COHERENT

Administrator's Guide

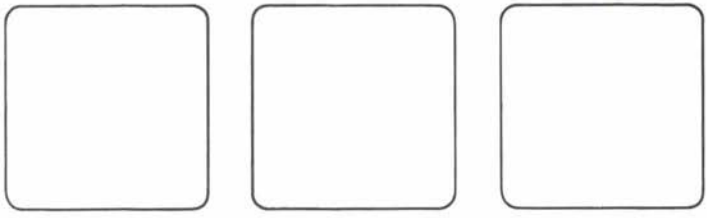

Table of Contents

1. Introduction

Congratulations on receiving the powerful COHERENT timesharing computer operating system. The COHERENT system is simple to install and easy to maintain.

The COHERENT timesharing system can be used by many people at the same time. A designated person should coordinate its use, like a key operator does for an office copier. This person is called the system administrator, and is responsible for seeing that the system runs smoothly every day. The administrator can customize the COHERENT system to the needs of an individual installation.

While your system may be used by only one person, many of the ideas discussed here will be important for making your system work at its best.

Please spend a few minutes reading this manual to familiarize yourself with the elementary concepts of COHERENT system maintenance.

The topics covered in this document are:

1) bringing up the COHERENT system,
2) running the system,
3) responding to problems which may arise, and
4) informing users of events relating to the system.

The first part of the manual, sections 1 through 8, contains instructions on how to take care of the COHERENT system and keep it working smoothly. Even if you are working with the COHERENT system for the first time, this part contains all the information you will need.

The second part, sections 9 through 16, has detailed information about the inner workings of the system. Advanced features that will help you maintain the system are discussed. Use this section to find information that will help you customize your COHERENT system.

If you would like more information on how to use the COHERENT system, see the *Introduction to the COHERENT System*. It contains a lot of useful information about how to use the system. You can follow the instructions in this manual without reading the *Introduction*, but you will have a better understanding of the topics if you read it first.

As system administrator, you will control the system by, among other things, entering information in files. The **ed** command is an easy way to do this. For information on how to use **ed**, see *ed Interactive Editor Tutorial*.

The commands available to COHERENT system users are described in the *COHERENT Command Manual*. Commands of special interest to the system administrator are labelled *Maintenance*. The *COHERENT System Manual* describes a few additional maintenance commands.

2. Booting the COHERENT system

This section discusses the procedure for installing, starting up and shutting down the COHERENT system.

You will need to install when you first receive the COHERENT system.

The startup procedure is called *booting* the system. You will need to do this after you install the COHERENT system, after the system has been shut down for any reason, or after the system unexpectedly halts.

To stop the COHERENT system, to turn off power or to reboot the system, you will need to use one of two shutdown procedures described below.

Steps for installing the COHERENT system

These installation steps are described in terms of the 8086/8088 version of the COHERENT system. Each COHERENT system is shipped with a copy of *Release Notes* specific to your type of hardware and specific to the version of COHERENT you receive. The *Release Notes* discuss the specifics of the general topics presented here.

You should receive the system on diskettes or some other medium that will enable you to load the software into a machine. The medium will depend on your supplier and the exact equipment that you have.

On every system, one terminal is the *console* terminal. You will use the console while you are starting the system, and other times when you are performing system administrator tasks. Any system error messages will be displayed on the console. Many installations use a hard copy terminal rather than a CRT as the console terminal, to provide a record of all error messages and other important events.

To install the COHERENT system on an 8086/8088 PC, insert the **build/boot** diskette into floppy drive zero, restart the computer by powering it on or by holding the **<Ctrl>** and **<Alt>** keys, then pressing the **<Del>** key. The screen will shortly say

```
PC boot
?
```

In response, type

```
. Build
```

The computer will prompt you with step-by-step instructions from that point on.

Regular startup

Once your COHERENT system has been installed, you may have need to restart it, perhaps after powering on your computer. Using the PC as an example, type **<Ctrl-Alt-Del>** with the **build/boot** diskette in floppy drive 0. To the prompt

```
PC boot
?
```

reply

```
coherent
```

in lower case letters followed by the **<RETURN>** key. If you make a mistake in typing, use the **#** key, which will erase one character each time it is struck. The system will reply with a message like

```
COHERENT (227k, 7554) Version 2.4
(c) 1982, 1983, 1984 Mark Williams Company, Chicago
#
```

Then type the command

```
check -s /dev/hd0
```

The **/dev/hd0** will be replaced by a different device name for systems other than PCs. This checks information stored in the com-

puter. You must perform this step each time you start the system, or important information could be destroyed or lost.

If no errors are detected, the system will reply

```
/dev/hd0:
/dev/hd0:
```

and again respond with **#**, waiting for your next command.

If errors are reported by **check**, you must immediately reboot *without* typing **sync**. More information on **check** is given in the section below titled "File system integrity".

Once the system has been checked without errors, type

```
<ctrl-D>
```

To type this special character, hold down the **ctrl** key, then strike the **D** key. The **ctrl** key is on the left side of the keyboard.

The system is now available to all users. The system will now type

```
Name:
```

on each terminal available for login.

The **check** command and related topics are discussed in "File system integrity" below. Also see the *COHERENT Command Manual*.

Bringing the system down

If you need to power down the system to add more hardware, you will need to bring the system down. Similarly, you will need to reboot to perform daily backups.

Normal shutdown

If your daily shutdown time is regular or predictable, you can use **cron** to notify all users two or three times before the system does go down. **cron** is discussed in the section "Tools for the administrator".

For example, if the system goes down at 5pm each day, you can have **cron** send out shutdown notices at 4:30, 4:45, and 4:55. Use **ed** or another editor to add the commands to **crontab** to do this:

```
ed /usr/lib/crontab
a
30,45,55 16 * * 1-5        /etc/wall%\
                           System going down at 5pm\
                           to save files
.
wq
```

You must be logged in as the superuser to edit this file.

If the shutdown time is not regular, you can issue a **wall** command directly to request users to log out:

```
/etc/wall
System going down at 5pm.
Please log out.
<ctrl-D>
```

Then use the **who** command to check when users have finished.

When all users have logged off, issue the

```
sync
```

command. This command will finish any pending disk I/O operations. If you do not do this, your file system could be garbled. Then, you can power off or reset the system safely.

Sudden shutdown

If circumstances require a very quick shutdown, such as a pressing need for hardware repair, use the **kill** command to remove each user (except yourself) listed by the **who** command. Use the **ps** command to determine their process numbers. The **ps** option **-a** will list all users, and the **-l** option will list the user name:

```
ps -al
```

If there is time, you should use the **wall** command to tell users so that they don't try to log in again until it is appropriate.

Once users are off, perform the **sync**, then power down or reset. Normally, the system will be left running continuously. Hardware repairs or expansion will certainly require powering down.

Superuser

A special user in the COHERENT system called the *superuser* has privileges greater than other users. The superuser can access any files and privileged programs. You will be logged in as the superuser during certain phases of your work as system administrator.

There are two ways to access the COHERENT system as the superuser. The first is to login under the user name **root**. When the system prompts

```
Coherent login:
```

or

```
Name:
```

you should reply

```
root
```

This automatically makes you superuser. To remind you that you are superuser, the COHERENT system prompts you with **#** instead of the usual **$**.

The second way to acquire the privileges of superuser is to issue the command

```
su
```

when you are logged in as a user other than **root**. You must have privileges to access **root** to do this. When you type

```
<ctrl-D>
```

in this mode, the system will return you to your previous status.

To be superuser for only one command, use the form of the command

```
su root command
```

command is the command to be executed as superuser. For example, to edit the message of the day file **/etc/motd** if you are not the superuser, type

```
su root ed /etc/motd
```

When you finish using **ed**, your original user id will be unchanged.

To limit access to privileged resources, the COHERENT system requires users to enter *passwords* before being granted that privilege. Users may be required to enter passwords before logging in.

If the **root** user has a password, you will be prompted for it. If you do not enter it correctly, the system will tell you

```
Sorry
```

and not allow you to become the superuser.

Summary

This section describes how to get the COHERENT system running, and how to stop it. The startup process is called *booting*. You will boot in two different ways, depending upon whether you are installing the COHERENT system for the first time, or whether you are simply restarting it after it has been turned off.

