Burroughs

XE 500 CENTIX™

Administration Guide

Relative To Release Level 6.0
Priced Item
November 1986

Distribution Code SA
Printed in U S America
1207776

Burroughs

XE 500 CENTIX™

Administration Guide

Relative To Release Level 6.0
Priced Item
November 1986

Distribution Code SA
Printed in U S America
1207776

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Burroughs, if any, with respect to the products described in this document are set forth in such License or Agreement. Burroughs cannot accept any financial or other responsibility that may be the result of your use of the information or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded, using the Product Improvement Card at the back of this manual, or remarks may be addressed directly to Burroughs Corporation, Corporate Product Information East, 209 W. Lancaster Ave., Paoli, PA 19301, U.S.A.

About This Guide

Purpose

This guide describes how to administer the XE 500 CENTIX™ system using CENTIX shell commands and BTOS Master Commands (MCommands). The CENTIX system also provides centrEASE™, a menu-driven administrative facility, with which you can perform most administrative functions. For information on centrEASE, see the *centrEASE Operations Reference Manual*.

Scope

This guide describes the tasks needed to administer the XE 500 CENTIX operating system after it has been installed. For installation information, see the *CENTIX Installation and Implementation Guide*.

Audience

The audience for this guide is the experienced CENTIX administrator. An inexperienced system administrator should refer instead to the *centrEASE Operations Reference Manual*.

Prerequisites

The system administrator who uses this guide should be very familiar with the CENTIX system.

How to Use This Guide

Use this guide when performing administrative tasks without using the administrative facility, centrEASE.

Organization

This manual contains the following sections and appendices:

Section 1, Introduction, defines the role of the system administrator.

Section 2, Starting Up and Bringing Down the CENTIX System, describes the system software bootup and shutdown procedures.

Section 3, Setting Up the /etc/inittab*nn* Files, describes how to read and alter the /etc/inittab*nn* files, which control the system operating levels.

Section 4, Configuring I/O Devices, describes I/O device file names and how to configure devices into the system.

Section 5, Managing File Systems, describes how to create, manage, and check CENTIX file systems. The internal file system structure is also described.

Section 6, Managing User Accounts, explains how to add, delete, and organize users on the CENTIX system.

Section 7, Managing System Security, describes how to use the security functions of the CENTIX system to keep your system secure.

Section 8, Backing Up and Restoring Files, describes how to save and recover files.

Section 9, Configuring and Administering the Printer Spoolers, describes how to set up and manage the printer spoolers.

Section 10, Administering the uucp Network, describes how to set up and use the uucp communications network.

Section 11, Tracking System Activity, describes how to use the system activity package to monitor the system.

Section 12, Running Periodic Jobs with the **cron** Command, describes how to create and alter user **cron** files.

Section 13, Accessing BTOS from CENTIX, describes the methods you use to copy from, edit files in, and administer the BTOS operating system.

Section 14, Initializing and Verifying XE 500 Volumes, describes how to format a disk.

Section 15, Running the System in Different Operating Modes, explains how to run the system in normal, restricted, and customized modes.

Section 16, Interpreting and Recovering from Error Conditions, explains how to analyze system errors and provides procedures for recovering from some error conditions.

Appendix A, **fsck** Messages, lists the messages that can be generated by **fsck**, the file system checking program.

Appendix B, XE 500 BTOS-Specific System Error Codes, lists the error codes that are specific to the XE 500 BTOS operating system. For a complete list of BTOS Error Codes, see the *BTOS Status Codes Reference Manual*.

Appendix C, Hardware Configuration Information, includes figures and tables that define the conventions for naming disks and processors and numbering slots.

Appendix D, FCF Reporting Procedure, outlines how to file a Field Communication Form when you have system problems.

A glossary and index follow Appendix D.

Related Product Information

BTOS Status Codes Reference Manual

This manual lists and describes the BTOS error codes that can occur on the XE 500 CENTIX system.

XE 500 BTOS Customizer Operations Guide

This guide discusses customizing the BTOS portion of the CENTIX operating system.

XE 500 BTOS Operations Reference Manual

This manual describes the BTOS Master Commands that can be used to administer the BTOS portion of the CENTIX operating system.

XE 500 CENTIX centrEASE Operations Reference Manual

This manual describes how to use the CENTIX administrative facility centrEASE.

XE 500 CENTIX Installation and Implementation Guide

This guide describes how to install and implement the CENTIX operating system on the XE 500.

XE 500 CENTIX Kernel Customizer Operations Guide

This guide describes how to customize the CENTIX kernel.

XE 500 CENTIX Operations Guide

This guide describes how to operate the CENTIX system. The CENTIX text editors are also described in detail.

XE 500 CENTIX Operations Reference Manual

This manual lists and describes all CENTIX shell commands, library functions, systems calls, and special files.

XE 550 System Capabilities Overview

This manual provides a technical overview of the XE 500 CENTIX system.

Conventions Used in This Guide

- Both BTOS and CENTIX commands are shown in boldface.
- Variables are shown in italics. For example, in the following command, *oldfile* and *newfile* are both variables.

  # cp *oldfile newfile*

  When you enter the actual command, you substitute the name of the file which you are copying and the file to which you are copying for *oldfile* and *newfile*.
- In both BTOS and CENTIX command lines, optional fields are enclosed in square brackets.

- Data that you are to enter at your terminal keyboard are shown between quotation marks ("...") or as indented text. You should not enter the quotation marks themselves unless specifically told to do so in the text.

Contents

Illustrations

Tables

Introduction

This guide is used in administering the XE 500 CENTIX system. The CENTIX operating system is based on UNIX™ System V and is licensed from AT&T.

The system administrator is responsible for all system management tasks. In particular, the administrator handles any task that affects more than one user, and any task that affects system security.

The Administrator as Superuser

The administrator on this system is, by convention, referred to as superuser. Superuser has access to many commands and files that other users cannot access. To become superuser, the administrator signs onto the system as "root". By signing on as root, you have automatic access to all superuser functions. These include:

- Writing to or reading from any ordinary or special file.
- Creating a file in or deleting a file from any directory.
- Executing all administrative commands.
- Using all command options.
- Using the BTOS Master Commands (MCommands). See "BTOS in the CENTIX System," below.

The superuser prompt is a pound sign (#).

The administrator performs administrative functions at the system console. You can also use the system console as a user terminal by signing onto the console as a user other than root. If you need to change the console—or any user terminal—from normal user status to superuser status, use the **su** command. In normal user status, the user dollar sign prompt is on the terminal. Enter:

```
$ su
```

A password prompt appears. Enter the password for root. The superuser pound sign prompt appears. You can begin performing superuser functions.

™ *UNIX is a trademark of AT&T Bell Laboratories.*

BTOS in the CENTIX System

The XE 500 CENTIX system is based on two software operating systems: CENTIX and a Burroughs workstation operating system called BTOS.

All CENTIX-based system software and applications run on the Applications Processor (AP) in the XE 500. If your XE 500 contains more than one AP, CENTIX can be configured to run on all of the APs in the system.

BTOS runs on all other processor types. A BTOS-based processor, the master File Processor (FP00), is responsible for controlling the front panel and downloading the other processors with their operating systems. BTOS-based processors also control input/output (I/O) to the peripheral devices, such as disk drives, tape drives, terminals, printers, data comm lines, and so on.

To properly install and maintain the system, you must manage both BTOS and CENTIX files and services. As administrator, you can, of course, use any CENTIX commands to administer the CENTIX portion of your system. To administer BTOS, you must access the BTOS MCommands *through* CENTIX. There are two ways to do this:

- You can use the "Issuing BTOS Commands through CENTIX" function of the centrEASE administrative facility. The next subsection, "The centrEASE Administrative Facility," gives a brief explanation of centrEASE. The facility is explained in detail in the *centrEASE Operations Reference Manual*.
- You can use the CENTIX **ofcli** command to invoke the BTOS MUtilities. This is explained in detail in Section 13 of this guide.

Throughout this guide, you are told when you must administer the CENTIX portion of your system, and when you must administer the BTOS portion of your system.

The centrEASE Administrative Facility

centrEASE is a menu–driven, interactive facility that you can use to perform many of the administrative tasks described in this guide. Throughout the guide, tasks that can also be done through centrEASE are noted.

Unless you are an experienced CENTIX administrator, you should use centrEASE, rather than the processes described in this guide. centrEASE not only allows you to perform administrative tasks quickly and easily, it also provides safeguards that can stop you from doing damage to your file system.

The tasks that you can perform through centrEASE are:

- Manage user accounts
- Manage file systems
- Issue BTOS commands through CENTIX
- Reconfigure the system
- Install CENTIX products
- Print configuration reports
- Back up and restore
- Manage log files
- Maintain other products

See your *centrEASE Operations Reference Manual* for instructions on how to use centrEASE.

Starting Up and Bringing Down the CENTIX System

Starting Up the System Software

To boot up your system, follow these steps:

1 Power up the XE 500. This is explained in detail in the CENTIX *Installation and Implementation Guide*.

2 Turn on your system console. If your console is not turned on when you boot up the system, the boot up procedure may not complete.

3 Turn the front panel keyswitch to NORMAL. While the system software is booting up, the front panel status display shows numbers ranging from 1 to 20. When a 20 is displayed, the BTOS operating system is running. You can now boot up the CENTIX system at your console. (If the display stops at 19, your system has not come up correctly. See the paragraph at the end of this subsection.)

Note: *If there are problems with the boot up procedure, the front panel display may show an error code. See both Appendix B in this Guide and the* BTOS Status Codes Reference Manual *for lists of XE 500 status codes.*

When you first turn on your system, your CENTIX system is in single user mode. Single user mode is one of several operating modes defined by the /etc/inittab*nn* files (see Section 3). In single user mode, only the console is active. No other terminal can be accessed.

4 A few minutes after you boot up the system, the current date and time and the following message appear on your console:

```
Is the above date/time correct (y or n)
```

If they are not correct, enter n, then the correct date and time, then press RETURN. If the date and time are correct, enter y.

Note: *If you have a mixed system (both full CENTIX and full BTOS operating systems), the time will not be correct the first time you bring up the system. Do not correct the time. Enter y, then proceed. See "For Mixed Systems Only: Changing the Time Zone Variable."*

5 This message then appears:

```
Do you want to check the file systems? (y or n)
```

Enter y. The system automatically runs the **fsck** file system checking program (see Section 5). **fsck** checks and makes minor repairs to the file systems.

6 The **fsck** program takes a few minutes to run. As each file system is checked, status information on that file system appears on your screen.

Messages then appear telling you the programs that are being started by the /etc/inittab00 file (see Section 3). A few moments later, the Burroughs logo appears on the console screen. At the lower left of the screen appears:

```
login:
```

Enter:

```
root
```

A password prompt appears. Enter the password for root (see Section 6 for instructions on assigning passwords).

