cwrunix              002163683240
cwruecmp             002163683906
ecmp                 002163683906
navy                 423012273700
gi                   006028997900
gi750                008015823032
texas                512-474-5511
sri                  415-326-7005
texas-20             512-477-6542
cstspr               305-797-5837
```

To add a telephone number, enter a new line containing the host name followed by the full number to dial. If you have a source distribution, the source directory **/usr/src/src/usr.bin/tip** contains the source file **phones-file** for **/etc/phones**. When a **make install** is done in the source directory, the **phones-file** file is installed in **/etc/phones**. For this reason, if you have a source distribution, it is recommended that you modify **/usr/src/src/usr.bin/phones-file** and then copy it to **/etc/phones**, being careful that the correct permissions are preserved. For more information, see *phones*(4).

13.23 /etc/printcap

The **/etc/printcap** file lists the printing devices and their capabilities and attributes.

The **/etc/printcap** file, as distributed, is

```
#NOTE - the lpf filter is undocumented in the BSD distribution.
#           It is a filter that folds backspace overprinting as produced
#           by nroff into overprinted lines, thus making underlined and
#           overprint emboldened text print much faster.  It should be
#           used only for line printers, NOT for hard copy terminals being
#           used as printers.
lp|line printer:lp=/dev/lp0:sd=/usr/spool/lpd:if=/usr/lib/lpf:rf=/usr/lib/lrf:
```

This file must be edited when new printing devices are added to the lineprinter spooling system. Each printing device requires its own entry in the file. See *printcap*(4) and Section 9.1, ''The Lineprinter System.''

13.24 /etc/protocols

/etc/protocols provides mappings between protocol names and protocol numbers.

/etc/protocols, as distributed, is

```
#
# Internet (IP) protocols
#
ip          0           IP          # internet protocol, pseudo protocol number
icmp        1           ICMP        # internet control message protocol
ggp         3           GGP         # gateway-gateway protocol
tcp         6           TCP         # transmission control protocol
pup         12          PUP         # PARC universal packet protocol
udp         17          UDP         # user datagram protocol
```

This file will not need to be edited unless new protocols are added to the kernel. For more information, see *protocols*(4).

13.25 /etc/psdatabase

/etc/psdatabase is a data file used by **ps**. It is recreated whenever **ps –U** is run and updated whenever the system is rebooted. This file should not be edited or deleted. See *ps*(1) for more information.

13.26 /etc/rc.boot

The **/etc/rc.boot** file is invoked whenever the system is booted, before the system reaches single- or multi-user mode. It is responsible for setting up important system information that need only be set once for each boot of the system. It does file system checks on and mounts the root file system. It also sets the host name and the domain for Sun Yellow Pages (if appropriate). It then runs **ps –U** to properly initialize the **ps** database. For more information, see *ps*(1).

You should modify this file, changing the line

```
    /bin/hostname noname
```

to contain the correct host name. Be warned that if you modify this file and it fails to function properly, your system will not boot correctly.

13.27 /etc/rc

The **/etc/rc** file is invoked when the system is brought up to multi-user mode from single-user mode. It is responsible for starting up required system processes and daemons. Before exiting, **/etc/rc** invokes **/etc/rc.local**.

/etc/rc, as distributed, is large and is not reproduced here. This file should rarely require modification. The only modifications should be alterations of an entry. Nothing should be deleted. Be extremely careful when modifying this file. See *rc*(8) for more information.

> WARNING: If you *do* edit this file, make sure that it still exits with status 0 when run successfully; otherwise, the system will fail to come up multi-user.

13.28 /etc/rc.local

The file **/etc/rc.local**, invoked by **/etc/rc**, contains local additions to the actions performed at boot time. Possible actions are enabling disk quotas, starting up accounting, saving a dump produced by a previous kernel, and starting networking and other daemons.

If you choose to save system core images, **dumpdirectory** should be modified to indicate the name of the directory in which to place system core dumps.

/etc/rc.local, as distributed, is large and is not reproduced here. See *rc*(8) and *savecore*(8) for additional information.

13.29 /etc/remote

/etc/remote contains the names and the attributes of the systems known to **tip**. **/etc/remote**, as distributed, is large and is not reproduced here.

If you use **tip** on your system, you will need to modify this file to define your own system. Use the entry for the host named **cstspr** as an example to guide your addition. **cstspr** is the Gould customer service Software Problem Report (SPR) database.

If you have a source distribution, the source directory **/usr/src/src/usr.bin/tip** contains the source file **remote-file** for **/etc/remote**. When a **make install** is done in the source directory, the **remote-file** file is installed in **/etc/remote**. For this reason, if you have a source distribution, it is recommended that you make modifications to **/usr/src/src/usr.bin/remote-file** and then copy the file to **/etc/remote**, being careful that the correct permissions are preserved. Binary distribution customers do not receive this file. For more information, see *remote*(4).

13.30 /etc/rmtab

/etc/rmtab contains a list of all file systems that are remotely mounted (exported). Each entry consists of a remote machine name and the name of the local file system that machine has mounted. Entries are made whenever a remote **mount** is done, and **umount** removes entries. The file is used only to preserve information between crashes and is read only by **mountd** when it starts up. This file should not be edited manually. For more information, see *rmtab*(4).

13.31 /etc/services

/etc/services is used to provide mappings between Internet port addresses and standard services provided at those addresses.

/etc/services, as distributed, is large and is not reproduced here. For more information, see *services*(4).

13.32 /etc/shells

/etc/shells lists the legal login shells. The shells listed are the only valid arguments for **chsh** or **passwd –s**. The distributed **/etc/shells** file is

```
# List of acceptable shells for chsh or passwd -s
# Ftpd will not allow users to connect who do not have one of these shells
#
/bin/sh
/bin/csh
/usr/5bin/sh
```

For more information, see *passwd*(1) and *chsh*(1).

13.33 /etc/syslog.conf

/etc/syslog.conf is the configuration file for the error-logging subsystem, **syslog**.

The **/etc/syslog.conf** file, as distributed, is

```
*.crit;kern.debug;auth.notice       /dev/console
auth.info                           /usr/adm/auth.info
mail.debug                          /usr/spool/mqueue/syslog
*.warning;kern.debug                /usr/adm/messages
*.alert                             root
*.emerg                             *
```

The contents of this file indicate who is to be notified by mail if a system error message of the given severity level or higher is written to the named log file. Unless you decide not to use the **syslog** mechanism, you will need to tailor this file to your specific needs. At the very least, either change the name **root** to the user name of one of your system administrators or add a mail alias for **root** to the

/usr/lib/aliases file. Otherwise, system error messages in the alert class will be lost. See Section 13.40, "**/usr/lib/aliases**," *syslogd*(8), and *syslog*(8) for more information.

13.34 /etc/syslog.pid

/etc/syslog.pid is the file used by **syslog** to keep track of the process ID. Do not edit this file. See *syslog*(3C) for more information.

13.35 /etc/termcap

/etc/termcap defines the attributes and characteristics of a large number of terminal types.

The **/etc/termcap** file, as distributed, is large and is not reproduced here.

This file should only rarely require modification. There are three instances in which a modification is necessary:

- If you need to improve performance, determine which terminal types are used most often by looking in the **/etc/ttys** file and asking the user community which **termcap** entries they use most often. Move the entries for the terminal types that are used most often to the beginning of the file.
- If you acquire a terminal for which there is no entry already in the file, you must create a new entry.
- If an entry proves to be incorrect for your particular needs, you may wish to modify that entry.

If you have a source distribution, the source directory **/usr/src/src/etc/termcap** contains the source file **termcap.src**. Binary distribution customers do not receive this file. When a **make install** is done in the source directory, the **termcap.src** file is rearranged in an attempt to put the most frequently used entries at the top; it is then installed as **/etc/termcap**. For this reason, if you have a source distribution, it is recommended that you make modifications to **/usr/src/src/etc/termcap.src** and then copy it to **/etc/termcap**, being careful that the correct permissions are preserved. This is also recommended that you make modifications to the file **/usr/src/src/etc/termcap/reorder**. This file contains commands usable by the **ex** editor to place frequently used entries near the top of the file. After editing these files, run **make install**:

```
# make
# make install
```

For more information, see *termcap*(4) and *ex*(1).

13.36 /etc/ttys

The **/etc/ttys** file lists all terminal devices on the system, including pseudoterminals and the console. It also includes information about the type of device on each terminal line and whether user logins and/or root logins are permitted over that terminal line.

The **/etc/ttys** file, as distributed, is

```
#
# name      getty                     type      status    comments
#
console     "/etc/getty std.9600"     vt100     on        secure"Console"
#
# Following are examples - setup as you desire
#
tty00       "/etc/getty std.9600"     vt100     off       "Regular tty"
tty01       none                      vt100     off       "Regular tty"
ttyd0       none                      vt100     off       "Dialout Line #1"
ttyd1       "/etc/getty D1200"        dialup    off       "Dialup Line #1"
tty04       "/etc/getty std.19200"    T9        off       "Regular tty"
tty05       none                      vt100     off       "Unused port"
tty06       none                      vt100     off       "Unused port"
tty07       none                      vt100     off       "Unused port"
#
# Following are  pseudo devices

#
ttyp0       none                      network
ttyp1       none                      network
ttyp2       none                      network
ttyp3       none                      network
ttyp4       none                      network
ttyp5       none                      network
ttyp6       none                      network
ttyp7       none                      network
ttyp8       none                      network
ttyp9       none                      network
```

The file, as distributed, contains entries for the console, eight terminal lines, and ten pseudoterminals. Only the console device is permitted user logins, as indicated by the word **secure** appearing as the status word in the line. To enable the other lines, replace the **offs** with **ons**. If more terminal lines are added to the system, update this file with the added device names and attributes.

> NOTE: Not marking a port secure will still allow a user whose login name appears in **/etc/groups** in group zero to become superuser.

getty must not monitor the lines used by **uucp** for originating calls. Verify that the port is set to **off** instead of **on** for outgoing lines. For example, the entry for a dialup line on tty11 should be:

```
    tty11    "/etc/getty D1200"    dialup off    "Dialup Line #1"
```

To enable logins on the receive line, **getty** must be active. Ensure that **/etc/ttys** has the following type of entry for incoming lines, for example,

```
    tty10    "/etc/getty D1200"    dialup on    "Dial in"
```

After modifying the **/etc/ttys** file, become superuser and enter the command

```
    # kill -HUP 1
```

to make the changes known to the system.

See *ttys*(4) and Sections 7.15 and 7.16, ''Adding Terminals'' and ''Adding Pseudoterminal (PTY) Devices,'' respectively.

13.37 /etc/utmp

/etc/utmp is a data file used by **who**. This file is cleared on reboots. Do not edit this file. See *who*(1) for more information.

13.38 /etc/zoneinfo/*

The files in **/etc/zoneinfo** and its subdirectories are binary data files containing local timezone and Daylight Saving Time information. This information is used by the **tzset** function, which sets the timezone for a process. **tzset** is called the first time a process calls **ctime**, **asctime**, **localtime**, or **gmtime**. The environment variable **TZ** determines which data file is used. If **TZ** is not set, the file **/etc/zoneinfo/localtime** is used. For more information on the function calls associated with **/etc/zoneinfo**, see *tzset*(3C), *ctime*(3C), *asctime*(3C), *localtime*(3C), and *gmtime*(3C).

At installation time, **/etc/zoneinfo/localtime** must be created. It should be a link to one of the distributed standard timezone data files. In the United States, the link would be to one of the files in the directory **US**, such as **US/Eastern** or **US/Central**. In Europe, the link would be to a file in **/etc/zoneinfo** such as Western European Time (**WET**), Mid-European Time (**MET**) or Eastern European Time (**EET**). The default is **US/Central**. Source sites should edit the **LOCALTIME** definition in **/usr/src/src/etc/zic/Makefile**. For more information, see Section 4.2, ''Setting Timezone and Daylight Savings Time Information.''

13.39 /sys/obj/CONFIGURATION

/sys/obj/CONFIGURATION is used by **config** to generate various configuration files which are used when reconfiguring the kernel. The file contains information about devices, swap partitions, and various tunable system parameters. See Chapter 7, ''Reconfiguring the System,'' for more information.

13.40 /usr/lib/aliases

The **/usr/lib/aliases** file contains mail aliases for the mail subsystem.

The **/usr/lib/aliases** file, as distributed, is

```
##
#  Aliases in this file will NOT be expanded in the header from
#  Mail, but WILL be visible over networks or from /bin/mail.
#
#>>>>>>>>>>The program "newaliases" must be run after
#>> NOTE >>this file is updated for any changes to
#>>>>>>>>>>show through to sendmail.
##

# Alias for mailer daemon
MAILER-DAEMON:root

# Following alias is required by the new mail protocol, RFC 822
postmaster:root

# Aliases to handle mail to msgs and news
msgs: "|/usr/ucb/msgs -s"
nobody: /dev/null
```

WARNING: The mail aliases in the distributed file *must* be there; do not delete them.

Additional aliases may be included in this file. The **newaliases** command must be executed after each edit.

See Section 9.2, ''The Mail System,'' for suggestions in setting up mail aliases and *newaliases*(1).

13.41 /usr/lib/crontab

The file **/usr/lib/crontab** is used to specify system activities which are to be performed at particular times.

/usr/lib/crontab, as distributed, is

```
45 00 * * 1-5 daemon /usr/bin/calendar -
55 07,08,10,11,12,13,14,16,17 * * 1-5 uucp sh /usr/lib/uucp/uu.hourly -local
55 06,09,15,18 * * 1-5 uucp sh /usr/lib/uucp/uu.hourly -all
00 08 * * 1-5 uucp /usr/lib/uucp/uu.daily
55 23 * * 1-5 uucp /usr/lib/uucp/uucp.daily
55 08 * * * daemon sh /usr/adm/newsyslog
00,10,20,30,40,50 * * * * root /etc/dmesg - >>/usr/adm/messages
00 00 * * 7 root mv /usr/adm/messages /usr/adm/messages.old
00,05,10,15,20,25,30,35,40,45,50,55 * * * * root /usr/lib/atrun
00 02 * * 1 root find /usr/msgs -name '[1-9]*' -mtime +14 -exec rm -f {} ;
00 * * * * root /usr/ucb/logger 'date'
00,05,10,15,20,25,30,35,40,45,50,55 * * * *root /bin/date > /datesync
```

See Section 4.3, ''**/usr/lib/crontab**,'' for more information.

13.42 /usr/lib/Mail.rc

/usr/lib/Mail.rc is the system-wide mail startup file. Users can also have a personal mail startup file, **.mailrc**, in their home directories. The distributed **/usr/lib/Mail.rc** file is

```
set append dot save ask askcc
```

You may want to modify this file depending on the preferences of the users at your site. See *mail*(1) for a description of these and other options.

13.43 /usr/5etc/magic

/usr/5etc/magic is a data file used by the System V **file** command to determine the arguments of a file. This file should not require editing unless unusual cross-system development work is necessary. See *file*(1) in the System V environment for more information.

13.44 /usr/5etc/profile or /etc/profile

/usr/5etc/profile is a data file used by the System V **profile** command. **/etc/profile** is a system link to **/usr/5etc/profile**. Users of **/usr/5bin/sh** will have this file run before the users' **.profile**. Edit this file during installation to define the timezone (**TZ**) correctly. Also edit **TZ** in **/usr/bin/sv**. See *profile*(4) in the System V environment for more information.

14 Monitoring System Performance

Monitoring UTX/32 is an integral part of ensuring proper system performance. UTX/32 provides tools for this purpose. It is the responsibility of the system administrator to be familiar enough with these tools and their output to know the standards for a given system and recognize deviations from those standards.

If there are signs of degradation in the system's performance, the system administrator should examine the system console for error messages or use **/etc/dmesg** at any terminal. Messages about a broken device or lack of space on a device should be dealt with immediately. Other error messages should be tracked down to their source. See *dmesg*(8) for more information. The system administrator should also examine the messages produced by **/etc/syslogd**; see *syslog*(8) for more information.

If system performance is poor, but console messages give no clue about the cause, a combination of the tools **uptime**, **ps**, **pstat**, **w**, **vmstat**, and **iostat** can be used. These commands give statistics on system performance. **uptime**, **ps**, and **pstat** provide *static reports*, that is, snapshots of system activity. **vmstat** and **iostat** provide *dynamic reports*; that is, given an appropriate argument, the display of information is updated periodically. The commands are discussed in the following sections.

14.1 uptime

uptime shows the current system time, how long the system has been up, the number of users currently on the system, and the average number of outstanding jobs in the run queue (the load average) for the last 1, 5, and 15 minutes. For example,

```
# uptime
 11:26am  up 1 day, 17:55,  31 users,  load average: 1.70, 1.60, 1.54
```

The load average is a good indicator of how busy the system is at a particular moment in time. A high load average (generally above 10 or 15) over several days on an optimally configured machine is a sign of inadequate machine resources.

ruptime displays similar information to **uptime** about all hosts on the local network. **w** reports who is logged onto the system and what they are doing.

See *uptime*(1), *ruptime*(1), and *w*(1) for additional information.

14.2 ps

ps gives a static picture on the state of processes on a system. **ps** can be run to find processes that are hogging system and user time. For example, typing

```
# ps laxw
```

will give you information like the following:

```
       F UID   PID  PPID  CP PRI NI ADDR   SZ RSS      WCHAN STAT TT    TIME COMMAND
04000003   0     0     0   0 -25  0   6e    0   0     runout D    ?     0:00 swapper
05008001   0     1     0   0   5  0   fe  120  64       proc I    ?     0:00 init
05000003   0     2     0   0 -24  0   ff 4096   0       proc D    ?     0:00 pagedaemon
05008000   0    55     1   5   1  0    0   80   0    selwait IW   ?     0:00 /usr/lib/lpd
05008201   0    60     1   0  15  0  166   32  16          u I    cons  0:08 /etc/update
05008201   0    63     1   0  15  0  14b   56  32          u I    ?     0:01 /etc/cron
05008001   0    88     1   0   1  0  15b  112  56    selwait I    ?     0:00 /etc/syslogd
05008001   0    92     1   2   1  0  163  144  80    selwait I    ?     0:00 /etc/inetd
05008001   0    96     1   0   1  0  199   96  64      mbutl S    ?     0:11 /etc/rwhod
05008001   0   100     1   0   1  0  1ab  112  48    selwait S    ?     0:01 /etc/routed
05008001   0   104     1   0   1  0  1b8   72  32    selwait I    ?     0:00 /etc/timed
05008001   0   110     1   2   1  0  1cf  128  72    selwait I    ?     0:00 /etc/sund
05008001   0   114     1   4   1  0  1c8   88  72    selwait I    ?     0:00 /etc/portmap
05008001   0   118     1   0   1  0  1e3   48  16      mbutl S    ?     0:08 /etc/nfsd 4
05008001   0   121   118   0   1  0  3fd   48   0      mbutl S    ?     0:09 /etc/nfsd 4
05008001   0   122   118   0   1  0  31a   48   0      mbutl S    ?     0:09 /etc/nfsd 4
05008001   0   123   118   2   1  0  1ee   48   8      mbutl S    ?     0:11 /etc/nfsd 4
05008000   0   125     1   0   1  0    0    0   0 async-bufh IW   ?     0:00 (biod)
05008000   0   126     1   0   1  0    0    0   0 async-bufh IW   ?     0:00 (biod)
05008000   0   127     1   0   1  0    0    0   0 async-bufh IW   ?     0:00 (biod)
05008000   0   128     1   0   1  0    0    0   0 async-bufh IW   ?     0:00 (biod)
05008001   0   144     1   3   1  0  370  144 120      mbutl I    ?     0:00 /usr/lib/sendmail -bd -oL9
05000000   0   178     1   7   5  0    0  120   0       proc IW   cons  0:00 make
05008001   0   179   178   0   5  0  295  112  24       proc I    cons  0:00 sh -ce for i in at basename bc
05000000   0   480     1  15   5  0    0  120   0       proc IW   cons  0:00 make
05008000   0   491   480   3   5  0    0  112   0       proc IW   cons  0:00 sh -ce /bin/make -f mtmp
05000000   0   492   491  17   5  0    0  136   0       proc IW   cons  0:00 /bin/make -f mtmp CC /bin/cc
05008000   0   494   492   4   5  0    0  112   0       proc IW   cons  0:00 sh -ce (cd 'gould'/csu
05000000   0   495   494   9   5  0    0  144   0       proc IW   cons  0:00 /bin/make CC /bin/cc
05008000   0   577   495   2   5  0    0   96   0       proc IW   cons  0:00 /bin/cc -S -O gmon.c
11008001   0   580   577   0  25  0  360  240  72            2    cons  0:00 ccom /tmp/ctm005774
05000001   0   624     1  13   5  0  2b8  120   8       proc I    cons  0:00 make
05008001   0   625   624   0   5  0  2fc  112  24       proc I    cons  0:00 sh -ce for i in adventure
05000001   0   728     1  16   5  0  2b1  120 104       proc I    cons  0:00 make
05008001   0   731   728   8   5  0  26e  112  24       proc I    cons  0:00 sh -c if test -d ../local
05000001   0   733   731   4   5  0  2f6  208 192       proc I    cons  0:07 /bin/make DESTDIR obj CC
05000001   0  2486   179   3   5  0  29a  160 144       proc I    cons  0:00 /bin/make CC /bin/cc AS as
05008001   0  2564   110  13   1  0  158  144  88    selwait I    ?     0:00 rpc.rquoted
05008201   0  2669     1   2  15  0  242  152  64          u S    cons  0:01 -csh (csh)
04008001   0  2772  2486   4   5  0  2de   96  40       proc I    cons  0:00 /bin/cc -0 -c wdslex.c
01008001   0  2940  2772 134  58  0  222  632 424            R    cons  0:45 as -o wdslex.o /tmp/ctm027723
05000001   0  2957   625  22   5  0  345  136 120       proc I    cons  0:00 /bin/make CC /bin/cc
04008001   0  2961  2957   5   5  0  3ee   96  40       proc S    cons  0:00 /bin/cc -O -o canfield canfield.c
04008001   0  2978   733   4   5  0  151  112  24       proc I    cons  0:00 sh -ce cd obj
```

To find user processes that might be using excessive amounts of time, look at the combined user and system time given under the TIME field. Also, look at the total time used by individual users and commands. This might point out users that are abusing their system privileges. You can generally ignore system server and daemon processes, signified by a ? under the TT field, that normally run for long periods of time.

If after doing a **ps** you discover that tty04 has a process named **runaway** with 57:33 of time, the next step would be to see how fast **runaway** is using up time. Issue the command

```
# ps laxwt04
```

The following would be displayed on your screen:

```
       F UID  PID PPID CP PRI NI ADDR SZ RSS WCHAN STAT TT  TIME COMMAND
5008001  13  121    1  0   3  0  1ce 88  48 73708 I    04  0:03 -csh (csh)
5008001  13 1302   56  0   3  0  2d2 96 144 73809 I    04 57:52 runaway
```

Note that **runaway** has gained another 19 seconds of system and user time since the last **ps** was issued. **runaway** is probably a cause of poor system performance. After asking the user on tty04 about the process, **runaway** should probably be killed. See *kill*(1) for more information.

There are other useful option combinations that can be used with **ps**, such as **av** and **uax**. Different options for **ps** are available in the BSD and System V environments. See the *ps*(1) manual pages in both environments for additional information.

14.3 pstat

pstat gives information about how well some system configurable parameters are set and whether your system has enough core memory. **pstat** gives a static picture on the state of a system.

Normally, **–T** is used with **pstat**. For example,

```
# /etc/pstat -T
 87/400 files
 74/275 inodes
 39/200 processes
 21/ 90 texts
 46/490 00k swap
```

pstat –T yields a two-number ratio for each field. The numerator is a static value for the actual allocation used by the system. The denominator is the maximum allocation configured for the system. For maximum efficiency, the first number should not exceed 80% of the second number. For example, if the

files ratio is over 80%, the **nfile** parameter in your **CONFIGURATION** file should be increased and a new kernel should be made. If the **inodes**, **processes**, or **texts** ratios are over 80%, the respective parameters **ninode**, **nproc**, or **ntext** should be similarly increased in your **CONFIGURATION** file, and a new kernel should be made. If your swap space ratio is over 80%, it might indicate that your swap partition is too small.

There are other useful options to **pstat**. For more information, see *pstat*(8). and Chapter 7, ''Reconfiguring the System.''

14.4 vmstat

vmstat is a program that monitors system performance dynamically. System activity is measured by job distribution, virtual-memory load, paging and swapping activity, and CPU usage. On a multi-CPU system, the CPU usage reported is the average across all CPUs. Under normal circumstances, the following conditions should exist:

- There should be few blocked jobs (column **b** under **procs**).
- Paging or swapping (the columns under **page**, mainly **po** and **sr**) should be low.
- The percent of user CPU usage (**us+sy** under **cpu**) should be high (above 60%) when the system is not idle (the column **id** under **cpu**).

The kernel updates the **vmstat procs** and **memory** column information every five seconds, so entering **vmstat 5** is recommended. For example,

```
# vmstat 5
```

might generate

```
procs    memory                       page          disk faults          cpu
r b w  avm  fre re at pi po fr de sr d0 d1 d2 d3 in  sy cs us sy id
1 0 0 1096 3792  0  0  5  0  0  0  0  0  5  0  0 26  72 25  9 22 69
0 0 0 1304 3424  0  0  0  0  0  0  0  0  0  0  0  1   9  1  0  2 98
0 0 0  744 3424  0  0  0  0  0  0  0  0  0  0  0  0   6  0  0  1 99
1 0 0 1096 3792  0  0  0  0  0  0  0  0  0  0  0  5 192  8  0  2 98
0 0 0 1096 3792  0  0  0  0  0  0  0  0  3  0  0 12 120 19  4 18 78
1 0 0 1280 3704  0  0  0  0  0  0  0  0  1  0  0  6  64 12 10 11 79
1 0 0 1168 3696  0  0  0  0  0  0  0  0  3  0  0 10  50 12 34 25 40
1 0 0 1168 3696  0  0  0  0  0  0  0  0  0  0  0 11  50 15 71 25  4
1 0 0 1168 3696  0  1  0  0  0  0  0  0  4  0  0 14  61 26 57 35  9
3 0 0 1240 3664  0  0  0  0  0  0  0  0 13  0  0 21 235 59 26 67  7
2 0 0 1696 3608  0  0  0  0  0  0  0  0 15  0  0 24 219 73 34 49 17
0 0 0 1280 3608  0  0  0  0  0  0  0  0  7  0  0 15 135 44  5 12 83
0 0 0 1280 3640  0  0  0  0  0  0  0  0  0  0  0  7  56 17  0  1 99
```

To stop **vmstat**, enter the interrupt character at your keyboard.

If many processes are blocked (column **b** under **procs**), the disk subsystem is overloaded or imbalanced. Shifting file systems among packs may help balance the load. Also, note that the file system usage, as reported by **df**, should be below 90% for optimal disk access time. See *df*(1) for information on file system usage; see *newfs*(8), *mkfs*(8), and Section 5.5.1, ''Organizing Partitions and File Systems,'' for additional information on file system organization.

If paging is continuously high (this is indicated in the **pi**, **po**, **fr**, and **sr** columns), then you may have too many users for your system, or you may simply need to add more memory. Lowering the **nbuf** tunable parameter would provide more memory for users. If **nbuf** is too low, however, disk I/O will become a bottleneck. See Chapter 7, ''Reconfiguring the System.''

If the system time (**sy/sy+us** under **cpu**) is 70% or higher, user programs may be doing excessive unbuffered I/O, or there may be excessive interrupts from broken devices. Excessive context switching, interrupt activity, or system call activity should be further investigated by the system administrator to find the cause of the problem.

There are other options to **vmstat**, such as **–c** and **–C**, that can be useful. See *vmstat*(1) for more information.

14.5 iostat

iostat is a program used to monitor terminal, disk, and cpu usage dynamically. Terminal activity is measured by counting the number of characters input (tin) and the number of characters output (tout) per second. Disk activity is measured by counting the number of bytes transferred per second (bps), the number of transfers per second (tps), and the number of milliseconds per average seek (msps). Cpu activity is measured by the amount of time spent in user mode (us), time spent running niced processes (ni), time spend in system mode (sy), and time spent in idle mode. The following is an example of how **iostat** might be used for four disks (dk0...dk3). See *iostat*(1) for more information.