In shutting down the COHERENT system, be sure you know when you should use the **sync** command, and when you should not. If you are booting after **check** has detected errors, you should *not* use **sync**. If you are shutting down normally, you *must* use **sync**.

3. Day to day operation

This section discusses activities that you should perform each day to maintain your COHERENT system.

Check the system console each day for error messages or requests from users. You should also check your mailbox each day for letters from your users. If you are not familiar with **mail**, see the discussion in *Introduction to the COHERENT System*.

If there are error messages on the console, or the system has stopped running, you will need to restart it. See the following section titled "System halts".

As your system is used, more information is being stored on the disk. If large amounts are stored, all available space on the disk can be used up. You should monitor the amount of disk space yet unused with the command

```
df
```

which means **d**isk **f**ree. The **df** command reports the number of free blocks in the system. Each block contains 512 characters or *bytes* of information. If **df** replies with a number less than 1000 on a hard disk or 200 on a diskette based system, you may be running too low on disk space, and you should read the section entitled "Conserving disk space" and follow the recommendations presented there.

At the end of each day, you should back up information to protect your system from loss of valuable information. Doing so on a regular basis will allow you to retrieve lost information from the saved backup copy. The section entitled "File system backup" will tell you how to save information.

Preparing system dumps

As soon as your system has been delivered and is functional, you will need to prepare *system dumps* for saving information on a daily, weekly, and monthly basis. These dumps will be done on magnetic tape or floppy diskette.

All information in the system will be dumped each month. You should prepare at least three sets of tapes for the monthly saves,

giving you three months of full backup. You can use more sets if you wish for a period longer than three months of safekeeping. You will use the tapes in rotation, with the oldest always used next.

Information in the system that is new or has been changed since the beginning of the month will be dumped each week. You will need five tapes, since some months have five weekends in them.

Finally, you will save information that has been changed or is new since the beginning of the week every day. You will need five of these tapes, one for each working day. You will need extras in case of weekend work.

Label each tape carefully as *monthly*, *weekly*, or *daily*. Label the daily tapes Monday through Friday, the weekly tapes Week 1 through Week 5, and the monthly tapes Month 1 through Month 3. When you do the dump, write the date on the label.

Depending upon how much information is in your system, the monthly dumps may require more than one tape. In this case, you will need to prepare more than one tape for each month. Label the first "Volume 1", the second "Volume 2", and so on if more are necessary.

If you have diskettes

Some COHERENT systems use diskettes instead of magnetic tape to back up individual files or complete filesystems. These systems usually have some form of streaming tape that will back up the entire contents of a disk drive.

It is less convenient to back up files onto diskettes. A double sided double density diskette will hold at most a fraction of one million bytes. If your system has a 10 million byte hard disk which is full of information, it will require at least ten diskettes to back up all information.

To reduce diskette handling, you might back up files half as often or even less frequently.

However, to complement this, you can save important files individually on the diskettes. To prepare diskettes for this, use the **mkfs** command as described below to build empty file systems on diskettes. To create a default file system on a diskette use the command

```
/etc/mkfs /dev/fd0 720
```

To copy the files to the diskette, first **mount** the file system:

```
/etc/mount /dev/fd0 /f0
```

Here **/dev/fd0** is the name of the diskette device (for 9 sectors per track on an 8086/8088 COHERENT system) and **f0** is the directory to contain files on the diskette. Never **mount** a diskette (or any disk, for that matter) which has not been prepared with **mkfs**.

Copy files to the diskette with commands of the form

```
cp file01 /f0
```

or copy complete directories with the command

```
cpdir /usr/stuff /f0/stuff
```

Be sure to unmount the diskette with the command

```
/etc/umount /dev/fd0
```

before removing it.

Backing up information daily

This section describes how to save disk information each day. This process, called *backup*, protects information from inadvertent modification or destruction.

If you have diskettes, see the above section "If you have diskettes".

Backup will be described in terms of the 8086/8088 COHERENT system. Your release will tell you any differences that apply for your computer.

1) Log into the system as **root**. You will need the privileges of superuser to perform the dump.

2) Tell other users to log off the system by typing

```
/etc/wall
Please log off.
Time for file dump.
<ctrl-D>
```

3) Be sure that all users are logged off the system by typing the command

```
who
```

This command lists the names of all users that are still on the system.

If they have not logged off in a few minutes, send another message. Repeat the process until **who** shows no users except yourself. Note that a user may have left for the day without logging out, even though this is not a recommended practice. If backups are performed each day at the same time, users will develop the habit of logging off in time.

When all others have logged off, perform the boot process described in section 2 above. Do *not* type **<ctrl-D>**; you want to perform the dump while the COHERENT system is in single user mode.

4) If this is the last workday of the month, you will perform a *monthly* dump. Insert the first volume of the correct monthly dump diskette in the floppy drive, after adding today's date to the label, and type the command:

```
dump 0fS /dev/fd0 720 /dev/hd0
```

This will dump all files on the first hard disk partition /dev/hd0 to the 720 block diskette /dev/fd0. The details of this command may be different for your hardware.

If more floppies are needed, the computer will ask you to mount them. Be sure to label each with the volume number, with "Volume 1" for the first, "Volume 2" for the second, and so on.

5) If this is the last work day of the week, but not the last workday of the month, you will perform a *weekly* dump. Prepare the correct weekly dump diskettes, add today's date to the label, insert the first diskette, and type the command:

```
dump 6fS /dev/fd0 720 /dev/hd0
```

This will dump files that have been changed or created this month to the diskette.

6) If this is neither the last workday of the month or the last workday of the week, you will perform a *daily* dump. Prepare the daily dump diskette with today's day of the week, add today's date to the label, insert the first diskette into the drive, and type the command:

```
dump 9fS /dev/fd0 720 /dev/hd0
```

This will dump files that have been changed or created this week to the diskette.

7) You can now power down the system, or continue operation by typing **<ctrl-D>**.

Restoring information

If a user finds a file has been inadvertently destroyed or removed, for example, by typing **rm** incorrectly, you can restore the information to disk from backup tapes or diskettes.

Since backups are performed daily, you must determine the date and time that the file was last known to be good. From this date, determine on which tape the file was last correctly dumped. Find

the tape labeled with the date determined and mount it. The command

```
dumpdir
```

will list files dumped on the tape.

When the tape has been located, issue the command

```
restor x file
```

substituting the full name of the file for *file*, including all parent directories.

Conserving disk space

If disk space begins to get low, or you run out of space altogether, you need to ask users to remove obsolete or unneeded information from the system.

If space on the disk remains a problem, you might request that users place files that have not been accessed for some length of time onto backup storage. Encourage users to develop habits of removing unneeded files promptly.

Additionally, the COHERENT system provides an **ar**chive command **ar** to reduce the use of file space. To obtain information on how to use archives, issue the command

```
man ar
```

or read the **ar** section of the *COHERENT Command Manual*. **ar** condenses many files into one. **tar** will archive files onto tape or floppies. For information on **tar**, type

```
man tar
```

or see the **tar** section of the *COHERENT Command Manual*.

A later section in this document describes how to use the computer time accounting commands. If in use, they will consume disk space as work progresses on the system. If the files they use are not prop-

erly attended to, they can become quite large. If you use accounting, you should condense the accounting files often.

System halts

If your system stops running unexpectedly, you have a system *crash*. The COHERENT system has detected a problem and halted processing, or some hardware has failed.

Should this happen, examine the system console for any error messages. If there are any, carefully record them. This information will help COHERENT system experts diagnose the problem. The following section contains system error messages printed when the system halts.

If the error message is

```
Out of i-nodes
```

there are too many files in your system, even though there may be disk space available. If your system gives this message, you must delete files in order to keep running.

If the error message is

```
Out of space (m,n)
```

there is no more room for information on the disk. You will need to condense or remove information on the disk as described above.

System error messages

The following error messages may be put out by the COHERENT system. These messages indicate serious problems with your system hardware. If any of these appear, you need to contact a representative of the hardware manufacturer.