When the login appears at the console, the system has entered multiuser mode. Multiuser mode is one of the operating modes defined by the /etc/inittab*nn* files (see Section 3). In multiuser mode, all configured terminals can access the system.

When the system enters multiuser mode, logins appear at any terminals that are configured and turned on. Users can sign on and begin work. The environment in which the users terminals are brought up—including opening messages, terminal definitions, and so on—are defined in the /etc/profile file. See "Changing the /etc/profile File," at the end of this section.

7 A message welcoming you to the system appears. Under it is the root prompt, a pound sign (#). You can now begin entering commands.

If, when bringing up your system, the front panel display stops at 19, the boot up procedure will not complete and you will not get a login at the console. To determine what is wrong, follow these steps:

1 At one of the terminals connected to your system, there should be a prompt for a new run level. At the terminal, enter "6", and press the RETURN key.

2 The login prompt appears at your console. Initiate the **fsck** program to check your file systems. Enter:

```
# /etc/fsck
```

See Section 5 for details on the **fsck** program.

Bringing Down the System Software

To bring down the system software, follow these steps:

Warning: *Do not turn off your system without running the* ***shutdown*** *command. You can do extensive damage to your file systems.*

---

1 At the console, initiate the **shutdown** command. Enter:

```
# shutdown
```

This message appears at the console screen:

```
SHUTDOWN PROGRAM

current date

SYSTEM BEING BROUGHT DOWN NOW!!

All processes being killed.

Do you want to continue? (y or n)
```

The message SYSTEM BEING BROUGHT DOWN NOW is also broadcast to any user terminals currently logged on. All users should log off within the next minute.

2 If you have changed your mind about bringing down the system, enter n. The **shutdown** procedure stops, and a prompt appears.

   To continue with the **shutdown** procedure, enter y.

3 A message appears telling you to wait for an okay before you stop or reset the processor. DO NOT TURN OFF YOUR SYSTEM UNTIL A MESSAGE APPEARS GIVING YOU THE OKAY TO STOP OR RESET THE PROCESSOR.

4 After a few minutes, these messages appear at the console:

```
INIT: New run level: 5

INIT: New run level: S

INIT: SINGLE USER MODE

Wait for ok (about 15 seconds) before
stopping or resetting processor...
```

   After about 15 seconds, this message appears:

```
ok to stop or reset processor

#
```

   When this message appears, the system has entered single user mode. You can now turn the XE 500 keyswitch to STOP.

The **shutdown** command has several options. With the **-g** option, you can change the amount of time that the system waits between warning users and turning off the system (60 seconds is the default). With the **-i** option, you can select which operating mode the system enters at the end of the **shutdown** procedure (single user mode is the default). See your *CENTIX Operations Reference Manual* for details on the **shutdown** options.

Taking the System to Single User Mode

There are two methods of taking the system to single user mode:

- Use the procedure described above for using **shutdown**.
- Use the **init** command. Enter:

```
# init s
```

When you use this method, DO NOT turn off the XE 500 after you enter single user mode. You must still run the **shutdown** procedure to guarantee an orderly closing down of your file systems.

Taking the System from Single User to Multiuser Mode

To reenter multiuser mode from single user mode without rebooting the system, enter:

```
# init 2
```

This message appears:

```
INIT: New run level 2
```

When the Burroughs logo and the login prompt reappear, the system is in multiuser mode.

Changing the /etc/profile File

The /etc/profile file is read when the users' terminals are brought up. The file is a shell script that sets up the environment for users. The /etc/profile file provided with the system software defines, for example, the PATH variable and the terminal type.

Use a text editor to make any changes or additions to the file. For example, if you want a message to appear at the users' terminals when they log in, add the following to the /etc/profile file:

```
echo message
```

Each user also has a .profile file in which the user can define his or her own environment (see your *CENTIX Operations Guide* for details.) The system reads the .profile files after the /etc/profile file. If there are any contradictions between the two files, the system uses what is defined in the .profile file.

For Mixed Systems Only: Changing the Time Zone Variable

When the CENTIX operating system is brought up, it reads the base system time to determine the current time, then adjusts that time as defined by the TZ (time zone) variable. If you have a mixed system, the base system time is BTOS time, which has already been adjusted for time zones. You, therefore, need to redefine the CENTIX TZ variable so that it does not adjust an already-adjusted time.

The TZ variable is set in three files: /etc/bcheckrc, /etc/rc, and /etc/profile. In each of these files, use a text editor to change the TZ variable line to:

```
TZ=GMT0; export TZ
```

With the time zone in CENTIX defined as GMT0 (Greenwich Mean Time), the CENTIX system does not adjust the BTOS time, and the times match.

Setting Up the /etc/inittabnn Files

Note: *The /etc/inittab*nn *files are installed with your system software. Unless you choose to add terminals to your system without using centrEASE (see Section 4 for details), you may never need to make any changes to these files. This section is for information only.*

The /etc/inittab*nn* files are text files that list the major processes that must be started on the Applications Processors (APs) each time the operating system enters a different operating level. There is a separate /etc/inittab*nn* file associated with each AP on the system. The *nn* is the number of the AP with which the file is associated.

The files are used by the init process. The init process looks for, and spawns (starts up), the processes listed in the /etc/inittab*nn* files for the current operating level. That is, if level 2 is the current level, the init process starts up all processes defined in the /etc/inittab*nn* files as level 2.

init Operating Levels

There are eight possible operating levels in this system: 0 through 6 and s. In the original /etc/inittab*nn* files released with your system software, only these levels are used:

- Level s is defined as single user mode. Only the console has access to the system.
- Level 2 is defined as multiuser mode. In multiuser mode, the user terminals are readied for login.
- Level 6 is used to provide an administrator login if the system does not come up properly (see "Starting Up the System Software" in Section 2 for details).

Note: *Level 5 is used by the* ***shutdown*** *program to kill all currently running processes. There are, therefore, no processes listed for level 5 in the /etc/inittab*nn *files that are provided with the system software.*

There are also three temporary levels, a, b, or c, that you can assign to processes. Like a process labeled with a normal level, a process labeled with a temporary level is started when init is invoked with that level as an argument. init, however, does not itself change levels. That is, if init is running in level 2, and "init a" is invoked, any entry with an "a" level is started up, but the level 2 processes are not terminated.

Any temporary processes are stopped when init enters single user mode. Any other change of level does not kill temporary processes. To stop a temporary process once it has been started up, you must remove the process entry from the /etc/inittab*nn* file, or change the *entry* type to off (see "Format of the /etc/inittab*nn* Entries," below).

Note: *The temporary levels are not used in the /etc/inittab*nn *files that are released with your system software.*

When init Scans /etc/inittabnn

The init process looks at the /etc/inittab*nn* files whenever one of these conditions occur:

- The system is booted.
- The **init** command is issued with a new level as an argument. For example, the following command causes the init process to look at the /etc/inittab*nn* files and spawn any level 2 processes:

```
init 2
```

  The **init** command can be issued by the administrator, but it is usually issued from a shell script or program. The **/etc/shutdown** shell script, for example, issues **init** at the end of its program to enter either the default mode (single user) or the operating mode selected by the administrator. (See Section 2 for details on **shutdown**.)

Warning: *Do not issue the* ***init*** *command for a level that is not assigned any processes in the /etc/inittab*nn *files. In particular, if you cause init to enter a level for which there is not a getty process for the console (see "/etc/getty," below for an explanation), you will lose access to your console.*

Note: *The* ***telinit*** *command can be used interchangeably with the* ***init*** *command.* ***init*** *resides in the /etc directory (****/etc/init****) and* ***telinit*** *resides in the /bin directory (****/bin/telinit****).*

- One of the processes that was spawned by init terminates, for example, when a user logs off a terminal. The init process is informed (by a system call) that one of its "children" has died. It looks again at the /etc/inittab*nn* files for further instructions.

  In the example of a terminated terminal process, the further instructions may be to "respawn", that is, to regenerate the terminal login prompt. (See "Format of the /etc/inittab*nn* Entries" for more details on respawn.)

Format of the /etc/inittabnn Entries

Entries in the /etc/inittab*nn* files have the format:

id:level:type:process

where:

- *id* is a unique one-to-four character identifier that init uses internally to label entries in its process table. It is also placed in the dynamic record file, /etc/utmp, and the log file for the login process, /etc/wtmp.
- *level* is one or more alphanumeric characters that specify the operating level or levels at which this process is to run. Whenever the init internal level matches a level specified by *level*, this entry is active. If the init internal level does not match any of the levels specified, init makes certain that the process is not running. If this field is empty, the process runs at all levels.
- *type* specifies some further condition required for the execution of an entry. The most commonly used *types* are as follows (for a listing of all *types*, see your *CENTIX Operations Reference Manual*):

  bootwait. Entries of the bootwait type have the execution behavior of wait entries and are only run when **init** switches to a numeric level for the first time.

initdefault. The initdefault entry is scanned when init is first invoked when the system is booted up. It specifies the level at which init should begin operation when it first comes up. If no initdefault entry exists, **init** asks at the system console, /dev/console, for the initial run state.

off. The entry is not to run even if the process *level* matches the current init level. To temporarily disable an entry, change its *type* to off.

respawn. The entry is to run as long as init is running in a level specified in the level field. Whenever init detects the termination of a process labeled respawn, it restarts the process.

wait. The process is started when init enters the same level as the process *level*. init then waits for the process to terminate. If init rereads the /etc/inittab file before changing to a new level, init will not rerun this process.

sysinit. The process is executed before init accesses the console.

- *process* is the shell command that is to be performed for the entry. Any shell command, using any standard command syntax, can be listed. A series of commands can be listed, with a semi-colon between each command.

Note: *A line in the /etc/inittab*nn *file that begins with a colon (:) has been "commented out" and will not be read. Commenting out a line is a way of temporarily deleting the line. The centrEASE program may comment out lines in the /etc/inittab*nn *files. Do not remove any colons without knowing why the line was commented out.*

The init Process: A Summary

The init process is started when the system is first booted up. When it is first started, init looks in the /etc/inittab*nn* files for an entry that has initdefault in its *type* field. The level of this entry is the level at which init begins operation.

init scans the /etc/inittab*nn* files for all entries at the level defined by initdefault. It handles each of the entries as defined by the *type* field in the entry. That is, if the entry is labeled respawn, for example, init starts it up. Or if the entry is labeled wait, init starts it up, and then waits until the process is terminated before going on to the next process.

After it has started up all of the processes for the initial level, init's task is done until it is called on to change levels. The operating level can be changed by the superuser from the console or through a shell script. For example, when you run the **shutdown** command, the **shutdown** shell script issues **init** at the end of its program to enter either the default mode (single user) or the operating mode selected by the administrator. (See Section 2 for details on **shutdown**.)