```
tty
      tin   tout          bps               tps                msps              bps
14    315   32   5   21.4  123   18   17.5  194   29   19.9   22   4   15.2  25   0   56   18
18    297    0   0    0.0  147   24   18.9   90   16   20.2   42   6   15.9  44   0   43   13
15    611    0   0    0.0  193   35   20.8  141   26   18.6   21   1   28.7  49   0   34   16
19   1351   42   6   25.8  256   40    194   96   10    228   45   5   22.0  41   0   54    5
```

15 Performing System Backups

This chapter presents methods for providing your site with a system for backing up online material onto magnetic tape. Because of the possibility of losing large amounts of data through hardware failure or user error, it is highly recommended that you incorporate a systematic backup scheme.

dump and **restore** are the utilities used in most systematic backup schemes. **dump** saves files on magnetic tape, while **restore** retrieves files stored on a dump tape. **dump** and **restore** can also be used with disks rather than tapes. See *dump*(8) and *restore*(8) for more information.

tar and **cpio** can also be used for creating system backups. These utilities are used for backing up a directory, a small file system, or part of a larger file system, on a single tape. They cannot easily handle copying a large file system to multiple tapes. See *tar*(1) and *cpio*(1) for more information.

The following sections explain how to determine the size tape you will need, how to implement and schedule dumps, and how to use **restore**, **cpio**, and **tar**.

15.1 Determining Tape Requirements

Before actually copying files onto tape, it is useful to know the number and sizes of tapes required to hold the necessary data.

The **df** utility gives the disk usage for a mounted file system, and the **du** utility gives the disk usage for a directory or file. Use this information, along with the tables in this section to estimate how many tapes of a given size and density are needed to back up your file(s). The number of blocks held on a given size tape should be valid for the **dump**, **tar**, and **cpio** utilities.

For more information, see *df*(1) and *du*(1).

Tables 15-1 through 15-4 show recommended tape sizes to use for various amounts of data (in blocks). The tables provide guidelines for tapes made at different bpi (bytes per inch) densities. For example, if you want to back up a 21000-block file system at a density of 800-bpi, Table 15-1 shows that you should use a 3600-foot tape.

Table 15-1
Capacity for 800-bpi Tapes

Reel Size (inches)	Length (feet)	Blocks (1024-byte blocks)
7.0	600	4760
8.5	1200	9520
10.5	2400	19050
10.5	3600	28580

Table 15-2
Capacity for 1600-bpi Tapes

Reel Size (inches)	Length (feet)	Blocks (1024-byte blocks)
7.0	600	9520
8.5	1200	19050
10.5	2400	38110
10.5	3600	57170

Table 15-3
Capacity for 3200-bpi Tapes

Reel Size (inches)	Length (feet)	Blocks (1024-byte blocks)
7.0	600	19050
8.5	1200	38100
10.5	2400	76220
10.5	3600	114340

Table 15-4
Capacity for 6250-bpi Tapes

Reel Size (inches)	Length (feet)	Blocks (1024-byte blocks)
7.0	600	37180
8.5	1200	74410
10.5	2400	148860
10.5	3600	223320

15.2 dump

System backups are called *dumps*. When performing dumps, the contents of a working medium, such as a disk, are copied onto tape. At some sites, backup material is copied to disks instead of onto tapes. Dumps can be performed in a multi-user environment, although they should be performed during off-hours to reduce system load and preserve files in a somewhat static state. The dump tapes serve as backups of information currently on the working medium—information that could be destroyed accidentally or maliciously in the normal working environment. These tapes are the source of restores. Tapes held for long periods of time also serve as historical archives.

dump is implemented with 10 dump levels (0 through 9) that can be used to develop a system backup scheme.

Level 0	Dumps an entire file system.
Level 1	Dumps all files on a file system that have changed since the most recent level 0 dump.
Level 2	Dumps all files on a file system that have changed since the most recent level 0 or level 1 dump.
Levels 3 - 9	Each dumps all files that have changed since the most recent date of all the previous lower-level dumps.

So that information is copied and protected in a timely manner, the following dump schedule, using three dump levels, is suggested:

- Daily dumps (level 8)
- Weekly dumps (level 5)
- Monthly dumps (level 0)

This schedule provides a great deal of protection at a low cost for the amount of work performed.

dump records the date a file system is dumped and the level of that dump in **/etc/dumpdates**. See Section 13.2, "**/etc/dumpdates.**"

On an active system, it is recommended that you do the following:

- Every month, complete *full dumps* of all file systems, except temporary file systems such as **/tmp**, and save them for at least two years.
- Every week, perform *weekly dumps* that save information since the last monthly dump, and save them for at least six weeks.
- Every working day, perform incremental dumps that save files changed since the last weekly or monthly dump and save them for at least eight days.

On a less active system, a less frequent dump schedule may suffice.

The first dump taken on a file system should be a level 0 dump. After that, go through the daily/weekly/monthly cycle of dumps that you decide on for your system.

15.2.1 Maximum Dump Schedule

The overlapping tape cycles of the maximum dump schedule always retain two different dumps of any material older than one week. You can restore material even if one tape fails. This schedule has a rotation of 8 daily tapes, 6 weekly tapes, and 24 monthly tapes.

- Every working day, take a level 8 daily dump.
- Every week, on a given day, take a level 5 weekly dump.
- Every month, on a given day, take a level 0 monthly dump on a set of fresh tapes that is saved for two years. (Note that a daily dump is not done on this day.)

Table 15-5 illustrates this maximum tape rotation schedule.

Table 15-5
Maximum Tape Rotation Schedule

			Month 1			
S	M	Tu	W	Th	F	S
M1	D1	D2	D3	D4	D5,W1	
	D6	D7	D8	D1	D2,W2	
	D3	D4	D5	D6	D7,W3	
	D8	D1	D2	D3	W4,M2	
			Month 2			
S	M	Tu	W	Th	F	S
	D4	D5	D6	D7	D8,W5	
	D1	D2	D3	D4	D5,W6	
	D6	D7	D8	D1	D2,W1	
	D3	D4	D5	D6	W2,M3	

Tape **M1** is a level 0 tape of your initial configuration. **D** tapes are daily level 8 tapes. **W** tapes are weekly level 5 tapes. **M** tapes are monthly archive level 0 tapes.

15.2.2 Minimum Dump Schedule

A recommended minimum schedule for a file system has a rotation of 6 daily tapes, 4 weekly tapes, and 12 monthly tapes.

- Every other working day, take a level 9 daily dump.
- Every other week, on a given day, take a level 5 weekly dump.
- Every other month, on a given day, take a level 0 monthly dump. The monthly dump is made on a set of fresh tapes that is saved for two years. (Note that a daily dump is not done on this day.)

Table 15-6 illustrates a minimum tape rotation schedule.

Table 15-6
Minimum Tape Rotation Schedule

Month 1						
S	M	Tu	W	Th	F	S
M1	D1		D2		D3	
	D4		D5		D6,W1	
	D1		D2		D3	
	D4		D5		D6,W2	
Month 2						
S	M	Tu	W	Th	F	S
	D1		D2		D3	
	D4		D5		D6,W3	
	D1		D2		D3	
	D4		D5		W4,M2	

15.2.3 Special Considerations

On a day designated for weekly dumps (for example, Friday, the last working day in the week), do daily dumps *before* weekly dumps. The extra daily dump provides added protection in case an important weekly dump tape contains errors. If there is an extra daily dump, you can easily recreate the weekly dump by using the previous weekly dump and copying the daily dump on top of it.

On a day designated for monthly dumps (it should be the same day of the week as the weekly dumps, for example, the last Friday in the month), do weekly dumps *before* monthly dumps. You do not have to do a daily dump on the last Friday in this instance. The weekly dump provides added protection in case a monthly dump tape develops tape-read errors.

Whether you should increase the *save time*—the difference in the number of days between the last dump on tape X and the previous dump on tape X—for a given dump-level cycle depends on

- Your magnetic tape supply (a medium-sized file system can use one 2400- or 3600-foot tape per monthly dump)
- The level of system activity
- The length of the project
- The policy on archiving project data

If you have an active system, you should do dumps every day and have a large number of tapes in daily and weekly tape rotations. If you have projects that last longer than two years, you might consider having your monthly dump cycle cover more than the suggested two year period. If you wish to archive monthly tapes, then just save all monthly tapes. (Additional special-purpose archives at the end of projects can also be useful.)

The Tower of Hanoi method discussed in *dump*(8) is useful for minimizing the amount of information dumped (and time spent) daily per file system. However, the Tower of Hanoi method does not redundantly back up the weekly and monthly tapes as do the methods proposed in this section. Also, the Tower of Hanoi method may take a greater number of tapes (up to five) to restore a file, compared to a maximum of three tapes needed using the methods recommended here.

15.3 restore

A *restore* operation involves retrieving a file, a collection of files, or an entire file system stored on dump tapes and placing it onto the working medium. A restore operation is the only way to recover information that has been put on disk or tape by the **dump** command. See *restore*(8) for the required command syntax and details of use.

15.4 tar

tar is useful for easily backing up files in a directory that can fit onto one reel of magnetic tape. Its simple syntax is one of its most attractive features. See *tar*(1) for more information.

15.5 cpio

cpio performs similar to **tar**. **cpio**'s options are sometimes useful for transferring tapes between machines with different architectures. See *cpio*(1) in the System V environment for more information.

16 Troubleshooting

16.1 Device Errors

Any errors detected on peripherals or inside the kernel generate error messages to the console and to the system error log file, **/usr/adm/messages**.

/usr/adm/messages should be examined periodically. It can be automatically printed and truncated every Monday morning with the following **/usr/lib/crontab** entry:

```
5 5 * * 1 lpr -r -JERR_LOG /usr/adm/messages
```

The command can also be put into a shell script called **/usr/adm/weekly.sh**, and that shell script can be executed from **crontab**. See Section 4.3, ``**/usr/lib/crontab**,'' for more information.

16.2 What to Do if the System Crashes or Hangs

If the system gives no response to commands, it is hung or has crashed irrecoverably. If it has crashed irrecoverably, the halt panel light turns on and a //**HALT** message appears on the console. With a system hang, the run panel light will be lit, characters may be echoed at terminals, but the system gives no other response.

A system hang can occur if the operating system enters an infinite loop due to software or hardware problems or if there is a resource deadlock. In either case, an analysis of a core dump can often pinpoint the cause of the problem. For more information on obtaining and analyzing core dumps, see *crash*(8).

16.3 CPU Trap Handling

If a trap was taken by the kernel and resulted in a **die** or **firm** error, console error messages will be displayed. If the trap was taken by the kernel and resulted in a **halt** error, the system will halt, with GPRO set to the trap number.

See *trap*(8) for information on the console error message and trap handling facilities.

17 Hardware Considerations

17.1 Disk and Tape Reuse

The only way to completely erase disk packs is to hardware reformat the disk medium. This may be accomplished for DPII, UDP, or HSDP disks by using the Gould RPU Disc Processor Media Verification Program. For SCSI or MFP disks, the **prep** program can be instructed to hardware reformat the medium. Either of these programs will clear all data from the disk. See the level 2 diagnostic program, Gould RPU Disc Processor Media Verification Program (RP.MVP), in the *RPU Disc Processor Media Verification (RP.MVP.J) Disc Utility Description*, the *Gould High Speed Disc Processor Media Verification Program (DP.MFP.R) Program Description*, (not included with the UTX/32 distribution) or *prep*(8) for more information.

> NOTE: The commands, **prep**, **newfs**, and **mkfs** clear various tables set up on your disk packs, but these utilities do not actually remove data and are thus not a secure way to clean disks. See *prep*(8), *newfs*(8), and *mkfs*(8) for more information.

To remove data from your magnetic tapes before reuse, it is recommended that you either buy a bulk tape eraser or a degaussing coil. Either of these tools induces a magnetic field that effectively erases an entire tape. No other methods of tape erasure are recommended.

17.2 Configuring the System for Automatic Boot on Power-up

The auto-IPL feature enables UTX/32 to boot from the default disk drive on power-up. For the auto-IPL feature to operate correctly, the IPL I/O device must power up in an online condition and be fully operational. I/O devices that do not power up online or devices that do not correctly position the I/O media during the IPL, cannot be used for the auto-IPL feature.

I/O devices that will power up online, but that indicate inoperable status prior to achieving online status, will not work with the auto-IPL feature. However, I/O devices that indicate busy status prior to achieving the online status will work with the auto-IPL.

To enable the auto-IPL feature:

On the PN90xx

> The key switch must be in the auto position to be able to reboot from the default disk drive (normally at address 0x800).

On the PN60xx

R24 pins 1 and 5 on the MS board must be jumpered to enable UTX/32 to reboot from the default disk drive specified by the hardware (address 0x800 is not necessarily the default).

If you have a PN60xx with a key switch, the key does not need to be in the auto position unless you have a diagnostic processor.

The auto-boot feature is not intended to allow unattended system operation.

NOTE: UTX/32 is not able to recover from a power-fail trap, which is generated by the CPU after momentary outages. In this case, the system will attempt to execute a panic sequence, and if that is successful will attempt an auto-reboot. The panic sequence may cause file system damage leading **fsck** to fail during the auto-reboot and necessitating manual intervention on the system by the administrator. See *fsck*(8) for more information.

NOTE: Manual intervention is required to clear disk drives that fault as a result of a power outage.

NOTE: MICROSel does not support the auto-boot feature. For information on MICROSel, see the *Virtual Memory Diagnostic Facility User/Reference Manual* included in the *Virtual Memory V6 Documentation Set.*

Once the system reboots, if the file system damage is too severe, the startup sequence will fail.

17.3 Hardware Diagnostics

This manual does not cover hardware maintenance or diagnostic procedures. Questions about these should be referred to your area service representative. For information on the hardware diagnostics available, ask your representative about the following documents:

- *Virtual Memory V6 Documentation Set for Model 1906-690X*
- *Virtual Memory V6 Documentation Set for Model 1906-990X*

17.4 8614 Terminals (TV 9220)

When using 8614 terminals, the TERM environment variable and Handshaking Protocol Field of the Communications Screen should be set.

If the terminal is set for 8-bits and no-parity, then the login prompt will be garbled until the login name is typed and the parity setting of the terminal is determined. **getty** outputs the login prompt in 7-bits and even-parity and the terminal, being a true 8-bit terminal, tries to interpret the parity bit as a data bit.

If the smooth-scroll mode is selected, the buffer can overflow and output garbage to the screen. This may occur when **catting** a file to the terminal or scrolling through a file in **vi**. See "Display Set-up" in the *Televideo Display Terminal Operator's Manual* for directions about selecting smooth-scroll mode. Setting the TERM environment variable as follows:

```
# TERM=vt200-ss                    (Bourne shell)
```

or

```
# setenv TERM 'vt220-ss'   (C shell)
```

and setting the Handshaking Protocol Field of the Communications Screen as follows:

1. Press the Set-Up key (F3)
2. Press the Enter key until the communications screen appears
3. Use the right-arrow key to move the cursor to the Handshaking Protocol field (last field, first row)
4. Press the Enter key until **XOFF at 64** or **XOFF at 128** appears
5. Press CTRL-S to save the set-up

If the terminal is connected with a Current Loop Interface, performance may be hindered at baud rates greater than 4800.

17.5 8356/8357 Printers (NEC P6/P7)

nroff supports the NEC P6/P7 dot matrix printer in UTX/32. The file **/usr/lib/term/p6p7** defines the settings for the printer and can be invoked by using **nroff -Tp6p7**. Note that the default switch settings for the printer are 6 lines per inch (LPI) and 10 characters per inch (CPI).

An entry must be made in the **/etc/printcap** file if the NEC P6/P7 printer is to be used. See the following example.