Note that the symbol # in the following messages will be replaced by a number when the message appears on the console. When reporting the problem, be sure to include the number actually printed out.

```
Arena # too small
Corrupt arena
Bad free #
Raw I/O from non user
Inode table overflow
Inode # busy
Bad block # (alloc)
Bad block # (free)
Cannot allocate stack
Cannot create process
System too large
Not a separate I/D machine
System too large
Bad segment count
Swapio bad parameter
Swapio error
Random trap
Bus error at #
Illegal instruction at #
```

If any of the following appear, you will need to **check** the indicated filesystems.

```
Out of inodes
Out of space
Bad freelist
```

Summary

As system administrator, you should check on the system each day to be sure that things are running smoothly. You should save information from the disk each day to be sure that your system runs reliably. This section tells you how.

4. Establishing a user base

Each user allowed to use your COHERENT system must have a *user name* and a *user id*; the user may also have a password. The user name is usually the user's initials or a nickname. The *user id* is an integer number used to identify the user internally to the system. As system administrator, you will assign both of these for each user. This section tells you how.

To log in to the system, a user must have an entry in the *password* file **/etc/passwd**. The password file contains each user's name, id, and password if any. As system administrator, you will maintain this file.

Similarly, each group of users is assigned a *group name*, as well as a *group id*. Groups are not necessary for the use of the COHERENT system, but some installations prefer to set up groups by project or department.

It is simple to add a new user to the system. The **newusr** command takes care of all the details, and makes an entry in the password file. You must be logged in as the superuser **root**. To create an entry for a user named **henry**, issue the command

```
/etc/newusr henry
```

When a user logs in, the shell first reads the file **.profile** from the user's home directory, if it is there. The user's home directory is normally

```
/usr/name/
```

where *name* is the user's user name.

You can give each user a **.profile** file when you establish his password. This may have only a command to set **PATH**. The user may later add more commands to **.profile** to tailor it to individual needs. A typical **.profile** is

```
PATH=:/bin:/usr/bin:..:/usr/henry/bin
stty kill ^u erase ^h int ^c
MAIL=/usr/spool/mail/henry
```

These commands control the behavior of the terminal and the command language interpreter called the *shell*. **PATH** lists the directories that contain commands, and **MAIL** gives the file name of the mail box. The COHERENT command processor examines **MAIL** after each command and reports

```
You have mail.
```

if anyone has sent you mail. The **stty** command defines special terminal keys that the user needs in communicating with the system. For a description of these commands and the COHERENT shell, see *Introduction to the COHERENT System*.

These commands are usually optional. In that case, users may not need a **.profile** file, but may make one to suit their own needs.

Summary

People who use the COHERENT system are assigned user names and user ids. The system administrator sets these up and removes them as necessary.

5. Maintaining the ttys file

The **/etc/ttys** file describes the kinds of terminals that can be attached to the COHERENT operating system. You need to customize the **ttys** file to fit your particular system when it is first installed, and to add new terminals to your system later on.

Because the COHERENT system is a flexible system, many different kinds of terminals can be attached to it. Terminals range from teletypewriters to terminals with a video screen. Since the collection of terminals varies from installation to installation, you must tell the system what configurations to expect.

Each terminal has one of several possible *speeds* of transmission. In order to communicate with each terminal, the COHERENT system must know the speed.

To keep track of the information flowing between the computer and the terminal, the system uses a terminal *name*.

All this information is contained in a COHERENT file named **/etc/ttys**.

Configuring terminals

This section outlines the steps you need to do to configure terminals.

Terminals attach to the computer by a cable plugged into a terminal *port*. Each line in the **/etc/ttys** file defines one terminal port.

First, determine what kind of terminals will be attached to your computer, and how many *ports* there are; your COHERENT system supplier will help you with this information.

Also determine the speed of each port, whether or not each port is to be in use, and the device name of each port.

Then log in to the COHERENT system as user **root** and type the command

```
ed /etc/ttys
```

and add or change information that you have written down. The following section gives details on how the lines are constructed.

ttys — File format

An example of the **ttys** file is:

```
1Gconsole
1Itty50
1Itty21
```

Although all the numbers and letters are jammed together, each line is made up of three separate parts, or fields. In the line

```
1Gconsole
```

the three fields are **1**, **G**, and **console**.

The first field tells whether or not the terminal port is currently in use. **1** means in use, **0** means not in use.

The second field describes the type of terminal port it is. Since many COHERENT systems can be called by telephone, some terminals may have a variable speed. Local terminals will have a fixed speed.

The terminal type **G** signifies 300 baud, such as that found on some printing terminals. The letter **I** signifies a terminal type having a speed of 1200 baud.

The common fixed-speed terminal types are:

type	baud rate
C	110
G	300
I	1200
L	2400
N	4800
P	9600
Q	19200

The variable speed possibilities are:

0	300, 1200, 150, 110
3	(Bell 212) 1200, 300

When a user dials into a variable speed line, a message is sent to the terminal using the first speed listed. If the result is unintelligible, the user hits the **<BREAK>** key, and the system tries the next speed. Once the speed is established, the **login** command completes the sign on process. For other types, type the command

```
man tty
```

or

```
man getty
```

To make the changes to the **/etc/ttys** file effective, issue the command

```
kill quit 1
```

as superuser.

Once a user logs into the system successfully, the **stty** command can be used to change the characteristics of the terminal. See the *COHERENT Command Manual* for details.

Summary

This section shows you how to describe the number and properties of terminals that are attached to your COHERENT system.

6. Communicating with users

As system administrator you will need to communicate with users of the system. This section discusses several ways the COHERENT system provides for you to do this. Others, including **msg**, **write**, and **mail**, are described in the *Introduction to the COHERENT System*.

wall — Broadcast message

If you need to communicate quickly with all users logged in to the system, use **wall**, or write to *all* users. The message that you send will appear on the terminal of each user. For example, to inform users that the system will become unavailable, type

```
/etc/wall
The system will be going down
at 5pm to backup files.
<ctrl-D>
```

This message will appear on user's terminals as

```
Broadcast message ......
The system will be going down
at 5pm to backup files.
```

motd — message of the day

The COHERENT system provides two convenient ways to give users news about the system. The first is called the message of the day, and is printed on each user's terminal at log in time. If the user has already seen the message, it is not shown again. To provide such a message, put information in the file **/etc/motd**.

For example, to tell the users that the system will be unavailable because new hardware is being installed in the afternoon, create the file **/etc/motd** containing:

```
The system will be unavailable in the afternoon
because new hardware is being installed.
It will go down at 2 pm.
```

When a user logs on, **login** examines **/etc/motd** and determines the date and time it was last changed. If the user has not logged in to the system since that time, the **login** process prints the information in **/etc/motd**.

news — cumulative news bulletin

The message of the day is deleted when a new message is inserted. If a user does not log in for several days, the message of the day may no longer be there. For news items that you want everyone to see, such as hours of operation or new operating procedures, you should use **news** instead of the message of the day.

The news bulletin helps users get all important messages, even if they don't log in every day. The system remembers which users have seen each news item. As a user logs in, **login** shows the parts of the **news** file that the user has not yet seen. Therefore it is easy for the user to stay up to date with the news.

To add a news item to the file, add the message to the end of the file **/etc/news**, say with **ed**. It is a good practice to put the date as the first part of each message so that users can tell when the message was put there. You need to be logged in as **root**, since the file is protected against modification by other users.

As you add news items to this file, the file will grow in size. You should purge the file of old messages every three months or so, depending upon how much you use the news file. You can remove all old news messages by typing the commands

```
>/etc/newslog
>/etc/news
```

while logged in as **root**.

The news process functions by keeping track, in the **newslog** file, of the last message seen by each user. When each user logs on, the **newslog** file is inspected to determine the last message seen. If there are messages added after that, they are printed on the user's terminal. The entry in the **newslog** file is changed to indicate that the messages have been printed.

Summary

To be on top of the system operation, you will need to stay in communication with the users of your system. This section discusses several ways that you can do so.

7. Accounting

The COHERENT system provides two types of computer time accounting to help you track the use of the system. Three commands control the accounting and provide reports at various levels of detail.

ac — Login accounting

The COHERENT system keeps a record of each time each user logs in to the system. This record contains the user name, the terminal number, the date and the time of the login. The time of logging out is also kept.