When init changes levels, it first scans the /etc/inittab*nn* files, looking for processes that should not be active at the new level. It terminates any of these that are active. init then rescans the /etc/inittab*nn* files, and starts up any processes that should be active at the new level.

init also rescans the /etc/inittab*nn* files when it receives a signal telling it that one of the processes that it has spawned has died. If the process that died, for example, was labeled wait, init can go on to the next process. If the process that died was labeled respawn, init restarts the process that died.

This sequence continues as long as the system is running. init runs quietly in the background, waiting to be told to wake up and read the /etc/inittab*nn* files.

If You Make Changes to /etc/inittabnn

When the system is first booted up, the /etc/inittab*nn* files are copied into memory. Whenever the init process looks at /etc/inittab*nn*, it looks at this copy of the original file.

When you make a change to one of the /etc/inittab*nn* files, you are changing the original, not the copy that was read into memory when the system was booted up. If you want init to use the changed /etc/inittab*nn* file, you must force the changed file to be loaded into memory. Rebooting the system is one way to do this. An easier way is to issue the **init** command with an argument. Enter:

```
# init q
```

The changed /etc/inittab*nn* file is copied into memory. The init process reads the new /etc/inittab*nn* file and makes any necessary changes to the current processes.

The Processes Started Up by /etc/init

The processes described in the following subsections are those listed in the original /etc/inittab*nn* files provided with your system software.

/etc/getty (/etc/inittab00, /etc/inittab01)

The /etc/inittab00 file contains an entry for each line on which users can log into the system, typically terminal lines. The process listed in the entry is the /etc/getty process. The getty process conditions the terminal connection for login. To get information on the terminals, the getty process reads the /etc/gettydefs file. Each entry in /etc/gettydefs specifies a set of communications options, including a baud rate, and a login message for a terminal.

The /etc/gettydefs File

Each entry in /etc/gettydefs is a line of the form:

label#initial-flags#final-flags#login-prompt#next-label

where:

- *label* identifies the entry. The *label* must be unique in the file. By convention, *label* for a multiuser mode entry is the baud rate and *label* for an administrator mode entry is the letter C followed by the baud rate.

- *initial-flags* is a list of communications options for getty to apply when it first opens the terminal. The first flag listed is always a capital B followed by the baud rate for the terminal when it is first opened (B*baudrate*). The following flags are bit options used in the termio general terminal interface program.

  There are dozens of possible flags. These are listed in Section 6 of your *CENTIX Operations Reference Manual* under termio. There are four categories of flags listed:

  - Flags that describe input modes are listed under the variable, *c_iflag*.
  - Flags that describe output modes are listed under the variable, *c_oflag*.
  - Flags that describe hardware control modes are listed under the variable *c_cflag*.
  - Flags that describe local terminal functions are listed under the variable, *c_lflag*.

  The flags are listed in uppercase letters and are separated from each other by spaces or tabs.

- *final-flags* is a list of communications options for **getty** to apply after the user has logged in. The first flag listed is always a capital B followed by the baud rate for the terminal after the user has logged in (B*baudrate*). The *final-flags* are taken from the same set described for *initial-flags*.

- *login-prompt* is the text printed to the terminal screen when the terminal is first opened. The text should end with the login prompt:

```
login:
```

If you want to change the *login-prompt*, use the following as control characters:

Use This	To Get This
/n	newline
/t	tab
/v	vertical tab (CODE-K on dumb terminal)
/b	backspace
/r	carriage return
/f	form feed
/*xxx*	*xxx* is a 1- to 3-digit octal number. Octal numbers can be used for special unprintable characters.

- *next* indicates the label of another entry to use if **getty** receives a break signal (a stream of null characters) while it is running this entry. This field is only appropriate if the line is being set up for a dial-up modem. It is **not** appropriate for a direct connection.

  If you are setting up a line for a dial-up modem, you may not know the baud rate of the terminal that is going to log on. Use this field to list one or more alternate baud rates. While the getty process is trying to log the terminal onto the system, the user can press the CANCEL key (or BREAK key, if there is one on the terminal keyboard) to cause the getty process to try the next baud rate listed in the *next* field. The user can continue to push the CANCEL (or BREAK) key until the correct baud rate is reached, and the getty process can successfully log the terminal onto the system.

Sample /etc/gettydefs File

The following is a sample /etc/gettydefs entry. It is appropriate for a direct-connect 9600 baud terminal.

```
9600# B9600 CLOCAL BRKINT IGNPAR ISTRIP IXON IXOFF ECHO
OPOST ONLCR # BRKINT ISTRIP ICRNL IXON OPOST ONLCR B9600 CS8
CREAD ISIG ICANON ECHO ECHOE ECHOK IXANY TAB3 #BURROUGHS
XE500 OPERATING SYSTEM UNIX B2.00.01\n\nlogin: #9600
```

Making Changes to /etc/gettydefs

You should not directly edit the /etc/gettydefs file. To make changes, follow these steps:

1 Use the **cp** command to make a copy of the /etc/gettydefs file. Enter:

```
# cp /etc/gettydefs newname
```

where *newname* is the name assigned to the copy of the /etc/gettydefs file.

2 Use a text editor to make your changes to the *newname* file.

3 Use the **getty** command with an argument to copy your changed file back to the /etc/gettydefs file. This command ensures that the changes are correctly syntaxed. Enter:

```
# getty -c newname
```

4 The getty process usually uses new entries in /etc/gettydefs without any prompting. However, if getty tries to use an obsolete /etc/gettydefs, the terminal can be tied up. If a terminal remains unusable after you have changed /etc/gettydefs, go to the next step.

5 Determine the process number of the **getty** monitoring the terminal. If you know the terminal number of the terminal that is tied up, enter at a terminal that is *not* tied up (if all terminals are tied up, go to step 7):

```
# ps -tnnn
```

where *nnn* is the terminal number. The system returns the process number.

If you do not know the terminal number, at another terminal, enter:

```
# ps -e
```

The system returns the process numbers of all currently running processes.

6 Terminate **getty**. Enter:

```
# kill n
```

where *n* is the process number of the terminal. If you used "ps -e", and do not know which is the correct process number, list all possible numbers:

```
# kill n n n n n
```

The init process, sensing that one of its processes has died, rereads the /etc/inittab file. The getty processes are respawned, using the information from the new /etc/gettydefs file.

7 (Use this step only if all terminals were tied up after step 3.) If another terminal is not available, follow these steps:

a Reboot the XE 500 in the restricted mode. See Section 15 for details.

b Mount the root file system. Enter:

```
# /etc/mount devicename
```

where *devicename* is the pathname of the memory device where the root file system is stored.

c Use a text editor to change the /etc/gettydefs file back to its original state. To do this, follow the procedure described in steps 1 through 3, above.

d Reboot the system in normal mode.

/etc/mkconrc (/etc/inittab00)

The /etc/mkconrc file searches the BTOS configuration files to determine the location of the first PT 1500 on the system and establishes it as the system console. See Section 4 for a discussion of PT 1500 numbering.

If /etc/mkconrc cannot find a PT 1500, the system console is established on the first asynchronous line in the system (usually tty000).

/etc/mkconrc creates /etc/conrc.

/etc/conrc (/etc/inittab00)

This file contains one line that defines which terminal is the system console. The line is:

```
/etc/console -t /dev/ttynnn
```

where *nnn* is the number of the terminal that is assigned as the system console. (See Section 4 for details on terminal numbering.) If you are moving your system console to, for example, a dumb serial terminal, you may have to change this file.

/etc/conrc01 (/etc/inittab01)

This file provides the same function for AP01 as /etc/conrc provides for AP00.

/etc/bcheckrc (/etc/inittab00)

This file runs the file system check program (**fsck**) on the file systems before they are mounted (see Section 4 for details on **fsck**).

/etc/brc (/etc/inittab00)

The /etc/brc process removes the mount table that was previously set up for the system.

/etc/rc (/etc/inittab00)

This file sets up the new mount table for the system. It is also used as a general purpose file for starting up background processes, such as the lp spooler scheduler and cron (see Sections 9 and 12 for details).

/etc/allrc (/etc/inittab00, /etc/inittab01)

On each AP, the /etc/allrc file sets the system node name for uucp (see Section 10). The file also starts up system activity (sa) monitoring on each AP (see Section 11).

Configuring I/O Devices

The I/O devices on the system must be configured into both the BTOS and CENTIX operating systems.

To be configured into the BTOS operating system, a device must be listed in the configuration file for the processor board that controls the device. BTOS processor configuration files are explained in detail in your *CENTIX Installation and Implementation Guide*.

To be configured into the CENTIX operating system, each I/O device must be represented in the overall CENTIX file system by a device—or special—file, located in the /dev directory. The contents of a device file point to the device driver, located in the kernel, for the device.

When you send data to, for example, a disk, you send the data to the device file in the /dev directory that you have created for that disk. The data, however, is not actually stored in the device file (in the CENTIX file system), but on the disk itself. In the same way, when you load data from a tape, you call it from the device file for the tape device, but the data is actually loaded from the tape itself.

There are two types of CENTIX device files:

- *Block* device files are used for devices that handle I/O data in 1024 bytes (1 kB) blocks. The I/O size is controlled by the operating system's buffer size and is independent of the user's I/O size. Disk and tape devices can be configured as block devices.
- *Character* device files are used for devices that handle raw data streams. The size of I/O transfers in raw data streams are determined either by the software design of the device itself (for terminals and printers) or by the program controlling the device (for disks and tapes).

For those devices that can be used as either block or character, the difference between the two is in performance. One or the other type of device may be necessary for special applications.

With the CENTIX 6.0 system software release, the naming conventions for device files for tapes and disks have changed. (Device names for printers and terminals have not changed.) The system now supports both the old and new naming conventions. Old names are linked to the new names internally. If you have installed CENTIX 6.0 as a system update to an earlier version of CENTIX, you do not have to change your device file names to meet the new convention. Both old and new naming conventions for special files are described in this section.

Configuring a Disk

Warning: *You can easily destroy other disks while trying to configure a new disk. If you are not completely confident with the procedures described in this subsection, you should configure your disks through centrEASE. See your* centrEASE Operations Reference Manual.

Note: *An XE 500 disk can be a disk cartridge, a built-in disk, or a Storage Module Device (SMD) disk.*

In this guide, a "built-in disk" is assumed to be a 5 1/4-inch hard disk controlled by a File Processor (FP). In some base enclosure styles, SMD disks can also be built into the XE 500; however, they will be referenced as "SMDs." All SMDs, whether they are in an XE 500 base enclosure or an MD3 enclosure, are controlled by Disk Processors (DPs).

After a new hard disk has been installed, you must configure the disk in both the BTOS and CENTIX operating systems. Follow the steps described in the following paragraphs.

Step 1: Determining the Disk Name

In the BTOS operating system, disks are given both disk drive and volume names. The disk drive name refers to the hardware device. The volume name is assigned to the disk when it is initialized (see Section 14). In the CENTIX portion of the operating system, disk devices are given file names in the /dev directory.