```
lp|line printer:lp=/dev/lp0:sd=/usr/spool/lpd:if=/usr/lib/lpf:rf=/usr/lib/lrf:
dp|NEC8815:lp=/dev/tty00:sd=/usr/spool/dpd:br#9600:fc#43:fs#3280:pw#80:
nec|NECP7:lp=/dev/tty02:sd=/usr/spool/dpd:br#9600:fc#43:fs#3280:
```

Appendix A
Disk Partition Templates

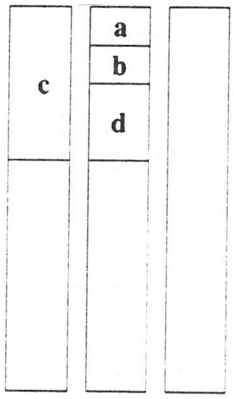
Disk Unit #________		Size ________	Model #________
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a			
b			
c			
d			
e			
f			
g			
h			

Disk Unit #________		Size ________	Model #________
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a			
b			
c			
d			
e			
f			
g			
h			

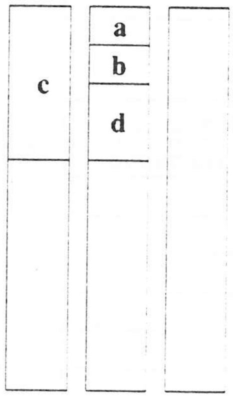
Disk Unit #________		Size ________	Model #________
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a			
b			
c			
d			
e			
f			
g			
h			

Example 1[*]

System with Two 80Mb Disks



Disk Unit # **dk0**		Size **80Mb**	Model # **9342**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	/	**0**	**18000**
b	**default swap**	**225**	**18000**
c	**all of disk**	**0**	**65520**
d	**/usr.POWERNODE**	**450**	**29520**
e			**0**
f			**0**
g			**0**
h			**0**

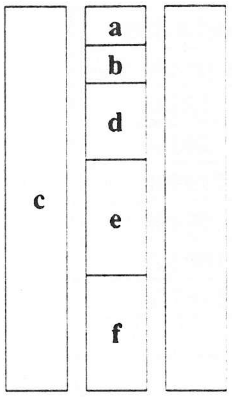


Disk Unit # **dk1**		Size **80Mb**	Model # **9342**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	**/tmp**	**0**	**18000**
b	**added swap**	**225**	**18000**
c		**0**	**65520**
d	**/mnt**	**450**	**29520**
e			**0**
f			**0**
g			**0**
h			**0**

[*] All the examples use the standard partition sizes and arrangement.

Example 2

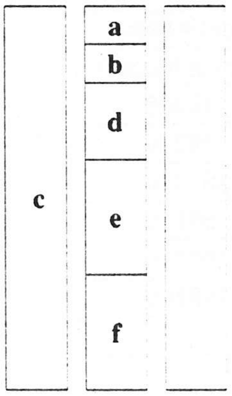
System with One 300Mb Disk



Disk Unit # **dk0**		Size **300Mb**	Model # **9346**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	/	**0**	**18240**
b	**default swap**	**59**	**20064**
c	**all of disk**	**0**	**248655**
d	**/usr.POWERNODE**	**125**	**50160**
e	**/usr.POWERNODE/src**	**289**	**100016**
f	**/mnt**	**618**	**60175**
g			
h			

Example 3

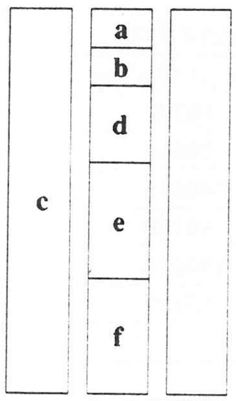
System with One 340Mb Disk



Disk Unit # **dk0**		Size **340MB**	Model # **8858**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	/	**0**	**18048**
b	**default swap**	**46**	**20352**
c	**all of disk**	**0**	**271488**
d	**/usr.POWERNODE**	**98**	**50304**
e	**/usr.POWERNODE/src**	**229**	**100224**
f	**/mnt**	**489**	**82560**
g			
h			

Example 4

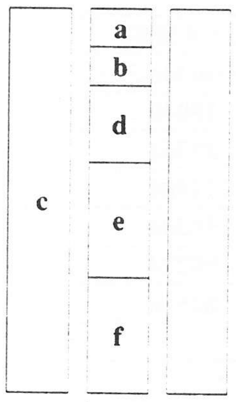
System with One CDC 9772 Disk



Disk Unit # **dk0**		Size **858Mb**	Model # **8888**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	/	**0**	**18144**
b	**default swap**	**27**	**20160**
c	**all of disk**	**0**	**715008**
d	**/usr.POWERNODE**	**57**	**50400**
e	**/usr.POWERNODE/src**	**132**	**100128**
f	**/mnt**	**281**	**526176**
g			
h			

Example 5

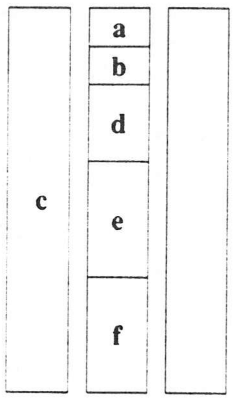
System with One Eagle 2351A Disk



Disk Unit # **dk0**		Size **474Mb**	Model # **8884**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	/	**0**	**18400**
b	**default swap**	**40**	**20240**
c	**all of disk**	**0**	**387320**
d	**/usr.POWERNODE**	**84**	**50140**
e	**/usr.POWERNODE/src**	**193**	**100280**
f	**/mnt**	**411**	**198260**
g			
h			

Example 6

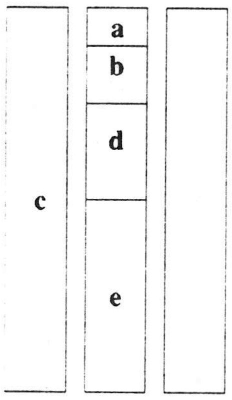
System with One Eagle 2361A Disk



Disk Unit # **dk0**		Size **689Mb**	Model # **8889**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	/	**0**	**18360**
b	**default swap**	**27**	**20400**
c	**all of disk**	**0**	**572560**
d	**/usr.POWERNODE**	**57**	**50320**
e	**/usr.POWERNODE/src**	**131**	**100640**
f	**/mnt**	**279**	**382840**
g			
h			

Example 7

System with One CDC Wren III Disk



Disk Unit # **dk0**		Size **150Mb**	Model # **8820**
Partition	Mounted on	Starting Cylinder	Size (in blocks)
a	/	**0**	**18144**
b	**default swap**	**112**	**20088**
c	**all of disk**	**0**	**156492**
d	**/usr.POWERNODE**	**236**	**50058**
e	**/mnt**	**545**	**68202**
f			
g			
h			

Appendix B
Useful Programs

This section presents a list of programs useful to a system administrator. For more information about each program, consult the various manual pages.

/bin/cat
Concatenate and print files.

/bin/chgrp
Changes the group ID of a file.

/bin/cp
Copies a file to another file or files to another directory.

/bin/date
Prints the current date and time and allows you to reset the date and time.

/bin/dd
Copies the specified input file to the specified output file with various conversion and copy options.

/bin/df
Prints the amount of free disk space.

/bin/du
Prints the number of kilobytes contained in files and directories.

/bin/grep
Searchs the input files for lines matching a pattern.

/bin/rm
Removes the entries for one or more files from a directory.

/bin/sync
Causes all information in memory which should be on disk to be written out.

/bin/wall
Writes a message to all logged-in terminals.

/etc/chown
Changes the ownership of a file.

/etc/cron
Executes commands at specified dates and times.

/etc/dump
Copies to magnetic tape all files in the file system that have been changed after a certain date.

/etc/fsck
Checks and repairs file systems.

/etc/getty
Handles terminals not yet logged in.

/etc/mklost+found
Makes a lost+found directory for the **fsck** program.

/etc/mknod
Associates a UTX/32X file name with a device or devices.

/etc/mount
Announces to the system that a removable file system is present on a named device, and lists currently mounted file systems.

/etc/rc
Executes automatically when the system changes from single-user to multi-user mode, the standard startup file.

/etc/rc.boot
Executes automatically during system boot, before the system reaches single-user mode. If the system is booted directly to multi-user mode, **/etc/rc.boot** executes before **/etc/rc**.

/etc/rc.local
Contains site-specific commands and is executed by **/etc/rc**.

/etc/restore
Restores files saved with the dump command.

/etc/umount
Announces to the system that a removable file system has been removed from a named device.

/etc/update
Executes **sync** every 30 seconds and is started by **/etc/rc**.

/etc/vipw
Allows editing of **/etc/passwd** while preventing another user from editing it simultaneously.

/usr/ucb/apropos
Prints the names of the manual pages containing instances of any of the specified keywords in their title.

/usr/man/man5/hier.5
Describes the directory hierarchy, including the location of the command sources, document sources, and binaries.

/etc/fstab
Contains information about disk partitions and file systems used by **mount**, **fsck**, and the dump program.

/etc/gettytab
Contains information about a terminal's logon initialization.

/etc/group
Contains information about different groups to which users may belong.

/etc/hosts
Identifies network hosts.

/etc/modcap
Contains information about different types of modems; the modem capability file.

/etc/motd
Contains a message that is sent to a user's terminal screen at login; the message of the day file.

/etc/mtab
Contains information about mounted file systems.

/etc/passwd
Contains information about all users that are permitted to use the system.

/etc/printcap
Lists the printing devices with their capabilities and attributes

/etc/termcap
Contains information about different types of terminals, the terminal capability file.

/etc/ttys
Contains information about the terminals connected to the system.

Appendix C
Configuration Files

The following sections contain a table of acceptable and default values for the tunable system parameters and a sample configuration file.

Section A.1	A table showing the acceptable values for the tunable system parameters, as well as the default values that are assumed if no other value is specified
Section A.2	A sample input configuration file

C.1 Accepted and Default Tunable Parameter Values

Table C-1
Accepted and Default Tunable Parameter Values

Parameter	Accepted Values	Default Value
cpu	67, 97	defaults to current machine type (info purposes only)
timezone	−720 : +720	300 (5 * 60)
dst	0, 1	1
hz	50 : 60	60
maxusers	1 : <licensed maximum>	<licensed maximum>
nproc	1 : 1024	20+8*maxusers
nbuf	1 : 1024	512
ntext	1 : 304	24+maxusers
ninode	1 : 2440	nproc+maxusers+48
nfile	1 : 3864	16*(nproc+maxusers)/10+32
ncallout	1 : 2120	16+nproc
nclist	1 : 4296	100+16*maxusers
nccbs	0 : 112	0
niccbs	0 : 1024	0
cistacksize	0 : maxinit	0
dumplo	0 : 10240	size of the first IPL volume swap partition minus the size of physical memory
maxuprc	5 : 1000	25
usrptpages	1 : 100	automatic
npty	0 : 256	32
gateway	0 : 1	0
dmmin	1 : 512	32
dmmax	128 : 2048	512
dmtext	128 : 2048	1024
swdevt-size	1 : 1024	16
msgmap	1 : 2048	100
msgmax	1 : 8192	8192
msgmnb	1 : 16384	16384
msgmni	1 : 256	50
msgssz	8 : 8	8
msgtql	1 : 128	40
msgseg	1 : 2048	1024
semmap	1 : 32	10
semmni	1 : 32	10
semmns	1 : 64	60
semmnu	1 : 64	30
semmsl	1 : 32	25

Table C-1 — *Continued*
Accepted and Default Tunable Parameter Values

Parameter	Accepted Values	Default Value
semopm	1 : 16	10
semume	1 : 16	10
semvmx	1 : 32767	32767
semaem	1 : 16384	16384
shmmax	1 : 131072	131072
shmmin	1 : 8192	1
shmmni	1 : 256	100
shmseg	1 : 16	6
shmbrk	1 : 16	4
shmall	1 : 1024	512
shmatt	1 : 512	256
svchown	1 : 0	0

C.2 Sample Configuration File

```
cpu                      67
maxusers                 32
nbuf                     256
npty                     32
gateway                  1
usrptpages               10
svchown                  1
nccbs                    5
niccbs                   10
cistacksize 384


disk     dc0 at 0x800
           unit          dk0              at 0x00
           unit          dk1              at 0x02
           unit          dk2              at 0x04
           unit          dk3              at 0x06

disk     dc1 at 0x0c00
           unit          dk4              at 0x00
           unit          dk5              at 0x02
           unit          dk6              at 0x04
           unit          dk7              at 0x06

memdisk    mc0
           unit          md0              at0x0

tape       tc0 at 0x1000
           unit          mt0              at 0x00

tape       tc1 at        0x1800           priority 10
           unit          mt1              at 0x00

hsd        hs0 at        0x4000           priority 4

ether      en0 at        0x0e00           priority 11

memory     memdisk0      type             general
           at 0x900000   length 0x300000

memory     shadow1       type             shadow
           at 0x400000   length 0x400000
           key 5         noncontiguous

region     shad1         in               shadow1
           at 0x2000     length 0x2000
           key 59        uid 4            gid 3mode 644

iop        iop0
           lpr           lc0              at 0xf8
                         unit             lp0at 0xf8
           floppy        fc0              at 0xf0
                         unit             fl0at 0xf0
                         unit             fl1at 0xf1
           asynch        as0              at 0xa0
           asynch        as1              at 0xb0 modems 0 7 dtrflow 1 2 3
           scm           sc0              at 0x80 uni
           scm           sc1              at 0xe0 uni km
           gbb           ca0              at 0xc0 quad
```