This information is used to summarize the length of time that each user or all users were logged into the system. If you say

```
ac
```

the total of all login times recorded in the file will be printed. An example of the result is

```
Total:    8357:00
```

You can ask for a summary of total login times for each day by typing

```
ac -d
```

An example result would be

```
Friday November 13:
        Total:    53:08
Saturday November 14:
        Total:    75:36
Sunday November 15:
        Total:    73:15
```

Finally, you can summarize the times for individual users with the command

```
ac -p jack ted frank
```

This will show the total times for **jack**, **ted**, and **frank**.

```
frank           1100:42
jack            910:41
ted             641:58
Total:          2653:21
```

Also,

```
ac -pd
```

will give the time for each user for each day that they were logged in.

Login accounting is not automatically operational. The login information will be collected only if the file **/usr/adm/wtmp** exists.

To start login accounting if it is not working, type the command

```
>/usr/adm/wtmp
```

while logged in as **root**. This creates the file **/usr/adm/wtmp** if it does not exist (and destroys existing information if it does) and thereby enables login accounting. You must be logged in as **root** to modify **/usr/adm/wtmp**.

To turn off login accounting while it is running, you can type

```
rm /usr/adm/wtmp
```

When login accounting is activated, it is a good idea to purge the file **/usr/adm/wtmp** periodically as it will grow continuously and on an active system can eventually consume a lot of disk space. To purge the current information but leave accounting turned on, type

```
>/usr/adm/wtmp
```

sa — Process accounting

While login accounting tells you how much time is spent logged in to the system, it does not tell you the individual commands used. *Process accounting* will do so. Each execution of each command is a separate process. Process accounting will keep track of system time, user time, and real time for each command executed by each user on the system. The **sa** command will report this information in many formats for you.

There are several options to the **sa** command to control the different reports. With no options, **sa** produces a listing for each command of the number of calls made, the total CPU time, and the total real time used by the command, ordered by decreasing CPU time. This is a summary by command. For example, the command

```
sa
```

will produce a report similar to

```
          #CALL   CPU     REAL
sh        61      1       832
ld        5       1       7
ar        5       0       1
ranlib    3       0       1
p         16      0       11
dld       2       0       1
lc        19      0       1
cc        4       0       8
atrun     43      0       1
find      1       0       0
ed        1       0       2
cat       4       0       1
rm        3       0       0
j         1       0       0
spin      2       0       1
grep      2       0       0
msg       4       0       0
ps        1       0       0
pr        2       0       0
watch     4       0       0
who       2       0       0
stty      3       0       0
chown     1       0       0
sort      1       0       0
mv        2       0       0
pwd       1       0       0
nm        1       0       0
df        1       0       0
ls        1       0       0
echo      3       0       0
accton    1       0       0
```

The actual numbers depend on what commands are used in your system, and the characteristics of your hardware. To summarize by user, use the **-m** option:

```
sa -m
```

The option **-l** will separate user CPU time and system CPU time:

```
sa -l
```

will produce

```
          #CALL   USER    SYS     REAL
sh        61      0       1       832
ld        5       0       0       7
ar        5       0       0       1
ranlib    3       0       0       1
p         16      0       0       11
dld       2       0       0       1
lc        19      0       0       1
cc        4       0       0       8
atrun     43      0       0       1
find      1       0       0       0
ed        1       0       0       2
cat       4       0       0       1
rm        3       0       0       0
j         1       0       0       0
spin      2       0       0       1
grep      2       0       0       0
msg       4       0       0       0
ps        1       0       0       0
pr        2       0       0       0
watch     4       0       0       0
who       2       0       0       0
stty      3       0       0       0
chown     1       0       0       0
sort      1       0       0       0
mv        2       0       0       0
pwd       1       0       0       0
nm        1       0       0       0
df        1       0       0       0
ls        1       0       0       0
echo      3       0       0       0
accton    1       0       0       0
```

To list the user name and the command name, use **sa** with the option **-u**. No times or counts are given. The command

```
sa -u
```

will produce output of the form:

```
tj              p
tj              lc
tj              find
tj              pr
bin             lc
tj              spin
tj              sh
bin             cc
bin             cat
bin             ld
bin             dld
farl            who
farl            sh
```

This report has been truncated and edited to save space. In practice, it is longer.

The **-u** option overrides other options.

Process accounting is on only if you specifically request it to be on. Because it consumes disk space as it operates, and adds a slight amount of time to each command, it is normally left off.

To turn on process accounting, type the command

```
/etc/accton /usr/adm/acct
```

while logged in as **root**. The file **/usr/adm/acct** holds the raw accounting information.

Process accounting is turned off by using the same command with no file name, as in:

```
/etc/accton
```

If accounting is not on when you type this command, you will get an error message. No information is gathered when accounting is turned off.

When process accounting is in use, the file containing the accounting information grows with each user command issued. It is wise to regularly condense or remove the information. The information is condensed by **sa** when given the **-s** option. Accounting must be turned off while condensing information.

To condense information by command after producing the default report:

```
sa -s
```

The information summarized by user will appear in **/usr/adm/usracct**, and information saved by command is placed in **/usr/adm/savacct**. These summarized files are used in future requests to **sa**. After condensing, you can turn accounting back on.

Additional options give flexibility to the report. Type

```
man sa
```

or examine the **sa** section of the *COHERENT Command Manual* for additional details on the options.

Summary

To help keep track of your resources and how much of the computer is used for what, the COHERENT system provides two different kinds of accounting. The information can be reported in various formats for your convenience.

8. cron — Event scheduling

A valuable tool for you in your role as system administrator is the command **cron**. With it, you can schedule announcements and other commands to be executed, even in your absence.

To specify a command to be executed at some later time, simply enter one line of information in the **/usr/lib/crontab** file. You must be logged in as **root** to modify this file.

For example, assume that you want to greet all users logged into the system on Monday morning. You can do this by sending them a message at 8:13 on Monday. Add the following lines to **/usr/lib/crontab** with **ed**:

```
13 8 * * 1      /etc/wall%Am Monday!
```

The numbers and * at the beginning specify the time:

```
13 8 * * 1
```

The **13** means "13 minutes past the hour". The **8** means "8 a.m.". The positions containing * normally specify the day and month. The use of the two * characters mean "any day" and "any month". Finally, the **1** means "day 1 of the week", which is Monday. Days of the week are numbered 0 (for Sunday) through 6 (for Saturday).

```
minute          13
hour            8
day of month    * -- all days
month           * -- all months
day of week     1 -- Monday
```

Since each entry in the **crontab** file must be on one line, the % symbol represents where you want a **<RETURN>** to appear. If the information is too long for one line, enter a backslash character before the **<RETURN>** at the end of the line. The backslash will cause **cron** to ignore the **<RETURN>**.

With this information in the file, **cron** causes the command

```
/etc/wall
Am Monday!
```

to be executed at 8:13 every Monday morning.

cron expects time to be in the 24-hour clock, so that 1 pm would be represented as **13** hours. If you need the % to appear as itself, precede it with a backslash:

```
\%
```

The times for **cron** commands can be even more complex than the numbers and * shown above.

You can express a range for any of the five parts of a time by separating two numbers with a hyphen. To send everyone a lunch break message on week days:

```
59 11 * * 1-5     /etc/wall%Lunch!!
```

To list choices of times, separate single numbers or ranges with commas but no spaces. To call a meeting on Monday, Wednesday, and Friday at 3 PM, use

```
0 15 * * 1,3,5    /etc/wall%meeting..
```

The time specification

```
0 15 * * 1,3,5
```

represents the time 1500 (3 p.m.) on every Monday, Wednesday, and Friday.

A recommended use of **cron** is to remind the users that it is time for a filesystem backup. With the flexibility of the date and time specification available with **cron**, you can encode your backup plan into the times. This would include the different level of backup on different days, weeks and months. As the time for backup approaches, a warning message would be sent to all users. An

additional message would be sent to the administrator to prepare to perform the backup.

wall is just one example of commands that can be used with **cron**, many others can be used. If you want to do periodic accounting reports, you can put a command like

```
sa -s | lpr
```

to print the accounting information.