BTOS Disk Device Names

XE 500 disk devices are named according to their physical location in the system. The naming conventions are different for disks controlled by FPs (the disk cartridge drive and built-in disk drives) and for disks controlled by DPs (SMDs in an XE 500 base enclosure or an MD3 enclosure).

Disks in the first enclosure that are controlled by an FP are named d0 through d3; those in a second enclosure are named d4 through d7; and so on.

SMD disks in a base enclosure, which are controlled by the first DP in the system (DP00), are named s0 and s1. Because each DP can control up to six SMDs, DP00 can also control MD3 SMDs, up through s5. SMDs in MD3 enclosures that are controlled by the second DP in the system (DP01) are named s6 through s11; and so on.

Figures in Appendix C show the various disk naming schemes.

CENTIX Disk Device File Names

In CENTIX systems **before the 6.0 release**, the disk devices are named as follows:

`/dev/[r]xpddn`

where:

- [r] is an optional field that defines the disk as a character—rather than block—device. See the beginning of this section for definitions of character and block devices.
- *xp* is fp if the disk device is connected to an FP or dp if the disk device is connected to a DP.

- *dd* represents the disk number. CENTIX disk numbers are the same as the BTOS disk device numbers, except that you must add a 0 in front of a one-digit BTOS disk number for CENTIX. That is, if a built-in disk is named d4 in BTOS, *dd* is 04 in CENTIX. Or, if an SMD disk is named s1 in BTOS, *dd* is 01 in CENTIX. (Do *not* add a zero in front of a two-digit BTOS disk number. For BTOS disk s10, *dd* is 10.) See "BTOS Disk Device Names" earlier in this section for details on numbering the disks in BTOS.
- *n* represents the disk partition. Each disk has a maximum of eight partitions (0 through 7).

With the **6.0 release**, the disk devices on your system are named as follows:

```
/dev/[r]dsk/cndnnsn
```

where:

- [r] is an optional field that defines the disk as a character—rather than block—device. See the beginning of this section for definitions of character and block devices.
- c*n* represents the controller number. The controller number is always c0 if the controller is an FP. The controller number is always c1 if the controller is a DP.
- d*nn* represents the disk number. CENTIX disk numbers are the same as the BTOS disk device numbers, except that you must add a 0 in front of a one-digit BTOS disk number for CENTIX. That is, if a built-in disk is named d4 in BTOS, *nn* is 04 in CENTIX. Or, if an SMD disk is named s1 in BTOS, *nn* is 01 in CENTIX. (Do *not* add a zero in front of a two-digit BTOS disk number. For BTOS disk s10, *nn* is 10.) See "BTOS Disk Device Names" earlier in this section for details on numbering the disks in BTOS.
- s*n* represents the disk partition. Each disk has a maximum of eight partitions (0 through 7).

Table 4-1 shows the correlation between the old (pre-6.0 release) and new (6.0 release) naming conventions for built-in disks connected to FPs. Table 4-2 shows the correlation between the old and new naming conventions for storage module device (SMD) drives connected to DPs.

Note: *In the disk names in Table 4-1,* n *represents the partition number. Each disk can have up to eight partitions (0 through 7).*

Table 4-1 **Naming Conventions for Built-In Disk Drives.**

Pre-6.0 Release	6.0 Release and Later	BTOS Disk Device Name
FIRST FP		
/dev/[r]fp00*n*	/dev/[r]dsk/c0d00s*n*	d0 (disk cartridge)
/dev/[r]fp01*n*	/dev/[r]dsk/c0d01s*n*	d1
/dev/[r]fp02*n*	/dev/[r]dsk/c0d02s*n*	d2
/dev/[r]fp03*n*	/dev/[r]dsk/c0d03s*n*	d3
SECOND FP		
/dev/[r]fp04*n*	/dev/[r]dsk/c0d04s*n*	d4
/dev/[r]fp05*n*	/dev/[r]dsk/c0d05s*n*	d5
/dev/[r]fp06*n*	/dev/[r]dsk/c0d06s*n*	d6
/dev/[r]fp07*n*	/dev/[r]dsk/c0d07s*n*	d7
THIRD FP		
/dev/[r]fp08*n*	/dev/[r]dsk/c0d08s*n*	d8
/dev/[r]fp09*n*	/dev/[r]dsk/c0d09s*n*	d9
/dev/[r]fp10*n*	/dev/[r]dsk/c0d10s*n*	d10
/dev/[r]fp11*n*	/dev/[r]dsk/c0d11s*n*	d11
and so on		

Note: *In the disk names in Table 4-2,* n *represents the partition number. Each disk can have up to eight partitions (0 through 7).*

Table 4-2 **Naming Conventions for SMD Disk Drives.**

Pre-6.0 Release	6.0 Release and Later	BTOS Disk Device Name
FIRST DP		
/dev/[r]dp00*n*	/dev/[r]dsk/c1d00s*n*	s0
/dev/[r]dp01*n*	/dev/[r]dsk/c1d01s*n*	s1
/dev/[r]dp02*n*	/dev/[r]dsk/c1d02s*n*	s2
/dev/[r]dp03*n*	/dev/[r]dsk/c1d03s*n*	s3
/dev/[r]dp04*n*	/dev/[r]dsk/c1d04s*n*	s4
/dev/[r]dp05*n*	/dev/[r]dsk/c1d05s*n*	s5
SECOND DP		
/dev/[r]dp06*n*	/dev/[r]dsk/c1d06s*n*	s6
/dev/[r]dp07*n*	/dev/[r]dsk/c1d07s*n*	s7
/dev/[r]dp08*n*	/dev/[r]dsk/c1d08s*n*	s8
/dev/[r]dp09*n*	/dev/[r]dsk/c1d09s*n*	s9
/dev/[r]dp10*n*	/dev/[r]dsk/c1d10s*n*	s10
/dev/[r]dp11*n*	/dev/[r]dsk/c1d11s*n*	s11
THIRD DP		
/dev/[r]dp12*n*	/dev/[r]dsk/c1d12s*n*	s12
/dev/[r]dp13*n*	/dev/[r]dsk/c1d13s*n*	s13
/dev/[r]dp14*n*	/dev/[r]dsk/c1d14s*n*	s14
/dev/[r]dp15*n*	/dev/[r]dsk/c1d15s*n*	s15
/dev/[r]dp16*n*	/dev/[r]dsk/c1d16s*n*	s16
/dev/[r]dp17*n*	/dev/[r]dsk/c1d17s*n*	s17
and so on		

Step 2: Initializing the Disk

Before a disk can accept data, you must format—or initialize—it with the BTOS **MIVolume** utility. Once a disk drive has been initialized, it is said to contain a volume. The term *disk drive* refers to the hardware device; *volume* refers to the complete BTOS file system unit of information stored on the disk. Each formatted disk in the system has a volume associated with it.

If the disk has not been initialized (or if you want to wipe out what is stored on the disk), use the BTOS **MIVolume** utility to initialize the disk. See Section 14 of this guide for details on when and how to initialize a disk.

Step 3: Creating Partitions on the Disk

After initializing the disk, you divide the disk into logical pieces called BTOS partitions. In step 4 (see the next subsection), you will create for each partition a CENTIX special device file. To the BTOS operating system, each partition used for CENTIX is simply one large file. To the CENTIX operating system, each partition looks like a separate storage device.

When you request an I/O operation, BTOS finds the correct partition on the disk, then CENTIX takes over to find the correct file on the partition.

A disk can hold from one to eight partitions. Each one of these partitions can hold one and only one CENTIX file system (only one file system can be assigned to each CENTIX device file). Determine the size of the file systems that will be stored in the partitions before deciding how to divide a disk. If, for example, you are going to store two very large file systems on a disk, you can make two very large partitions.

You do not have to partition the whole disk at one time. You can create one or more partitions at first, store file systems in them, work for a while, then add more partitions and file systems later (unless you used up all of the disk space on the first partitions).

Note: *If you decide that you want only one partition on a disk, you must still create that one partition. To store a file system on a disk, it must be assigned to a partition.*

To create a partition on a disk, use the **crup** command. Enter:

```
# crup '[BTOS device name]<sys>partition.n' blocks
```

where:

- *BTOS device name* is the name of the BTOS disk on which you are creating the partition. If the device is a built-in disk, the device name is d*n*, where *n* is the device number. If the device is an SMD disk, the device name is s*n*, where *n* is the disk number. See "BTOS Disk Device Names," earlier in this section.
- *n* is the number of the partition that you are creating. The eight partitions on a disk are numbered 0 through 7.
- *blocks* is the number of 512-byte blocks that you want in the partition. The partition should be the size of the file system that you are going to store in the partition. *blocks* must be an even number.

The following are examples of the **crup** command:

```
# crup '[d1]<sys>partition.0' 6000

# crup '[s2]<sys>partition.2' 20000
```

Note: *Make sure that you include the single quotation marks shown in the* ***crup*** *command line.*

If you need to determine the names and sizes of the partitions that have already been created on a disk, use the **ofls** command. Enter:

```
# ofls -l '[BTOS device name]<sys>partition.*'
```

Any partitions already created on the BTOS device that you specify are listed on the screen. The partition size in bytes, the number of 512-byte sectors (blocks) in the partition, and the last time the partition was modified are also listed.

Step 4: Creating the CENTIX Device Files

After you define the BTOS partition on the disk, you create the CENTIX special device file that represents that partition in the CENTIX file system. For each partition, create two device files, one making the partition a block device and one making the partition a character device. Use the CENTIX **mknod** command. Enter (the first line creates a block device; the second line a character device):

```
# mknod devicefilename b 0 minornumber
# mknod devicefilename c 1 minornumber
```

where:

- *devicefilename* is the name of the partition device file, in the form /dev/[r]dsk/c*n*d*nn*s*n*. See "CENTIX Disk Device File Names" for details.
- *minornumber* is an identifying number for the partition.
  - To derive the number for a partition on an fp, use this equation:

    partition number + (8 x disk device number)

    For example, for /dev/dsk/c0d02s7, the *minornumber* is:

    7 + (8 x 2) = 23
  - To derive the number for a partition on a dp, use this equation:

    partition number + (8 x disk device number) + 128

    For example, for /dev/dsk/c1d00s2, the *minornumber* is:

    2 + (8 x 0) + 128 = 130

Note: *These* minornumbers *are by convention only. You can assign any* minornumber *as long as it is unique to that disk device, and you enter the same number in the ConfigUFS.sys file (see Step 5).*

The following are examples of **mknod** command lines. The first two lines and the last two lines are each defining one partition as both block and character.

```
# mknod /dev/dsk/c0d01s3 b 0 11
# mknod /dev/rdsk/c0d01s3 c 1 11
# mknod /dev/dsk/c1d00s0 b 0 128
# mknod /dev/rdsk/c1d00s0 c 1 128
```

Step 5: Identifying the Partitions to the BTOS Operating System

The BTOS operating system uses several text files to define the systems hardware configuration. These text files are read when the system is booted. One of these files, [sys]<sys>ConfigUFS.sys, lists the CENTIX disk partitions. When you create a partition on a disk, you must list it in this file, or the BTOS operating system will not recognize it.