```
mfp         mfp1 at             0x7600
            asynch              as2
            lpr                 lp1

            scsi-bus1           at                 0x00
            scsi-disk           sd0                at 0x00
            scsi-tape           st0                at 0x08 fuji

            scsi-bus2           at                 0x40
            scsi-tape           st1                at 0x00

custiop     zc0 at              0x50 char length 4 subchannels 3
            unit                ez0                at 0x50
```

Appendix D
Creating a Boot Tape

This appendix describes the procedure for creating a boot tape, which is one of the tapes in the standard UTX/32 distribution. The procedure is new for UTX/32 release 2.1. The boot tape is one from which a system administrator may boot and run a specially configured UTX/32 kernel. The boot tape also loads a limited root file system into a memory disk device. While the specially configured kernel is executing, a system administrator may use the utilities in the memory disk to complete the installation of a full UTX/32 system.

D.1 Boot Tape Contents

The boot tape for UTX/32 release 2.1 or greater contains the following:

1. The tape boot program. As this program executes, it loads the remaining contents of the tape until it encounters the first double End-Of-File (EOF).
2. An EOF.
3. The tape boot kernel. This specially configured kernel executes with a memory disk as a root file system, and configures itself to support an MFP or an IOP.
4. An EOF.
5. A Common Object File Format (COFF) image of a memory disk. The COFF image's section header specifies the physical address at which the boot program will load the image. The memory disk contains a root file system suitable for use with the specially configured kernel. It contains only the utility programs necessary for UTX/32 to run properly and install a more complete UTX/32 system. This limited root file system is called the *miniroot*.
6. A double EOF.
7. An image of a complete root file system, called the *distribution root file system*, in **dump** format.
8. A double EOF.

D.2 Detailed Instructions

D.2.1 Requirements

- The procedure for creating a distribution tape must be performed by a superuser.
- The disk block size must be 1024 bytes.

- A disk partition large enough to contain a complete root file system must be available during the procedure. UTX/32 root partitions are typically 18,000 blocks in size.
- A disk partition large enough to contain the miniroot file system must be available during the procedure. To insure the creation of an optimal miniroot file system, it is strongly recommended that this partition be in a memory disk. The size of a miniroot file system for a machine containing 4 Mb of physical memory is 1472 blocks. If you are building a boot tape for a target system that contains more than 4 Mb of physical memory, you may create a larger miniroot file system containing a greater number of utility programs.

D.2.2 Building the Distribution Root File System

To create the distribution root file system

1. Halt your system and load the initial program from the UTX/32 installation tape.
2. Refer to the *UTX/32 Installation Guide* for instructions on how to install UTX/32, and install the distribution on the device which contains the distribution file system partition.
3. Reboot your system.
4. Make the distribution file system accessible:

```
# /etc/mount block_disk_device mount_directory
```

D.2.3 Configuring, Generating, and Installing the Distribution Kernel

A standard kernel, called the *distribution kernel*, must exist in the distribution root file system. To build a distribution kernel,

1. Change to the /sys/**obj** directory:

```
# cd /sys/obj
```

2. If desired, modify the **CONFIGURATION** file to configure the target system.
3. Compile the kernel using **make**:

```
# make
```

4. Copy the resulting kernel into the distribution file system:

```
# cp unix mount-directory
```

5. Create **vmunix** in the distribution file system as a symbolic link to the generated kernel:

   ```
   # ln -s mount-directory/unix mount-directory/vmunix
   ```

D.2.4 Configuring the Tape Boot Kernel

The **/sys/obj/CONF.DIST** file contains the directives necessary to create the kernel that is executed by the boot program. To create this tape boot kernel,

1. Edit the **CONF.DIST** file and tailor it to the target system. If the target system has more than 4 Mb of physical memory, the memory directive should be modified. The memory disk size may be increased as long as the base address of the memory disk is greater than or equal to the base address as it currently appears in the standard **CONF.DIST** file. Whenever possible, the base address of the memory disk should be increased. This results in an increase in the amount of free memory available for use by commands executing under the tape boot kernel.
2. Configure the kernel using **config**:

   ```
   # /etc/config < CONF.DIST
   ```

3. Compile the kernel using **make**:

   ```
   # make
   ```

D.2.5 Building the miniroot

Invoke the **build-miniroot** command to create the miniroot file system (see *build-miniroot*(8)). Because this command is implemented as a Bourne shell script, it may be modified to meet your needs. The arguments to **build-miniroot** must be

- The block device name of the partition on which the miniroot is to be built.
- The size, in disk blocks, of the target memory disk (1408 for a 4Mb target system).
- The directory on which the partition is to be mounted. This directory should be empty when **build-miniroot** is invoked.
- The pathname of the **makedev.sh** file generated by **config** during the generation of the tape boot kernel.

D.3 Modifying the Contents of the Miniroot

You may mount the miniroot file system and install the files of your choice in the miniroot, subject to the size constraints of the miniroot. The **build-miniroot** command installs only the files that are absolutely necessary for the generation and installation of a UTX/32 system. Unmount the miniroot file system when you are through modifying its contents.

D.4 Generating a COFF Image of the Miniroot File System

To generate a COFF image of the miniroot file system, invoke the **fs-to-coff** command (see *fs-to-coff*(8)). The arguments to **fs-to-coff** are

- The raw device name of the partition containing the miniroot.
- The name of the new file in which the COFF image is to be generated. **fs-to-coff** will create this file prior to generating the COFF image.

D.5 Making the Boot Tape

To create the boot tape, invoke the **make-boot-tape** command (see *make-boot-tape*(8)). The arguments to **make-boot-tape** are

- **–r**, if the last file on the tape is to be the COFF image of the miniroot. Typically, the miniroot image is followed by an image of the distribution file system, but if no such image is to be written to the tape, the **–r** option causes the tape to be rewound after the double EOF following the miniroot image is written to the tape.
- The name of the no-rewind raw tape device on which a writable tape has been mounted. The write ring must be in place on the tape.
- The pathname of the file containing the image of the tape boot kernel.
- The pathname of the file containing the COFF image of the miniroot file system.

D.6 Dumping the Distribution File System

If the **–r** option was not specified as the first argument to **make-boot-tape**, the tape will be positioned immediately following the double EOF following the miniroot image.

To dump the distribution file system, enter this command line:

```
# /etc/dump 0f raw_rewind_tape_device block_disk_device
```

Glossary

ARPANET EGP	The Exterior Gateway Protocol used on the Department of Defense ARPANET.
ACM	Asynchronous Communication Multiplexor.
bad block list	A list of blocks in a mirror set disk partition that is not readable.
backplane	The backplane is the hardware assembly that interconnects all the system boards. It includes the system bus and power lines.
boot	A routine that brings parts of the operating system from disk into memory and initializes the system.
boot-disk drive	The disk drive containing the boot volume.
boot tape	The distribution tape containing all files necessary to boot the system. For UTX/32, this includes initial boot programs, a distribution kernel and memory disk miniroot, and a copy of the UTX/32 rootfile system.
boot volume	The disk volume from which the machine was booted. (IPL).
BTP	Buffered Tape Processor.
cache	Cache memory is a dedicated intermediate memory residing between main memory and the CPU. The cache speeds processing by anticipating re-use by the CPU of the data in main memory and maintaining a copy of that data. This can reduce the number of reads of input data the system must make.
cache coherency	The ability of a multiprocessor system to keep the cache memory of each CPU consistent with the main memory.
COFF	Common Object File Format. This is a standard promoted by AT&T for an executable or object file format across different systems. It is used in System V and supported in UTX/32.
core	A generic name for all types of physical memory. Originally, a type of physical memory utilizing small magnetic rings or cores to store bits.
CPL	Gould's CONCEPT Product Line computers.
CPU	Central Processing Unit, the main processor in a computer.
datapool	A mechanism by which programs can share segments of their data region.

datapool editor	A utility that allows a user to modify datapool structures in memory.
disk pack	A removable disk volume.
disk partition	A subsection of a disk drive which may be treated as a small disk drive. There may be up to eight partitions on a UTX/32 disk volume.
disk volume	The media that contains the disk's information. A disk volume may be fixed or removable.
DTR	Data Terminal Ready.
domain	A logical group of computer names (addresses) within a computer network. Domain addresses allow computers in different domains to have the same name while still retaining a unique network address.
domain name server	A computer in a computer network domain that maintains information required for file sharing within the domain.
DOD domain server	Department of Defense domain server.
Eagle	Disk drives manufactured by Fujitsu Corporation. Eagle XP disk drives transfer data at up to 2.4Mb per second. Older Eagle drives transfer at 1.8Mb per second.
Ethernet	A computer network and interface developed by Xerox Corporation.
gate array	A semicustom integrated circuit that contains a large number of gates that can be programmed to interconnect in any manner. It reduces the number of logic circuits needed in typical designs and improves performance and reliability.
global common	Memory that can be shared or attached to the data area of one or more programs. This is useful for sharing or moving information at memory speeds.
HSD	High-Speed Device interface. A class E device interface that connects to the SelBUS. Achieves 3.2 Mb/sec transfers and is typically used for real-time applications. Custom devices can be connected to the HSD.
HSDP	High-Speed Disk Processor. A high-performance channel controller capable of handling any mixture of up to eight disk drives.
interleaving	A method of sequencing memory modules that increases the throughput by reducing the probability of encountering busy times when accessing memory.
IOP	Input/Output Processor, a part of the I/O system for Gould CPL computers. This processor controls slow-speed devices, boots the system, and runs panel mode.

IPC	Interprocess Communication, a technique by which programs communicate with each other. UTX/32 supports both BSD and SVID IPC. Typically, SVID IPC programs control message sizes, numbers of semaphore, and maximum number of shared pages as defined by the SVID. This mechanism allows separate processes to send each other messages.
IPL	Initial Program Load.
makefile, Makefile	A file containing instructions for the **make** command. Used to maintain large sets of programs or documents.
memory management	Memory management is a technique used to allow a user program to address more memory than is physically available. It also has a number of other benefits, such as datasharing and quicker program loading.
mirrored disk	A feature in the UTX/32 operating system that allows the replication of data from one disk partition onto one or more other disk partitions.
mirror set	A group of disk partitions in which all input/output to the first partition is replicated on all of the other partitions.
MFP	Multi-Function Processor. A single-board that provides the console and 7 asynchronous ports or 8 asynchronous ports, a lineprinter interface, two SCSI busses, a real-time clock, a programmable interval timer, and up to 12 external interrupts. The MFP is designed to replace the input/output processor on smaller systems.
MICROSel	A low-cost single-board processor that can be used to replace either a three-board PN6000 or PN6700 processor. When combined with an 8-slot backplane, MFP, and SCSI peripherals, the MICROSel makes for a very compact full-feature system at a low price.
mount-point name	The directory pathname where a file system is mounted.
MP or **MPB**	Multi-Purpose Bus. This bus is controlled by the IOP and is slower than the SelBUS.
NFS	Network File System. Network software facilities that allow files and file systems on one machine to be accessed by another machine.
NSFnet HELLO protocol	The routing protocol used by the National Science Foundation networks.
object code	Type of code produced when source code is compiled or run through an assembler so it can be executed on the target computer.

object file	A file containing relocatable program instructions and data. Object files result from a compiler processing a source file.
parity	A scheme for detecting errors typically used in communication devices.
paging	A technique for implementing a large virtual address space in a limited size physical memory. At any given time usually only some of the pages containing the text and data for a process are in memory, while the others are on disk. Reference to a page not in memory causes a page fault. In response to the page fault, the system reads in the required page and then continues executing the process.
panel mode	Mode in which commands are interpreted by the IOP. The panel mode prompt is //. This mode is used to perform diagnostics and error corrections. Panel mode is accessed via the console device.
pipelining	A process by which the CPU overlaps different stages of instruction fetch/execution to achieve maximum performance.
prepage	This refers to the ability of a virtual system (such as UTX/32) to preload a program in memory rather than to bringing in the pages on demand.
preprocessor	A program that transforms the input source to another format. Typically used by compilers to break down the program syntax to a format easier for the compiler to understand.
read distribution mode	The way in which read requests are distributed or shared among disk partitions in a mirror set to improve read throughput.
root directory	The topmost directory of the file hierarchy from which all basic file systems are built. It is designated by the slash (/) character.
root partition	The **a** partition of the boot volume, which contains the root file system.
RP.MVP	RPU Disk Processor Media Verification Program, a Gould diagnostic program for disk formatting.
scalar	Sequential processing of instructions/data. This typically means that a single instruction deals only with one data element.
SCCS	Source Code Control System, a program for logging and archiving changes made to a file.
SCM	Synchronous Communications Multiplexor. A piece of communications hardware that allows networking over serial lines used on Gould CPL computers.
SCSI	Small Computer System Interface. The SCSI bus is an industry standard bus capable of supporting a variety of peripherals including disks, tapes, or other hosts. The SCSI bus is packet oriented and can support multiple initiators. The MFP provides two SCSI busses.

scrubbing	The process of periodically reading memory and correcting any soft errors detected. This increases memory reliability.
SelBUS	The primary bus of the I/O subsystem. High-speed devices such as disks are connected directly off the SelBUS.
shared memory	An area of memory that can be accessed by more than one processor, allowing high-speed sharing of data.
SHM	Shared Memory, software allowing two applications to share the same segment of memory.
super block	The block following the boot block, which contains the information that describes the layout of the file system.
superuser	The name given to anyone logging in with the root password. The superuser can bypass many of the barriers the system has to prevent harmful tampering. The ability is required to properly maintain and administer the system, but is not required by system users. Normally only very few people, such as system administrators, are given the superuser password.
SVID	System V Interface Definition. An AT&T technical document describing standard interfaces that should be considered for porting software.
swap or swapping	To move core images of processes back and forth between computer memory and a swap device (a disk). Swapping is necessary because computer memory is not large enough to hold all text and data for all the running processes.
swap space	The area of the disk used for swapping. This area may be on the same disk as file system, but is not part of the file system.
symbolic link	An inode that describes a file that already existed before the link was created. Such an inode can be in any directory and contains a pathname different from another inode describing the same file. Symbolic links are interpolated during pathname expansion and allow links to files and directories which span file systems.
TCP/IP	Transport Control Protocol/Internet Protocol.
tunable system parameters	Variables that can be set by the system administrator, which affect system operation. Typically these involve sizes and threshold values.
UDP	Universal Disk Processor, a communications processor between the disk drives and the CPU.

UID	User IDentification, a unique integer assigned to a user and employed by the operating system to identify that user.
UNIX RIP	Routing Information Protocol for UNIX operating systems.
UUCP	UNIX-to-UNIX Copy.
virtual address space	The apparent address space available to a user when virtual memory is in effect. The physical memory address space may be quite smaller. See virtual memory.
virtual memory	A method of providing users with an address space larger than the physical address space.
X.25	Generic name for International CCITT protocol family.
XNS	Xerox Network Services.
XNS RIP	Routing information protocol for Xerox Network Services.
YP	Yellow Pages. A set of administrative system files shared across multiple machines. Can include a directory of usernames, passwords, and machine names on a local network that provides automatic machine name addressing. The yellow pages decrease the need for an **/etc/hosts** file.
YP server	The Sun Yellow Pages server daemon. Services Yellow Pages requests.

References

AT&T. *System V Interface Definition*. Issue 2, Volumes I-III. 1986.

AT&T. *System V Verification Suite Product Overview*. 1985.

AT&T. *System V Verification Suite Release Notes*. 1985.

AT&T. *System V Verification Suite User's Guide*. 1985.

Gould Inc. *Ethernet Controller Model 8516 Technical Manual*. 303-003370. January 1985.

Gould Inc. *Input/Output Processor (IOP) Model 8000 and 8001 IPU Console IOP Reference Manual*. 301-000170. April 1983.

Gould Inc. *Gould High-Speed Disc Processor Media Verification Program (DP.MVP.R) Program Description*. M326-06000.

Gould Inc. *Gould Multi-Function Processor Model 8002 Technical Manual*. 303-006540. June 1987.

Gould Inc. *Gould Reflective Memory Port (RM Port) Model 7302A Technical Manual*. 303-006360. May 1987.

Gould Inc. *RPU Disc Processor Media Verification (RP.MVP.J) Disc Utility Description*. M326-005320.

Gould Inc. *RPU Disc Processor Diagnostic Reference Manual*. 326-005310.

Gould Inc. *Virtual Memory V6 Documentation Set for Model 1906-690X*. 318-00020. February 1987.

Gould Inc. *Virtual Memory V6 Documentation Set for Model 1906-990X*. 318-00030. February 1987.

Index

Gould Inc., Computer Systems Division
6901 W. Sunrise Blvd.
P. O. Box 409148
Fort Lauderdale, FL 33340-9148
Telephone (305) 587-2900

GOULD
Electronics

Users Group Membership Application

USER ORGANIZATION: ____________________

REPRESENTATIVE(S): ____________________

____

____

ADDRESS: ____________________

____

TELEX NUMBER: ____________________ PHONE NUMBER: ____________________

NUMBER AND TYPE OF GOULD CSD COMPUTERS: ____________________

____

OPERATING SYSTEM AND REV. LEVEL: ____________________

____

APPLICATIONS (Please Indicate)

1. EDP
   - A. Inventory Control
   - B. Engineering & Production Data Control
   - C. Large Machine Off-Load
   - D. Remote Batch Terminal
   - E. Other

2. Communications
   - A. Telephone System Monitoring
   - B. Front End Processors
   - C. Message Switching
   - D. Other

3. Design & Drafting
   - A. Electrical
   - B. Mechanical
   - C. Architectural
   - D. Cartography
   - E. Image Processing
   - F. Other

4. Industrial Automation
   - A. Continuous Process Control Op.
   - B. Production Scheduling & Control
   - C. Process Planning
   - D. Numerical Control
   - E. Other

5. Laboratory and Computational
   - A. Seismic
   - B. Scientific Calculation
   - C. Experiment Monitoring
   - D. Mathematical Modeling
   - E. Signal Processing
   - F. Other

6. Energy Monitoring & Control
   - A. Power Generation
   - B. Power Distribution
   - C. Environmental Control
   - D. Meter Monitoring
   - E. Other

7. Simulation
   - A. Flight Simulators
   - B. Power Plant Simulators
   - C. Electronic Warfare
   - D. Other

8. Other

Please return to:

Users Group Representative

Date: ____________________

Gould Inc., Computer Systems Division Users Group. . .

The purpose of the Gould CSD Users Group is to help create better User/User and User/Gould CSD communications.

There is no fee to join the Users Group. Simply complete the Membership Application on the reverse side and mail to the Users Group Representative. You will automatically receive Users Group Newsletters, Referral Guide and other pertinent Users Group activity information.

Fold and Staple for Mailing

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 947 FT. LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESSEE

GOULD INC., COMPUTER SYSTEMS DIVISION
ATTENTION: USERS GROUP REPRESENTATIVE
6901 W. SUNRISE BLVD.
P. O. BOX 409148
FT. LAUDERDALE FL 33340-9970

Fold and Staple for Mailing

(Detach Here)

GOULD
Electronics