To start up the **cron** program, you need the following line in **/etc/rc**:

```
/etc/cron &
```

Summary

As system administrator, you are probably a busy person. **cron** can help you out by reminding you or users of upcoming events throughout the day. You can also use **cron** to automatically perform certain tasks daily.

9. File system backup

This section discusses the principles behind saving and restoring files. An earlier section detailed a standard backup procedure. If you wish to tailor your own backup procedure, information in this section can help.

For the best understanding of this section, you should be familiar with COHERENT files. The *Introduction to the COHERENT System* covers the basics of files.

Two programs, **dump** and **restor**, are the tools you will use to make your disk-based information secure from hardware failure or inadvertent user destruction.

The command **dump** will dump selected file systems to a diskette or magnetic tape or other backup media. **restor** will copy files from the external media and place them on an existing file system. You can restore all files, or just one or two. The regular use of these two commands will provide you with a secure backup copy of your files.

This section first discusses the concepts behind dumping and restoring files. Then, the specific uses of the commands are detailed.

Backup procedures

There are several reasons to backup COHERENT files and file systems.

The most common reason is to protect your valuable programs and information against accidental destruction.

Performing file backups on a regular basis can greatly reduce recovery costs should files be accidentally destroyed.

You may need to transfer files from one COHERENT configuration to another. Or, you may install larger capacity disk hardware. Using **dump** followed by **restor** is a convenient way to do these tasks.

Strategies

In order to minimize daily backup overhead, it is important to use a backup strategy suited to your environment and computer usage. The strategy presented above will work for most installations of

your COHERENT system, but with experience you may want to tailor the procedure for your installation.

Whatever your strategy, stick to it rigorously. Then your users will know that files will be protected with a constant level of reliability.

Some hardware implementations of the system have "streaming" tape drives which will save the contents of a complete disk in a matter of minutes. Other systems have conventional tape drives or diskettes. For the latter systems, saving the entire disk takes too long to be done daily. Saving only recently changed files will make daily backup more convenient.

The **dump** program provides the ability to perform **incremental** dumps. Only files that have changed since a *dump date* will be saved. Files that have not changed will not be dumped.

To keep track of dump dates, the **dump** command maintains a file of dump dates. The dump dates are kept for each dump level.

Dump levels

The **dump** program provides ten *levels* of incremental dump numbered 0 through 9. By definition, the level zero dump contains all files. All other levels define the dump date as the date that the next highest level was dumped. Only files changed or created *after* the dump date will be dumped.

To give an example of the levels and dates, assume that a level zero dump was taken on January 1, a level 3 dump on January 3, and a level 6 dump was taken on January 10:

```
level    date
 0       January 1
 3       January 3
 6       January 10
```

Assume that you do a level 9 dump. The next highest level previously dumped is 6. The level 6 dump was done on January 10, so this is the dump date. Thus, a level 9 dump will dump all files changed or created after January 10.

The dump dates actually include the time of day as well, but for the purposes of illustration that has been omitted.

A level 4 dump will save all files changed or created after the date of the level 3 dump.

Using this level feature can help you design your overall backup strategy. An alternative to the backup strategy presented above includes a quarterly dump:

```
level 0         quarterly
level 3         monthly
level 6         weekly
level 9         daily
```

If you implement this strategy, the level 0 tapes or floppies can be kept for as long as is reasonable, perhaps one year. A level 0 dump (and possibly other levels) may require more than one reel or diskette, depending upon your hardware and how many files there are and how often they are changed. Thus, you will need four sets of level 0 tapes or floppies to keep a year-long cycle. With this scheme, take a level 0 dump at the end of each calendar quarter.

At the end of each month that does not end a calendar quarter, perform a level 3 dump. Files changed since the quarterly dump was taken will be dumped. You will need only two sets of level 3 tapes.

Level 6 dumps should be taken at the end of each week. Level 9 dumps should be taken each day except for the day that the level 6, 3 or 0 dumps are taken.

As noted earlier, you need to design your dump strategy to suit your installation's individual needs. The above strategy is an example; if suitable, you can use it directly. If not, use this scheme as a starting point to design your own.

dumpdate — Dump dates

The **dump** command keeps a list of dates for each level on each device so that it can determine what the dump dates for any level are.

You can list these on your terminal with the command

```
dumpdate
```

A typical reply is

```
Level 0 Wed Feb 10 21:13:36 1982 hd0
Level 0 Wed Feb 10 21:47:39 1982 hd0
Level 9 Thu Feb 25 23:01:59 1982 hd0
Level 9 Thu Feb 25 23:05:51 1982 hd0
```

Here **hd0** is the name of the disk device, and will be different if you have different disks or a system other than the 8086/8088 PC.

restor — Restoring files

Now that your files are being saved regularly, what do you do if a file does get inadvertently destroyed? For example, a user accidentally removes or changes a file and needs to get back the original. You can restore the file individually from the dump tapes or floppies with the command **restor**.

To restore the file or files, you need to know which set of dump tapes the file is on. To start, determine the latest date that the file was known to be correct. Then, locate the latest set of tapes that was dumped before that date; this will be expedited by writing the date and time each tape is dumped on the tape itself. To verify that the file is on a given tape, you can check by mounting the tape and issuing the command

```
dumpdir
```

If an entire file system needs to be restored, you can start with an empty disk. To clear a disk, see the section on **mkfs**.

To completely restore a file system from various levels of tape, begin with the level 0 set of tapes. Then restore the next higher level, and so on, until you have restored the highest level of dump, probably level 9.

The **restor** command will do a full or partial restore of files previously saved by **dump**.

To restore an individual file from the default restore device, use the command

```
restor x file01 file02
```

which will restore files **file01** and **file02** from the dump tape. Your system supplier will tell you what your default restore device is. Be sure to include the complete pathname of the file.

To restore a complete tape, use

```
restor r
```

This will remove any files on disk that are also on tape before restoring them. All files saved on the tape will be restored to the disk file system.

There must be enough space in the file system to hold the files from the tape, whether you are restoring a single file or an entire file system.

dumpdir — List dump directory

The **dumpdir** command can help you analyze the contents of a dump diskette.

```
dumpdir f /dev/fd0
```

will list all file names on a dump diskette on the **fd0** device. Your floppy device may be different; the **fd0** is for the 8086/8088 PC. If the dump diskette was part of a multi-volume set, **dumpdir** will ask you to mount the following diskettes. When you have mounted the diskette, respond with **<RETURN>**.

Summary

This section discusses the operation of the tools used to dump and restore files with the COHERENT system. Using the procedures outlined here will significantly improve the safety of your information.

10. Tools for the administrator

This section discusses tools that will make your job as system administrator easier. Other useful tools for communicating with users, such as **mail**, **write**, and **msg**, are described in the *Introduction to the COHERENT System*.

help — Help with a command

If you know what a command does, but need a reminder on the exact format of the command, the **help** command will be useful. To use **help**, type

```
help help
```

and the COHERENT system will reply

```
help -- print concise command description
Usage: help [ command ... ]
help is intended for refresher information.
To learn about a new command, man or learn
is preferable.
```

man — Full description of commands

If you need more information than is provided by **help**, use the **man** command. It provides specific information on all commands and their options.

To get information on the **man** command itself, type

```
man man
```

Since the COHERENT system is available on systems with hard disk drives as small as 5 megabytes, the online manual pages are optional with some implementations of the system. For these systems, you can obtain the online manual pages for a nominal fee.

The manual information is available in printed form in the *COHERENT Command Manual* and the *COHERENT System Manual*.

ps — List active processes

Each command or program in the COHERENT system is a separate *process*. Each process in the system is assigned a number called the *process id*, or *PID*. You may need to control the actions of users occasionally, and you will do so by controlling processes.

Each user logged into the system has one or more processes. Except in special circumstances, the first process that a user has is the shell, or command line interpreter. Commands that the user types in will be run by the shell. The shell normally waits for termination of each command before processing the next. However, the use of '&' creates simultaneous processes executed while the shell accepts other commands.