Note: *The default [sys]<sys>ConfigUFS.sys that was provided with your system software may already have the new partition listed. Check before you add anything to the file.*

Use one of the CENTIX/BTOS text editors (**ofvi** or **ofed**; see your *CENTIX Operations Reference Manual*) to enter [sys]<sys>ConfigUFS.sys. Add a line to the file for the partition that you are adding, in the form:

```
[BTOS device name]<sys>partition.n        XP0n    minornumber
```

where:

- *[BTOS device name]*<sys>partition.*n* is how you named the partition in the **crup** command (see step 3).
- *XP0n* is the processor board (FP00, FP01, DP00, DP01, and so on) on which the [sys]<sys>UFS.run file that is controlling the partition is running. (The [sys]<sys>UFS.run file is the CENTIX file system server. See the *CENTIX Installation and Implementation Guide* for details.)
- *minornumber* is the number you assigned the partition's device file in the **mknod** command (see step 4).

Note: *The spaces between entries in the file must be tabs.*

Step 6: Rebooting the System

You must reboot the system to force the BTOS operating system to read the changed ConfigUFS.sys file.

To reboot the system, see Section 2 for directions on shutting down CENTIX. When CENTIX has entered single user mode, turn the front panel keyswitch to STOP, then back to NORMAL. When the front panel display shows a "20," you can bring CENTIX back up, as described in Section 2.

Step 7: Creating File Systems on the Partitions

To make a partition available for use, you create a file system and assign it to that partition. Data can then be stored in the file system. Creating file systems is described in Section 5.

Configuring a Tape Drive

After a new tape drive has been installed, you must configure the drive in both the BTOS and CENTIX operating systems. Follow the steps described in the following paragraphs.

Step 1: Determining the Tape Drive Name

BTOS Tape Drive Names

BTOS QIC Drive Name. The QIC tape drive is named:

qic

BTOS Half-Inch Drive Names. In the BTOS operating system, the half-inch tape drives are named:

tape*xy*

where:

- *x* is a value from 0 to 3 that represents the SP or DP to which the half-inch tape drive is connected. To determine *x*, look at the back of the XE 500 and identify the SP or DP to which the tape drive is connected. See Appendix C for an illustration of SP and DP numbering.

  The default for *x* is 0. This default refers to the first SP or DP on the system, **not** the SP or DP on which the first tape server is running.

- *y* is the specific half-inch tape drive. Because two half-inch tape drives can be daisy-chained together from a single SP or DP, *y* can be 0 or 1. See Appendix C for an illustration of how tape drives are numbered.

Note: *If only one digit is given in the tape drive name (for example, tape2), the digit is assumed to be the* y *value. The default value of 0 is used for the* x *value.*

CENTIX Tape Drive File Names

Table 4-3 shows the correlation between the old and new naming conventions for tape devices.

In CENTIX systems **before the 6.0 release**, the tape drives are named as follows:

/dev/[n][r]mt*n*

where:

- [n] indicates that the tape is not to rewind when a tape file closes. The default is that the tape automatically rewinds.
- [r] is an optional field that defines the tape device as a character—rather than block—device. See the beginning of this section for definitions of character and block devices.
- *n* represents the tape drive in the system. *n* is 0 for the first half-inch tape drive on the system, 1 for a QIC tape drive, 2 for the second half-inch tape drive on the system, 3 for the third, and so on.

With the **6.0 release**, The tape drives on your system are named as follows:

```
/dev/[r]mt/cndn[n]
```

where:

- [r] is an optional field that defines the tape device as a character—rather than block—device. See the beginning of this section for definitions of character and block devices.
- c*n* represents the controller number. For a QIC tape drive, c*n* is always c0. For a half-inch tape drive, c*n* is always c1.
- d*n* represents the tape drive on the controller. You can have only one QIC drive on your system; it is d0. The first half-inch tape drive is d0, the second is d1, and so on.
- [n] indicates that the tape is not to rewind when a tape file closes. The default is that the tape automatically rewinds.

Table 4-3 **Naming Conventions for Tape Drives.**

	Pre-6.0 Release	6.0 Release and Later
first half-inch drive	/dev/[n][r]mt0	/dev/[r]mt/c1d0[n]
QIC drive	/dev/[n][r]mt1	/dev/[r]mt/c0d0[n]
second half-inch drive	/dev/[n][r]mt2	/dev/[r]mt/c1d1[n]
third half-inch drive	/dev/[n][r]mt3	/dev/[r]mt/c1d2[n]
and so on		

Step 2: Creating the CENTIX Device File

The CENTIX device file represents the tape drive in the CENTIX file system. For each partition, create two device files, one making the partition a block device and one making the partition a character device. To create the device files, use the **mknod** command. Enter (the first line creates a block device; the second line a character device):

```
# mknod devicefilename b 2 minornumber
# mknod devicefilename c 4 minornumber
```

where:

- *devicefilename* is the name of the tape device file, in the form /dev/[r]mt/c*n*d*n*[n]. See step 1 for details.
- *minornumber* is an identifying number for the tape drive. *minornumber* is assigned as follows:
  - If the tape drive is automatic rewind (without the "n" in the device name) the *minornumber* is 0 for the first half-inch tape drive, 1 for the QIC tape drive, 2 for the second half-inch tape drive, 3 for the third half-inch tape drive, and so on.
  - If the tape drive is not automatic rewind, add 4 to the above numbers: 4 for the first half-inch tape drive, 5 for the QIC tape drive, 6 for the second half-inch tape drive, 7 for the third half-inch tape drive, and so on.

Note: *These* minornumbers *are by convention only. You can assign any* minornumber *as long as it is unique to that tape device, and you enter the same number in the ConfigUFS.sys file (see Step 3).*

The following are examples of the **mknod** command. These four lines configure a second half-inch tape drive for (in order): a block device without automatic rewind, a block device with automatic rewind, a character device without automatic rewind, and a character device with automatic rewind.

```
# mknod /dev/mt/c1d1n b 2 6

# mknod /dev/mt/c1d1 b 2 2

# mknod /dev/rmt/c1d1n c 4 6

# mknod /dev/rmt/c1d1 c 4 2
```

Step 3: Identifying the Special File to the BTOS Operating System

The BTOS operating system uses several text files to define the system's hardware configuration. These text files are read when the system is booted. One of these files, [sys]<sys>ConfigUFS.sys, lists the CENTIX disk partitions and tape devices. When you add a tape drive to the system, you must list it in this file, or the BTOS operating system will not recognize it.

Note: *The default [sys]<sys>ConfigUFS.sys that was provided with your system software may already have the new tape drive listed. Check before you add anything to the file.*

Use one of the CENTIX/BTOS text editors (**ofvi** or **ofed**; see the *CENTIX Operations Reference Manual*) to enter [sys]<sys>ConfigUFS.sys. Add a line to the file for the partition that you are adding, in the form:

```
Tape      BTOS device name                    minornumber
```

where:

- *BTOS device name* is [tapexy] for a half-inch tape drive or [qic] for a QIC tape drive (see step 1).
- *minornumber* is the number you assigned the tape drive's device file in the **mknod** command (see step 2).

Note: *The spaces between entries in the file must be tabs.*

Step 4: Rebooting the System

You must reboot the system to force the BTOS operating system to read the changed ConfigUFS.sys file.

To reboot the system, see Section 2 for directions on shutting down CENTIX. When CENTIX has entered single user mode, turn the front panel keyswitch to STOP, then back to NORMAL. When the front panel display shows a "20," you can bring CENTIX back up, as described in Section 2.

Configuring a Terminal

Terminal Device File Names

Note: *Terminals are not given BTOS names. BTOS recognizes terminals by the port to which they are connected (see "How CP and TP Ports are Numbered," below).*

CENTIX terminal device files are named as follows:

```
/dev/ttynnn
```

where *nnn* is the number assigned by the system to the terminal. The tty numbering is explained in detail in the following paragraphs.

How CP and TP Ports are Numbered

All serial RS-232-C devices—serial printers, modems, remote PT 1500s, and dumb serial terminals—are connected to RS-232-C ports on CP and TP boards. (An exception is a serial printer connected directly to a PT 1500; see Section 9.) PT 1500s are connected to RS-422 ports on CP boards. The ports—both RS-232-C and RS-422— are assigned three-digit numbers based on two things:

- The physical position in the enclosure of the CP or TP on which the port is mounted.
- The number of RS-232-C serial devices and the number of PT 1500s listed in the configuration files of the system's processor boards.

Each CP can support three RS-232-C serial devices and 16 PT 1500s. Each TP can support ten RS-232-C serial devices. The ports to which the devices connect are assigned tty numbers sequentially from 00, starting at the CP or TP closest to the FP in the enclosure.

To determine the tty number of a port, count how many serial devices and PT 1500s are connected between the port and the start of the numbering sequence (the CP or TP closest to the FP). To do this, look at the BTOS configuration file for each CP and TP on the system. (Processor configuration files are named [sys]<sys>*Xpnn*.cnf, where *Xpnn* is the processor designation, for example, Tp00 or Cp01.) These files show how many RS-232-C serial devices are configured for each CP or TP, and how many PT 1500s a CP is supporting.

Start at the CP or TP closest to the FP and count until you reach the port:

- For each CP, the number of serial devices (maximum of 3) plus the number of PT 1500s (maximum of 16) connected to the board.
- For each TP, the number of serial devices connected to the board (maximum of 10).

When you add together the results from the CPs and TPs, the next number, minus 1, is the tty number for the port. (You have to subtract 1 because the tty numbering starts at 00.)

Note: *The tty numbers are dependent on how many serial devices and PT 1500s are* currently *configured into the system. If, for example, you remove a PT 1500 with a tty number of 12 from the first CP board and make the appropriate configuration file modifications, the devices with tty numbers of 13 and up will change.*

How PT 1500 Terminals are Dynamically Assigned tty Numbers

The tty numbers for PT 1500s are dynamically assigned by the system as the PT 1500s are turned on. The first PT 1500 turned on when the system is booted up is assigned the first port number to which any PT 1500 is connected. That is, for example, the following can be connected to the first CP on your system:

- a dumb serial terminal on port 000
- a serial printer on port 001
- five PT 1500s on ports 002 through 006

The first PT 1500 that the system recognizes when it is brought up is given the lowest available tty number, 002, regardless of which port the PT 1500 is actually connected to. The second PT 1500 turned on is given the next available number, 003, and so on. Note that each time a PT 1500 is turned on, it can be assigned a different tty number. To determine the current tty number of a PT 1500, enter at the PT 1500:

```
$ who
```

The system returns the current tty numbers for each login name at each PT 1500 that is turned on.