You can examine the processes associated with your login, or all processes in the system, with the **ps** command. Type

```
ps
```

and the result will be similar to

```
TTY PID
31: 37 -sh
31: 4010 ps
```

The first column

```
TTY
31:
31:
```

is the terminal number. This number is taken from the **/etc/ttys** file with the **tty** removed from the beginning. The **tty** number is also output by the **who** command. The second column

```
 PID
  37
4010
```

lists the corresponding process id (PID). The third column contains the name of the command and command parameters:

```
-sh
ps
```

The **-sh** represents the shell process, and the **ps** represents the **ps** command process.

To see all the processes, type

```
ps -a
```

and the result will be similar to

```
TTY PID
3a: 41 -sh
39: 42 -sh
32: 47 - 3
31: 48 - 3
34: 193 -sh
36: 634 -sh
3e: 1738 -sh
20: 2568 -sh
3e: 2581 su
3c: 6317 -sh
3c: 6322 su
3f: 7333 - P
35: 7789 - P
3c: 8058
3d: 9053 - P
33: 9076 - P
30: 9814 -sh
30: 9829 ps -a
```

This display will be quite different from system to system.

For a full description of all **ps** options, see

```
man ps
```

kill — Terminating processes

There are occasions as system administrator when you will need to log other users out of the system. You may need to bring the system down quickly. Perhaps a user forgot to log out before leaving the terminal and did not see your broadcast message requesting that all users log out.

The **kill** command, when used by the superuser, will terminate processes. To log out a user whose shell has process number 300, use the command

```
kill -9 300
```

You must be logged in as **root** or use the **su** mode to **kill** a process that belongs to another user. The **kill** command has other uses as well—see **man kill**.

Summary

There are several tools that will help your task as system administrator. Some of them, like **man** and **help**, are useful to other users as well.

11. System security

Many COHERENT system installations have different groups of users whose tasks are separate. Consider a class of students all doing homework on the computer. No student should be able to read another student's files, but the teacher should be allowed to check each student's progress.

By using the flexible protection mechanisms provided as part of the COHERENT system, you can set up system security to suit the needs of your users.

Passwords

Passwords provide the first level of COHERENT system security.

For systems with passwords, each user with a password must issue the password as part of the login process. If the password is entered incorrectly, the user cannot log in.

You assign passwords when you create the user entry as described in the above section "Establishing a user base". If you do not assign a password to a user, anyone will be able to log in as that user.

In any system with passwords, it is especially important to assign a password to the **root**, or *superuser*. If it does not require a password, any user can log in as **root** and automatically have access to the powerful tools that control the operation of the system.

Similarly, any user with a password can control access to his files. Once the administrator assigns the password, the user can change it with the command **passwd**. However, because of higher privileges, the **root** user can always access everyone's files.

The passwords are kept in a file **/etc/passwd** with the rest of the user login information. Passwords are encrypted, so that reading the **/etc/passwd** file will not reveal passwords.

File protection

The second level of COHERENT system security is in file protection. A user can set each of three categories of protection for each of his files. A standard protection, or *access permission*, is given to each file when it is created.

The three categories of permissions are for the user, for other users in the same group, and for all other users. To see the levels of protection of current files, type the command

```
ls -l
```

and see the section on files in *Introduction to the COHERENT System*.

For each attempt to access a file, the COHERENT system first checks if the file belongs to the user, to someone in the user's group, or neither. The corresponding permission field is then used to decide if the access will be granted or not.

The permissions granted depend upon the type of access: *read*, *write*, or *execute*. Execute permission is needed for command scripts or executable commands. Read permission means that the file can be read. Write permission means that the file can be changed or deleted.

Permissions are interpreted differently if the file is a directory. Read permission for a directory means that the user may read the file names in the directory. Write permission means that the user may create files in the directory. Finally, execute permission means that the user may access a specific name in the directory. Thus, if a directory denies read permission and grants execute permission, the names in the directory may not be read, but a specific name may be referenced.

Changing file protections

The command **chmod** changes the access permission of a file. The owner of a file and the superuser are always allowed to change the permission of a file.

To make a file unwritable, type the command

```
chmod -w file
```

To make a shell command file executable, type

```
chmod +x script
```

When files are created, they are given the protection specified in the file creation mask. If a user needs a different protection to be specified for most of his files, you can put into his **.profile** file a line of the form

```
umask 022
```

to specify the default protection.

For a full discussion of the options of **chmod** and how to use **umask**, see the *COHERENT Command Manual*.

Encryption

The **crypt** command provides a third level of system security. It allows a user to encode and decode information in a file. Since the superuser can access any file in the system without restriction, sensitive information must be protected by encoding. For details about encryption, see the *COHERENT Command Manual*.

Summary

The COHERENT system is flexible in the degree of security that you can implement. You can require each user to enter a password, and set the default permission of each file in the system.

12. Tour through the file system

This section describes the layout of the COHERENT file system and points out files of interest to the system administrator.

General file system layout

The base of the file system is the **root** directory, whose name is simply

```
/
```

Most of the files in the root are directories. To list the files in the **root** directory, type

```
lc /
```

/bin

Most of the commonly used commands are programs contained in **/bin**, such as the **lc** command used in the above example. To speed finding commands, less commonly used commands are put in **/usr/bin**.

The command line interpreter, called the *shell* or **sh**, does not automatically look in **/bin** for commands but consults a special shell *parameter* that specifies in which directories to look for commands. Each time a user types a command to the shell, the shell examines the variable **PATH** to determine where commands are to be found. A typical value for **PATH** is

```
/bin:/usr/bin:.
```

This tells the shell to look for commands in three places (in this order): **/bin**, **/usr/bin**, and finally ., the current directory. The shell will not consult **PATH** if the command contains one or more / characters, indicating a complete or partial path specification.

/dev

Devices in the COHERENT system are accessed through files in the directory **/dev**. If there is a line printer available on the system named **lp**, you can print characters from a file named **testdata** by typing the command

```
cat testdata >/dev/lp
```

All devices on the system are represented in the **/dev** directory.

/drv

A unique feature of the COHERENT system is the concept of loadable device drivers. This feature allows COHERENT system programmers to write their own device drivers without modifying the rest of the system. Drivers may be unloaded, modified and reloaded without halting and rebooting the system.

These drivers are kept in /drv.

To load a driver, say

```
/etc/load /drv/lp
```

Unfortunately, not all hardware is capable of supporting loadable drivers. On such systems, the **/drv** directory is not supplied.

/etc

Several commands that you will use in your role as system administrator are kept in the **/etc** directory. These are described in detail elsewhere in this guide. They include accounting commands, system booting commands, file system mounting commands, file system creation commands, and system time control commands.

Also in **/etc** are several data files used in system administration. These include **/etc/passwd**, the file containing user names, ids, and passwords; news files; and a file **/etc/ttys** describing the properties of each user terminal attached to the system. The message of the day is kept in **/etc/motd**, and the news is kept in **/etc/news**. The file **/etc/newslog** keeps a note of which messages each user has seen.

/lib

Many useful I/O and mathematical functions are provided with the COHERENT system to assist in writing C programs. These and other libraries, along with the phases of the C compiler itself, are included in **/lib**. This directory includes files containing standard system calls, standard I/O handling, and mathematical routines such as **sin**, **cos**, and **log**.

/usr

The directory **/usr** contains user directories, along with a few system directories.

The system directories are **/usr/adm**, which contains additional information of interest to the system administrator.

/usr/bin contains less-often used commands.

/usr/games contains several computer games.

/usr/games/fortune will give a new fortune each time that it is called. A call to this game can be placed in a user's **.profile** file. The user will get a fortune printed each time that he logs on. To add fortunes of your own, edit the file **/usr/games/lib/fortunes**.

/usr/games/moo is a number guessing game. The program generates a number with four digits, all different. You are asked to enter a number with four different digits. If you have guessed the number with the digits in the proper order, the program responds with

```
Right!
```

and invites you to play a new game.

If your guess is incorrect, the program will tell you how many digits you guessed in their proper positions—labeled "bulls"—and how many digits you guessed, but not in their proper positions—called "cows". You can keep guessing until your guess is correct.

The directory **/usr/include** contains include files for C programs, such as **stdio.h**. Other include files define formats of files and other important data structures in the system.

/usr/lib contains the **ms** and **man** macros for the **nroff** text processor, unit conversion tables for **units** and the **crontab** file used to hold commands for **cron**.

/usr/man contains manual sections referenced by the **man** and **help** commands.

/usr/pub contains public files, such as phone numbers.

/usr/spool contains information for line printer spooling, and not yet received mail for each user.

/u

In some systems, users' directories are placed on a separate device for space reasons. Since a separate device has a separate file system, the directory on that device is called **/u**.

Summary

It is helpful for you as system administrator to know where many things that are part of the COHERENT system are kept. This section gives you a quick tour of the file system.

13. How booting works

This section discusses the events that take place while starting the system. You do not need to read this section to know how to boot, but if you are interested in the details, or want to tailor the system to your individual needs, this section will help.

Two I/O devices are involved in bootstrapping. The first device is called the *boot* device, and contains the program necessary to bring in the COHERENT system and start it running. The second device is called the **root** device, and will contain the root file system after the system is running. In most cases, these two devices are the same physical device.

Startup events

This section briefly describes what takes place when you perform the startup of the COHERENT system.

The initial installation of the system loads information from tape, floppy disk, or other medium to the hard disk. The release notes for your specific version of the COHERENT system describe the installation procedure in detail.

The procedure described above in the "Regular startup" section first loads a small program from a floppy disk. This program, called the bootstrap program, then reads in the COHERENT system itself. The boot procedure may be different on your system.

After the system image **/coherent** has been loaded from the root device, it starts a program named **idle**. This program uses all left-over computer time and performs other control functions.

The program **init** controls the operation of the system from this point on.

It loads the terminal device drivers. Then it calls the **shell** to handle commands from the system console. The shell responds by prompting with #, expecting regular commands. At this time, the system is in *single user* mode, meaning that no other users may log in to the system. The shell is running in superuser mode and only the console user is logged in.

At this point, you can enter commands to the system in a normal fashion. One difference from normal operation is that the system is in single user mode, so special processing can take place before

other users log in. Being in single user mode gives you the opportunity to **check** the file system and perform other administrative tasks before other users log in to the system.

When administrative activities are finished, you should type **<ctrl-D>**. This terminates single user operation; **init** then opens the system to other users.

The file **/etc/rc**, contains shell commands which the system executes just before making the system available to other users. This file typically includes commands to delete temporary files and mount standard devices. It also performs any installation-specific commands you require. As system administrator, you maintain this file. You must be sure that it is properly updated and never removed.

One command that must be included in **/etc/rc** is **/etc/update&**, which periodically calls **sync** to update buffered data to the disk. On many systems, **/etc/rc** also executes **/etc/swap**, called the *swapper*. The swapper writes inactive program images to the swapping device to make room for other user programs that are ready to run.

init also maintains the **/etc/utmp** and **/etc/wtmp** files, noting log in and log out of users.

Files used during startup

This section discusses files used during booting, distinguishing the files needed while the system is single user from those needed during the multiuser part of the booting process.

/etc/load	load device drivers
/etc/init	initiate a process on each terminal line, call login when appropriate
/drv/tty	device drivers for the terminals
/drv/swap	swapper device driver
/bin/sh	shell

The following files are needed after the system has gone multiuser:

/etc/rc	shell commands for startup
/etc/ttys	information about terminals
/bin/login	login program
/etc/utmp	who file

Summary

The process of starting the COHERENT system involves many events happening in response to your commands. This section details them so that you can have a better understanding of the startup process.

14. Devices, files and drivers

This section discusses files, special files, and devices.

The COHERENT system provides device-independent I/O. Devices and files are handled in a consistent way. Each I/O device is represented as a *special file* in directory **/dev**. If your system has a line printer device named **lp**, you can list a file, named **prog** for example, on the printer by saying

```
cat prog >/dev/lp
```

Another example is to copy the file **prog** with the **cp** command to your terminal:

```
cp prog /dev/tty
```

There are two types of special files represented in **/dev**, and when you list **/dev** with **lc** it will separate them.

The first type is a *block special* file. This type includes disks and magnetic tape. These devices are read and written in blocks of 512 bytes, and can be randomly accessed. (As a practical note, notice that magnetic tape can be read in a random fashion only by positioning backwards and forwards one record at a time; disks can be read or written in a totally random fashion.)

The I/O to and from block devices is buffered to improve overall system performance. When a program writes a block of data, the data will be held in the buffer to be written at a later time. If the same block is read twice in a row, the data for it is still available in memory and does not have to be fetched from the physical device.

A special program named **/etc/update** forces all buffered data to the physical device periodically by calling the command **sync**, to protect against losing data in the case of a system halt, such as a power failure. If you must bring the system down, you must force the latest data to be written by typing the command **sync**.

Character special files

The second kind of special file is called a *character special* file. Included in this class are devices that are not block special: terminals, printers, and so on.

However, disks and tapes can also be treated as character special files. For every block special file for a disk such as

```
/dev/rm00
```

there is usually a character special file

```
/dev/rrm00
```

Character special files are sometimes called *raw* files, hence the prefix **r** in **rrm00**. A raw file has no buffering or other intermediate processing performed on its information. This difference is an efficient benefit to the **dump**, **restor**, and **icheck** commands which do their own buffering.

tty processing

One special set of devices has other processing—the **tty** or terminal files. A terminal special file with this special processing is called a *cooked* device. The processing includes handling the **kill**, **erase**, **interrupt**, **quit**, **stop**, **start**, and **end-of-file** characters. Processing can be disabled with the **stty** command so that the program deals with the raw device. However, using a raw **tty** device will generally have negative effects on COHERENT system performance.

Summary

Files and devices are handled in a unique way under the COHERENT system. This scheme provides unusual flexibility in system use.

15. Creating and mounting file systems

This section discusses the creation of a file system on an empty device. You will need to do this to prepare a new removable disk pack or diskette before mounting it on your system. You may also do this if you want to rebuild one of your file systems from scratch.

mkfs — Create a file system

To create a file system, issue the command

```
/etc/mkfs special proto
```

where *special* is the special file on which the file system is to be built. The option *proto* is either a number or a file name. If it is a number, **mkfs** will build a file system of that size in blocks. For example, to create a filesystem on a 720-block floppy disk, use

```
/etc/mkfs /dev/fd0 720
```

Otherwise, *proto* is presumed to be a file, called a *prototype file*, containing a description of the file system to be built.

A sample *proto* prototype file is as follows:

```
/dev/null
800 128
d--755 3 1
        dev d--555 3 1
        $
        etc d--755 3 1
        $
        bin d--755 3 1
        $
        drv d--755 3 1
        $
        lib d--755 3 1
        $
        usr d--755 3 1
        $
        tmp d--777 3 1
        $
$
```

The file must consist of a set of character strings separated by spaces and newlines describing how the file system is to be built. These strings will be delineated by blanks or appear on different lines.

The first string **mkfs** will look for is the pathname of the file containing the bootstrap program to be written onto the first block of the device. In this case, the string is **/dev/null**, or the null device, so there will be no bootstrap written.

Next is the decimal number of blocks to be included in the file system; in this example, 800. This may be less than the total number of blocks on the device.

Third is the number of *i-nodes* to be on the *i-list*, in this case, 128. The next section describes the purpose of i-nodes. Briefly, there must be one i-node available for each file that will reside on the disk.

The next strings describe the root file. Generally, this root file is a directory, and therefore **mkfs** must be told about other files to be

created in this directory. Each of the files described to **mkfs** will be of the same form.

The strings

```
d--755 3 1
```

describe the properties of the root file. **d--755** describe the mode of the file, and the **3** and **1** describe the owner of the file.

The string describing the *mode* of the file is six characters long. The first of these characters specifies the type of the file, where **d** means directory, **b** means block special file, and **c** means character special file. Thus, the root file is a **d**irectory file.

The second character must be **u** if the set user id mode is to be in effect, and "–" if not.

The third character must be **g** if the set group id mode is to be in effect, and "–" if not.

The remaining three characters **755** specify the file access permissions. These are described fully in the manual for **chmod**, which you can obtain by typing

```
man chmod
```

or by reading the **chmod** section of the *COHERENT Command Manual*.

The user id and group id are both integers as defined in the password file.