Adding a Terminal to the System

Note: *You can add a terminal to the system through centrEASE. See your* centrEASE Operations Reference Manual *for details.*

To add a terminal, follow these steps:

1 Add an entry for the new terminal into the BTOS configuration file for the CP or TP to which the terminal is connected. Processor configuration files are named [sys]<sys>*Xpnn*.cnf, where *Xpnn* is the processor designation, for example, Tp00 or Cp01. Use a CENTIX/BTOS editor (**ofvi** or **ofed**; see the *CENTIX Operations Reference Manual*) to make these changes to the appropriate configuration file :

   a If you are adding a dumb serial terminal, add an [async] entry to the configuration file. You can copy one of the [async] entries already in the file and change the *channel* number to correspond to the terminal's port number. See your *Installation and Implementation Guide* for details.

   b If you are adding a PT 1500, change the last line of the CP configuration file to show how many PT 1500s are connected to the CP. The line takes the form:

   ```
   pt n
   ```

   where *n* is the number of PT 1500s. Change *n* to the new number of PT 1500s connected to the board (maximum *n* is 16).

   See your *Installaticn and Implementation Guide* for details.

2 You must reboot the system to force the BTOS operating system to read the changed configuration files.

To reboot the system, see Section 2 for directions on shutting down CENTIX. When CENTIX has entered single user mode, turn the front panel keyswitch to STOP, then back to NORMAL. When the front panel display shows a "20," you can bring CENTIX back up, as described in Section 2.

3 Create the CENTIX device file that represents the terminal in the CENTIX file system. Enter:

```
# mknod /dev/ttynnn c 0 minornumber
```

where:

- *nnn* is the number of the port to which the terminal is connected (see "How CP and TP Ports are Numbered," above). Note that because PT 1500 numbers are dynamically assigned, this may not be the number that the system assigns to the PT 1500 when it is booted up (see "How PT 1500 Terminals are Dynamically Assigned tty Numbers").
- *minornumber* is the same as *nnn*, but without any leading zeros. For example, if *nnn* is 003, *minornumber* is 3.

4 Add an entry into the /etc/inittab00 file for the terminal. The entry has the form:

```
nnn:2:respawn:/etc/getty ttynnn 9600
```

where *nnn* is the number that you assigned in the **mknod** command.

If you want the line to be able to receive an administrative console login in case of a system failure (see Section 2), add this line also:

```
Cnnn:6:respawn:/etc/getty ttynnn C9600
```

where *nnn* is the number that you assigned in the **mknod** command.

See Section 3, "Setting Up the inittab File," for more details on this file.

5 Changes that you make to the /etc/inittab file are not put into effect until the init process rereads the /etc/inittab file. To force init to read the file, enter:

```
# init q
```

6 If you are adding a new type of terminal to the system, you may have to add entries into the /etc/gettydefs and /usr/lib/terminfo files. See "Adding a New Terminal Type to the System."

Adding a New Terminal Type to the System

Note: *You cannot add a new terminal type to the system through centrEASE. It must be done as described in this subsection.*

When the system software was installed, your system was configured to accomodate a basic set of terminal types. If you are installing a terminal type not already supported in the system, you must add information to the /etc/gettydefs file and the /usr/lib/terminfo directory. For information on modifying the /etc/gettydefs file, see Section 3.

You must have a file that describes the functional characteristics of the terminal in the /usr/lib/terminfo directory. The directory is organized into many subdirectories, one subdirectory for each letter of the alphabet and one for each number from one through nine. Files describing the terminals are stored in the subdirectory named with the first character of the terminal name. For example, the file describing the PT 1500 is stored in the subdirectory named p.

The /usr/lib/terminfo directory provided with your system software already contains files for dozens of terminals. Make sure that the terminal type that you are adding is not already described in the directory before you try to create or modify any files. If you must create or modify files in the /usr/lib/terminfo file, use the **tic** command. For details on **tic** and terminfo, see your *CENTIX Operations Reference Manual.*

Note: *In the software that is provided with your system, there is no subdirectory for the number six in the /usr/lib/terminfo directory. This is because there are no terminals listed that begin with a 6. If you use the* ***tic*** *command to add a terminal that starts with a 6, the subdirectory will automatically be created for you.*

Configuring the Console

There must be a console configured for CENTIX. The administrator interacts with the system through the console. The console is used by CENTIX only. Although the terminal—PT 1500 or dumb serial—being used by the console must be configured into BTOS, the BTOS operating system does not need to know that it is different than the other terminals on the system.

If you have more than one AP on your system, you can have a console defined for each AP. The console for a specific AP receives the error messages associated with that AP. See "Creating a Console for an Additional AP," below.

The CENTIX device file for the consoles is named:

```
/dev/console
```

Note: *One /dev/console file acts for as many consoles as you have on the system.*

This file is provided for you when your system software is installed.

The system console for the first AP is automatically defined when the system is first booted up. The /etc/mkconrc file (see Section 3) searches the BTOS configuration files to determine the location of the first PT 1500 on the system and establishes it as the system console. If /etc/mkconrc cannot find a PT 1500, the system console is established on the first asynchronous line in the system (usually tty000).

After determining the system console, /etc/mkconrc creates the file /etc/conrc and puts this line into it:

```
/etc/console -t /dev/ttynnn
```

where *nnn* is the number of the terminal assigned as console.

If, for example, you have a dumb serial terminal connected to port 000, and the console is assigned to tty000, the dumb terminal acts as the console. Or, for example, if you have PT 1500s connected to ports 002 through 006, and the console is assigned to tty002, the first PT 1500 that the system recognizes at boot time acts as the console.

Creating a Console for an Additional AP

If you add an AP to your system and want to assign it a console, do the following:

1 If it has not already been done, change the name of /etc/conrc to /etc/conrc00. Enter:

```
# mv /etc/conrc /etc/conrc00
```

2 Use a text editor to rename /etc/conrc to /etc/conrc00 in the /etc/inittab00 file.

3 Use a text editor to create the /etc/conrc*nn* file for the new AP, where *nn* is the number of the AP (01 for AP01, 02 for AP02, and so on).

4 Insert this line into the new /etc/conrc*nn* file:

```
/etc/console -t /dev/ttynnn
```

where *nnn* is the number of the terminal assigned as console.

5 To ensure that **init** reads the changed files, enter:

```
# init q
```

Changing the System Console to an RS-232-C Terminal

If you have an RS-232-C serial ASCII terminal, you can connect it to the first CP (CP00) and use it as the CENTIX system console instead of a PT 1500. To do this, use the following procedure:

1 Using the Reconfiguring the System function of centrEASE, add a terminal to channel 1 of CP00.

centrEASE adds an async entry for the terminal to the CP00 configuration file. It also reassigns tty numbers so that the terminal is tty000, the serial printer connected to channel 3 is tty001, and PT 1500 tty numbers start at tty002.

centrEASE also modifies the /etc/conrc file to define tty002 as the new console, and relinks the devices /dev/systty and /dev/syscon to tty002.

2 Using an editor, modify the line in /etc/conrc to read

```
console -t /dev/tty000
```

3 Relink the devices /dev/systty and /dev/syscon to tty000 by entering the following commands:

```
# rm /dev/syscon
# rm /dev/systty
# ln /dev/tty000 /dev/syscon
# ln /dev/tty000 /dev/systty
```

4 Using an editor, add the following to the end of the terminal's async entry in the [sys]<sys>Cp00.cnf file:

```
,flowgen,flowact
```

5 Reboot the system to make these changes take effect.

Configuring Printer Spoolers

There are two separate printer spoolers in the XE 500 CENTIX system: the lpr and lp printer spoolers.

The lpr printer spooler is BTOS-based. Although you can use a CENTIX command, **lpr**, to print to an lpr printer, there is no CENTIX device file connected with the lpr printers. Configuring the lpr printer spooler is described in your *CENTIX Installation and Implementation Guide*.

The lp printer spooler is CENTIX-based. The device file for an lp parallel printer is named:

```
/dev/lpnn
```

where *nn* is a unique number based on the physical position in the enclosure of the processor board to which the printer is attached. (See Section 9 for details.)

Device files for lp serial printers are named:

```
/dev/ttynnn
```

where *nnn* is the number assigned to the port to which the terminal is connected. See "How CP and TP Ports are Numbered," above.

You can set up a terminal as an lp printing device. The device name for such a terminal is the same as for a regular terminal.

You can also connect a serial printer directly to a PT 1500. Device files for these printers are named:

```
/dev/tp1nnn
```

where *nnn* is equal to the tty number currently assigned to the PT 1500 to which the printer is connected.

Configuring the lp printer spooler is described in detail in Section 9 of this guide.

Managing File Systems

Note: *The file system management tasks can also be performed through centrEASE. See your* centrEASE Operations Reference Manual *for details.*

The term "file system" is used in two ways in the CENTIX operating system. It is used when discussing a specific file system, that is, a collection of files stored on a defined physical memory device. It is also used, as in "the CENTIX file system," to describe the entire hierarchy of directories, specific file systems, and files in a CENTIX system.

This section describes the tasks needed to manage specific file systems. The internal structure of a file system is described first.

Internal Structure of a File System

A file system is a collection of files stored on a defined physical memory device. Depending on both the size of the file system and the size of the memory device, a file system can take up a whole memory device, or can share the device with other file systems.

CENTIX divides each memory device into 1024-byte portions called blocks. Each file system contains the following blocks (see Figure 5-1 for an illustration):

- The boot block, which resides in block zero of the file system. On this system, the boot block is unused.
- The superblock, which resides in block one of the file system. The superblock contains descriptions of the file system, including the file system name, its size in blocks, the number of blocks reserved for inodes, the free inode list (see below for a definition of inode), and the free block list (the list of blocks still empty).