Following these numbers is the contents of the file. Since the root in this example is a directory, the contents are other files. These files are described in a similar fashion, except that the file name comes before the mode specification.

To make the prototype file more readable, files in a directory are indented. At the end of each file description is a **$**.

Consider the description of the file:

```
dev d--555 3 1
$
```

First comes a string **dev** specifying the name of the file. Next is the mode description **d--555**. Third is the user id **3**. Fourth is the group id **1**.

Next comes the description of the contents of the file. In this instance, no initial contents are given. Finally, the character **$** signifies the end of the file description.

If the type of file is either a block special file or a character special file, the next strings are the major and minor device numbers, in that order.

The final **$** in this example signals the end of the description of the root file on the device.

If the file is a regular file, the *contents* is the pathname of a file that is to be copied to the file being created.

mount — Mounting file systems

Some disk storage systems have removable modules, or *packs*. The COHERENT system allows you to put on and remove these packs while the system is running.

Each disk pack must have a file system already on it before mounting it. *If it doesn't, mounting it can cause the COHERENT system to halt and possibly damage the file system.* File systems are created by the **mkfs** command, as described above.

There must be a directory on the permanent file system (generally the primary disk or possibly a non-removable disk) that all files and directories for the mountable file system will be attached to. The directory should not have any files in it at the time of mounting. If it does, they will be inaccessible until the filesystem is unmounted. The root file on each physical device must be a file in a directory on another device, usually the **root** device.

The special file for the removable device must also exist in **/dev**. Special files are created with the **mknod** command.

To illustrate the **mount** command, assume the device is **/dev/hd1**, and that the directory that the files are to be attached to is **/u**. The **mount** command is

```
/etc/mount /dev/hd1 /u
```

or, to generalize

```
/etc/mount special directory
```

special is the special file corresponding to the mountable device. *directory* names the branch of the file system where files are to appear.

The file system can be protected as read only with the **-r** option:

```
/etc/mount special directory -r
```

The **mount** command will tell you which file systems are currently mounted if you give no arguments:

```
/etc/mount
```

To remove the file system, use the **umount** command and the name of the device:

```
/etc/umount /dev/hd1
```

Summary

Each device that has no file system on it must be prepared by the command **mkfs**. This includes disks that are removable, such as diskettes or disk packs.

The ability to mount disks with file systems on them can greatly increase the convenience of expanded offline storage available to your COHERENT system.

16. File system integrity

Each COHERENT file system is constructed in the same manner, regardless of which device it resides in. This results from the device independent I/O design principle of the COHERENT system.

In order to be sure that the file system on each device is in proper condition, you should check each file system's integrity periodically, preferably daily. This section discusses how the file system is constructed and how to check it for errors.

How a file system is built

Each device holding a file system can be thought of as a list or array of randomly accessible blocks. Each block contains 512 bytes. The array of blocks on each device is divided into several regions.

A list of available data blocks for a device is kept by the system on the device. This set of blocks is called the *free list*. Whenever a file is created or expanded, a data block is allocated from this *free list*. When all links to a file are removed, the file is deleted by putting all the data blocks onto the free list.

Similarly, a list of available i-nodes is maintained. This list is kept in memory and in part of the superblock (described below). However, each i-node also maintains information telling if it is free.

If the disk surface has defects, the block corresponding to that spot cannot be read or written. A list of bad blocks is kept so that files can be built around these bad spots.

The very first block, block 0, is reserved on each device for a boot program.

The next block, block 1, is called the *superblock* and contains information describing how the rest of the blocks of the device are partitioned.

Beginning with block 2 is the list of i-nodes, called the *i-list*. An i-node contains information about the location of each file's data, and the *mode* of the file. Each file has an i-node number called the *i-number*. From the i-node number, the system can directly find the i-node. The i-numbers uniquely identify the file on a device. The first i-node is reserved for the bad block list. The second i-node is

for the root file system. Each active i-node contains information about a file on that device.

Each user has a user name and a user id. Both of these numbers are assigned by you in your capacity as system administrator using the command **newusr**. The user id numbers begin with one and continue consecutively.

Similarly, each group of users is assigned a group name, as well as a group id.

The i-node contains several pieces of important information. The ownership of the file is given by the user id and group id. The data blocks of the file are indicated in the i-node. Also present are the size (in bytes) of the file, and the mode of the file. Finally, times of attribute change, modification and last access are maintained.

Directories are like other files, except that they can never be directly written by the user. A directory entry simply contains the file name and the i-number. When the link command **ln** is used, the new name is entered into the directory with the i-number of the existing file and the link count in the i-node is incremented. If one of the links to a file is deleted with a **rm** command, the directory entry is removed, and the link count of the i-node is decremented. Only when the link count reaches zero is the i-node freed, and the data part of the file is also freed.

icheck — Checking file system consistency

The **icheck** command checks the consistency of the blocks listed in the free list against the blocks that are actually used. For example

```
icheck /dev/hd0
```

where **/dev/hd0** is a disk device, will check the file system located on device **/dev/hd0**.

If possible you should **umount** the file system before you check it. You cannot **umount** the root file system. If you can't unmount it, be sure that no other users are on the system and that the system is immediately rebooted *without* performing a **sync**. If other users are creating or expanding files while the file systems are being checked, false errors will be reported.

If any discrepancies are found by **icheck**, appropriate messages are written on the terminal. An absence of messages indicates that there are no problems with the file system. An error message

```
bad ifree
```

should be treated as a comment. This message means that i-nodes that are in use are also noted in the free i-node list. This is not a problem since each i-node is marked used or unused in the i-node itself. This information is automatically checked by the system before using an i-node from the free list.

The message

```
dups in free
```

means that a block is listed more than once in the free list. Each block should be listed no more than once.

The message

```
bad freelist
```

means that the free list contains block numbers that do not exist on the device.

These errors must be corrected before the file system is mounted. There are two ways to correct these problems. First, you can **restor** all the files from their latest backup. Secondly, you can have **icheck** rebuild the free list by using the **-s** option:

```
icheck -s /dev/hd0
```

If you could not **umount** the file system, you should immediately reboot *without* performing a **sync**. Otherwise, the old and incorrect superblock will be retained. If the file system was unmounted when you started the **icheck** procedure, you can remount the system after running **icheck**. For more information, use the **man** command for **icheck**.

The error message

```
dup
```

means that a data block belongs to more than one i-node or to an i-node and the free list.

dcheck — Directory consistency check

As noted earlier, directories are files that contain file names and i-numbers. The **dcheck** command compares the number of directory entries referencing each i-node against the link count in the i-node itself. If there are differences, they are noted.

Two error conditions are of interest.

The first occurs if the link count in the i-node is less than the number of directory references pointing to it. This is dangerous, since it is possible to delete the data part of the file while there are still files referencing it.

When this error occurs, you must immediately mount the file system and delete the directory entries referencing this i-node, or use the **-s** option:

```
dcheck -s /dev/hd0
```

The second class of error is not serious. If the link count is larger than the number of directory references, then once all directory entries have been removed, the i-node will remain. The only disadvantage will be that the file data cannot be deleted, causing a waste of file space.

The check command

The **check** command is an easy way to call **icheck** and **dcheck** in that order for each file system *fs* in turn:

```
check fs
```

If errors are discovered, you can then repair them using techniques outlined in the **icheck** and **dcheck** sections above. The command

```
check -s fs
```

will try to repair any errors automatically.

Summary

This section presents details of COHERENT filesystem construction in order to help you preserve the integrity of all files in the system. Also, details of the operation of checking commands are presented.

Index

User Reaction Report

To keep this manual and COHERENT free of bugs and facilitate future improvements, we would appreciate receiving your reactions. Please fill in the appropriate sections below and mail to us. Thank you.

Mark Williams Company
1430 W. Wrightwood Avenue
Chicago, IL 60614

Name: _____

Company: _____

Address: _____

_____

Phone: _____ Date: _____

Version and hardware used: _____

Did you find any errors in the manual? _____

_____

_____

Can you suggest any improvements to the manual? _____

_____

_____

Did you find any bugs in the software? _____

_____

_____

Can you suggest improvements or enhancements to the software?

_____

_____

_____

Additional comments: (Please use other side.)