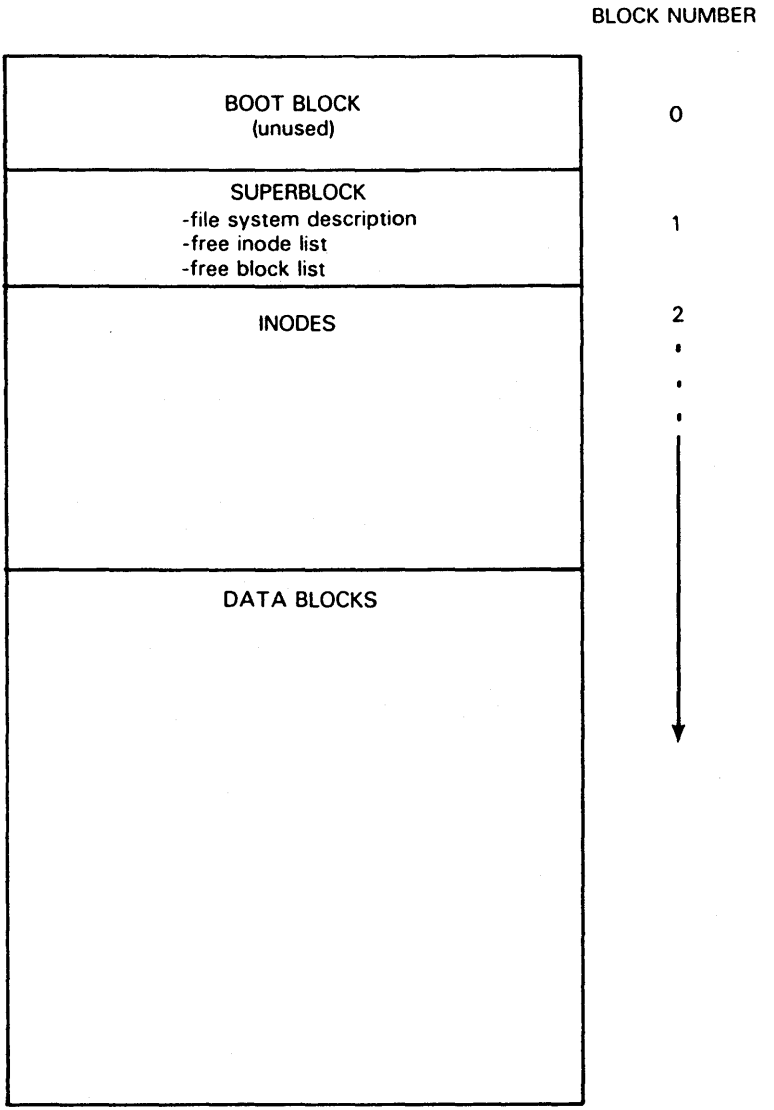
- The i-list, which contains the blocks that hold the records that describe CENTIX files, called inodes. There is one inode for each file and directory in the file system. Each inode has an i-number that gives the inode a place in the i-list. Each inode contains status information for its file or directory, such as the size, its owner and permissions, and whether it is a directory, an ordinary file, or a special file.

  The inode also contains a disk address list, which is a list of 13 block numbers. The first ten numbers, called direct pointers, are the numbers of the first ten blocks of the file. If the file is larger than ten blocks, the eleventh number, called an indirect pointer, is the number of a block that contains up to 256 more block numbers. If the file is larger than 266 blocks, the twelfth and thirteenth numbers in the disk address list point to blocks that can each contain up to 256 more block numbers.

  Also contained in an inode is the link count. The link count is an integer value that represents the number of directory entries linked to the inode. The link count is 0 when the inode is not in use. Creating a file sets the link count to 1. Each additional directory entry (link) for the file increases the link count by one. Each removal of a directory entry decreases the link count by one. If the link count returns to 0, the file blocks return to the free list.

- The data blocks, which contain the data stored in the files and directories. Large data files have data blocks that contain pointers to other blocks that hold the data.

Figure 5-1 **Internal Structure of a File System.**



Creating a File System

There are several steps needed to create a file system and make it available to the user. These steps are described in the following subsections.

Making the File System

Use the **/etc/mkfs** command to create a file system. Enter:

```
# /etc/mkfs /dev/filename [-P cluster] size[:inodes]
[gap/cylinder]
```

where:

- *filename* is the name of the file in the /dev directory associated with the memory device on which you are putting the file system. For example, if you are creating a file system on partition 6 of hard disk 0, /dev/*filename* is /dev/rdsk/c0d00s6. See Section 4 of this guide for a discussion on disk device file names.
- -P *cluster* is a Pilf factor value. You can use the Pilf factor to specify the size of blocks of data that will be moved in and out of the file system in I/O operations. See "Making a File System with a Pilf Factor," below.
- *size* is the number of 512-byte blocks contained in the partition in which you are storing the file system (as defined in *filename*, above) The size of a partition is defined when you create the partition with the **crup** command. See Section 4 for details on creating partitions.
- *inodes* is the number of inodes in the file system. Note that there cannot be more files than inodes in a file system. If you leave this field blank, the system provides a default value of 8.
- *gap* is an optional field that you can use to control the order in which blocks are accessed on a cylinder. The *gap* value is a single integer that determines how many blocks on the disk that the head skips while data is being written to memory.

EXAMPLE: CORRECT

mkfs /dev/rdsk/c0d03s0 75000 -1

FOR D3, PARTITION. 0

That is, after block 0 has been read from disk, that data is written to a buffer. Writing to the buffer takes a certain amount of time, during which the disk continues to spin, and no data can be read from disk. After the block 0 data has been written to buffer, block 1 must be read. If block 1 has been placed on the disk so that the head can access it immediately after block 0 data has been written to buffer, you lose minimum time. The *gap*, therefore, is the number of blocks that pass under the head while one block of data is being read to buffer.

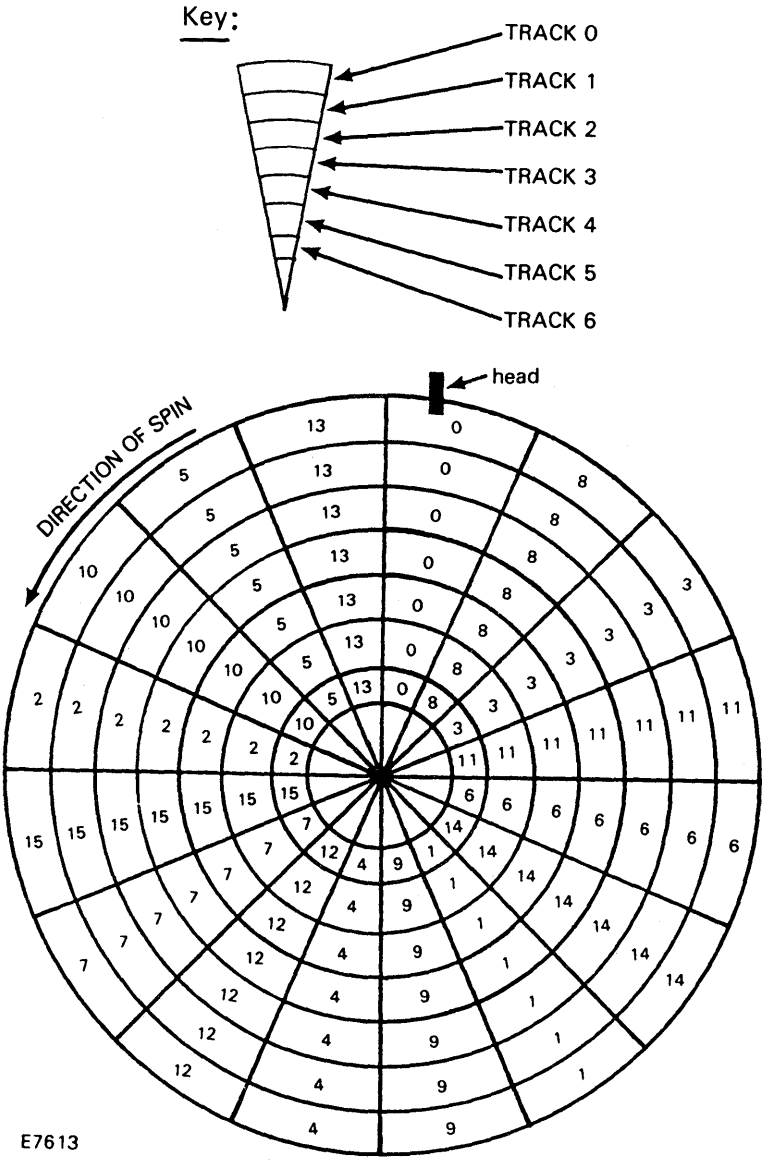
The best *gap* varies for different processors and applications. The recommended *gap* value for the processor on your system is 5 for file systems on built-in disks and 9 for file systems on SMDs. Figure 5-2 illustrates a 16-block disk (typical for this system) defined with a *gap* of 5. Note that the cylinder moves in a counterclockwise direction. With the *gap* set to 5, this is the sequence:

- The head reads block 0.
- Blocks 8, 3, 11, 6, and 14 pass under the head, unread.
- The head reads block 1.
- Blocks 9, 4, 12, 7, and 15 pass under the head, unread.
- The head reads block 2.
- Blocks 10, 5, 13, 0, and 8 pass under the head, unread.
- and so on.

Note that there is a gap of 6, rather than 5, between blocks 7 and 8. After the head reads block 7, then skips 5 blocks, it reaches block 0, which has already been read. It skips one extra block to reach block 8, then continues with the pattern.

You may want, particularly if you notice performance degradation on certain applications, to test gaps other than 5 (or 9) on file systems used with those applications.

Figure 5-2 **A 16-Block Disk with a Gap Factor of 5.**



- *cylinder* represents the number of blocks on a cylinder. A cylinder is a theoretical slice of the disk containing all of the tracks with the same number on each usable surface.

  To calculate *cylinder*, multiply the number of tracks on the disk times the number of sectors. If you do not know these values, use one of the CENTIX/BTOS text editors (**ofed** or **ofvi**) to look at the BTOS configuration file for the disk. The configuration files are named [sys]<sys>IV*disktype*.cnf.

  For example, to calculate *cylinder* for a built-in Atasi disk, look at the file [sys]<sys>IVAtasi46.cnf. These lines are in the file:

  ```
  TRACKS=7
  SECTORS=16
  ```

  Multiply 7 x 16 = 112. The *cylinder* value for this disk is 112.

  The configuration files are described in the *CENTIX Installation and Implementation Guide*.

Making a File System with a Pilf Factor

When you create a file system, you can specify a cluster—or block—size. This block size determines the number of contiguous 1 K–byte blocks in which data in files in the file system will be stored. The data is moved into and out of the files in blocks that are the size that you specify.

To specify the block size, you use the Pilf factor. The Pilf factor is an exponent of two. That is, to specify a block size of eight K–bytes, you use a Pilf factor of three (two to the third power = eight). Note that you can use only block sizes that can be factored by two.

When you use the **mkfs** command to create a file system, you use the -P (Pilf) option to specify a block size of 2 to the (Pilf factor power) K–bytes. For example, the following command line creates a file system with a block size of 2 to the third power, or eight K–bytes:

```
# /etc/mkfs /dev/filename -P 3 size
```

If your file system will contain large data files, using the Pilf factor can improve speed in I/O operations. If your file system will contain small files, using the Pilf factor can waste disk space. If you do not specify a Pilf factor, the file system uses the default Pilf factor of zero. The data in the file system, therefore, is stored in 1 K–byte blocks (2 to the zero power = 1).

You will see the greatest improvement in I/O speed when you match the Pilf factor of a file system with the buffer size specified in the application program that uses the file system. To determine this buffer size, see the documentation for the application program.

Files in a file system retain the Pilf factor specified when the file system was created. You can, however, specify a different Pilf factor for a file when you copy it into the file system with the **cp** command. For example, the following command creates *newfile* with a block size of 16 K–bytes (two to the fourth power), regardless of the Pilf factor of either *oldfile* or the file system in which *newfile* resides.

```
# cp -P 4 oldfile newfile
```

Changing the Pilf Factor of a File System

To change the Pilf factor of an already existing file system, or to add a Pilf factor to a file system that does not have one, follow these steps (see your *centrEASE Operations Reference Manual* for details).

1 Back up all of the files in the file system.

2 Remove the file system.

3 Create the new file system with the Pilf factor that you want. Any files that you create in this file system will have the new Pilf factor.

4 Restore the files to the new file system. These files will **not** take on the Pilf factor of the new file system. If you want these files to have the new Pilf factor, you must copy them to new files.

Mounting the File System

Note: *The root file system is automatically mounted when you boot up the system. You do not need to mount root.*

Once you have made the file system, you must make it a part of the CENTIX file system structure. To do this, you need an empty directory to which you can mount the file system. This directory will be the "root" directory of the file system.

Use the **mkdir** command to create the directory. Creating directories is described in detail in your *CENTIX Operations Guide*.

Notes:

1 *You must mount your file system to an empty directory. If the root directory is not empty, its contents will not be accessible after the file system is mounted.*

2 *Remember that all files that you will store in the root directory of your file system, or in any subdirectories of that directory, will be physically stored at the disk device on which you placed your file system.*

Use the **mount** command to mount the file system to the directory. Enter:

```
# /etc/mount filesystem directory
```

where:

- *filesystem* is the pathname of the memory device that you used in the **mkfs** command. Using the example described above, if you are mounting the file system that you created on partition 6 of hard disk 0, *filesystem* is /dev/dsk/c0d00s6.
- *directory* is the pathname of the directory to which you are mounting the file system.

Adding the File System to the /etc/rc.mounts File

Note: *File systems created through centrEASE are automatically added to the /etc/rc.mounts file.*

When the system is brought down, file systems are automatically unmounted. When the system is brought back up, therefore, the file systems must be remounted to their directories.

You do not want to run the **mount** command on all of your file systems each time that you bring up the system. You can, instead, use a text editor to list each file system in the /etc/rc.mounts file. When the system is booted, this file is automatically read and a **mount** command is performed on every file system that is listed. In this way, your file systems are automatically mounted each time you reboot the system.

Adding the File System to the /etc/checklist File

Note: *File systems created through centrEASE are automatically added to the /etc/checklist file.*

The /etc/checklist file contains a list of the file systems checked by the fsck program when the system is brought up (see "Checking a File System with **fsck**" later in this section). All file systems that are mounted when the system is in multiuser mode should be listed in this file.

The /etc/checklist file is created when the system software is installed. When you create a new file system, use a text editor to add the file system to this file.

Creating the lost+found Directory

Every file system must have a directory named "lost+found." This directory is used when the **fsck** command is run (see "Checking a File System with **fsck**" later in this section). After you have created the file system, create the lost+found directory. Enter:

```
# /etc/mklost+found
```

Unmounting a File System

There are times when you need to remove a file system from the file system structure. For example, before you physically

remove a disk device from your system, you must unmount the file systems stored on that device.

Warning: *If you do not logically unmount a file system before you physically remove it, you can lose data or even crash the system.*

Use the **umount** command to remove a file system from the file system structure. Enter:

```
# /etc/umount filesystem
```

where *filesystem* is the pathname of the memory device that you used in the **mkfs** command. In the example used previously, if you are unmounting a file system that was created on partition 6 of hard disk 0, *filesystem* is /dev/dsk/c0d00s6.

Using File System Status Commands df and du

The **df** and **du** commands can be used to check the status of file systems. The **df** command reports the number of free blocks in a specified file system. The **du** command reports the number of blocks contained in the files and subdirectories of a specified directory. For details on using these commands, see your *CENTIX Operations Reference Manual.*

Checking a File System with fsck

The file system check program (fsck) is an interactive file system check and repair program. fsck uses redundant structural information in the CENTIX file system structure to perform several consistency checks. When fsck finds an inconsistency, it reports an error message to the terminal. Appendix A lists the fsck error messages. fsck is also able to repair corrupted file systems.

You should run fsck each time that you bring up the system, before you enter multi-user mode. During the booting up process, a prompt at the console asks if you want the file systems to be checked. Enter "y" (yes). fsck is run automatically.

You should also run fsck whenever you suspect that there has been file system corruption. These are some of the possible causes of file system corruption:

- Shutting down the system incorrectly (see Section 2 for the correct shutdown procedure).
- Accessing an already-corrupted file system.
- Physically removing a mounted file system.
- Hardware failure.

See the subsection below, "Initiating **fsck**," for details on running fsck.

What fsck Checks

The fsck program checks for errors in three areas:

- In the superblock, fsck checks the following:
  - *File system size and i-list size*. The file system must be bigger than the superblock plus the i-list. There must be no more than 65,534 inodes. The file system and i-list size are critical pieces of information to the fsck program. All other checks of the file system depend on these being correct.

- *Free block list*. The free block list starts in the superblock and continues through the free list blocks of the file system. Each free block list is checked for a list count out of range (the count must be in the range of 0 through 50), for block numbers out of range (a pointer cannot point past the end of the file or before the first data block), and for blocks already allocated in the file system. fsck also checks that all blocks in the file system can be found.

  If errors are found in the free block list, fsck can rebuild the list.

- *Free block count*. The superblock contains a count of the total number of free blocks in the file system. fsck compares this count to the number of blocks that are actually free. If the numbers do not match, fsck can replace the count in the superblock with the actual free block count.

- *Free inode count*. The superblock contains a count of the total number of free inodes in the file system. fsck compares this count to the actual free inode count. If the counts do not agree, fsck can replace the count in the superblock by the actual free inode count.

- In the inodes, fsck checks the following:

  - *Format and type*. Each inode contains a mode word that describes the type and state of the inode. Inodes can be one of four types: regular, directory, special block, or special character. If an inode is not one of these types, it has an illegal type.

    Inodes can be in one of three states: unallocated, allocated, or neither allocated nor unallocated. This last state indicates an incorrectly formatted inode. An inode can be incorrectly formatted if bad data is written into the inode list, through, for example, a hardware failure.

    If fsck finds an illegal type or format, it can clear the inode.

- *Link count*. Each inode contains the count of the total number of directory entries linked to the inode. fsck verifies the link count of each inode by moving through the entire directory structure, starting at root, and counting the links. If the counts do not match, either a directory or an inode has not been updated. This error is always minor.

  If the stored link is not zero and the actual link count is zero, fsck can, under operator control, provide a link in the lost+found directory of the file system. If the inode link count and the actual link count are nonzero and unequal, fsck can replace the stored link count by the actual link count.

- *Duplicate blocks*. fsck compares each block number claimed by an inode to a list of already allocated blocks. If a block number is already claimed by another inode, the block number is a duplicate.

  If fsck finds a duplicate block, it looks again at the inode list to find the inode of the duplicate block, then checks the files associated with the inode. By examining the files, fsck can determine which inode is corrupted and should be cleared. Most of the time, the inode with the earliest modify time is incorrect and should be cleared.

  A large number of duplicate blocks probably indicates that CENTIX failed to write out a block of pointers to indirect blocks. When this problem is detected, fsck asks for permission to clear both inodes.

- *Bad blocks*. These blocks cannot be found because their addresses are invalid.

  If an inode has a large number of bad blocks, CENTIX probably failed to physically write out a block of pointers to indirect blocks. When this problem is detected, fsck asks for permission to clear the inode.

- *File size*. Each inode contains a 32-bit size field that indicates the number of characters in the file associated with the inode.

fsck checks for inconsistencies in directory sizes. A directory size must be a multiple of 16 because a directory entry contains 16 bytes (2 bytes for the inode number and 14 bytes for the file or directory name).

fsck also checks that the number of blocks actually used matches the number indicated by the inode size. The program calculates the number of blocks that should be in an inode by first dividing the number of characters in an inode by the number of characters per block. The program then rounds off the result, and adds one block for each indirect block associated with the inode.

If fsck finds an inconsistency in directory or inode sizes, it prints a warning to the screen. fsck cannot correct either of these problems.

- In the data blocks, fsck checks for the following:
  - *Reference to unallocated inodes*. This error probably means that CENTIX has failed to physically write out the inode. The fsck program requests permission to remove the directory entry.
  - *Invalid i-number*. This error is probably the result of bad data output to the directory. fsck requests permission to remove the directory entry.
  - *Incorrect current (.) and parent (..) directory entries*. A period (.) must be the first file name entry in the directory and have an i-number equal to the i-number for the current directory. A double period (..) must be the second entry in the directory and be a link to the parent directory. If these entries are incorrect, fsck asks for permission to correct them.
  - *Directories not connected to the file system*. If any are found, fsck can link the directory back into the file system in the lost+found directory. A directory can be disconnected when an inode is written to the file system and the corresponding directory data blocks are not written to the file system.

Initiating fsck

Note: *Except for /dev/root, a file system cannot be mounted when you run fsck on it.*

There are several ways in which you can initiate the fsck program:

- You can initiate fsck on a single file system.
- You can reboot the system, and enter "y" when the prompt appears asking if you want to check the file systems. fsck is then run automatically. USE THIS METHOD IF YOU SUSPECT CORRUPTION IN THE ROOT FILE SYSTEM.
- You can initiate fsck on all file systems listed in the /etc/checklist file.

Each of these methods is explained in the following paragraphs.

Initiating fsck on One File System

If you suspect corruption in one particular file system, you can run fsck on just that file system. Follow these steps:

1 Unmount the file system. Enter:

```
# umount filesystem
```

2 To initiate the **fsck** command, enter:

```
# /etc/fsck filesystem
```

where *filesystem* is the pathname of the file system that you want to check.

Note: *Use the -y option of* ***fsck*** *if you want the command to run as though you would answer yes to all interactive questions. See "Interaction with fsck," below, for an explanation of the questions.*

Initiating fsck on All File Systems

If you suspect corruption in the root file system, stop all work immediately (the damage can spread) and follow these steps:

1 Run the **sync** command three times. The **sync** command cleans out the system buffers. Enter:

```
# sync
```

When the prompt reappears, reenter the command, then repeat.

2 DO NOT run the **shutdown** command. Reboot your system.

3 When the prompt appears asking if you want to check the file systems, enter "y". The system starts up **fsck**.

If you do not suspect corruption in root, and you do not want to reboot, you can follow these steps:

1 Run the **shutdown** command. This automatically unmounts all file systems. Enter:

```
# shutdown
```

2 When the system has entered single user mode, the file systems have been unmounted. Enter:

```
# /etc/fsck
```

fsck is initiated on all systems listed in the /etc/checklist file.

Phases of fsck

fsck operates in five phases:

- *Phase 1 - Check Blocks and Sizes*

  In this phase, the inode list is checked for inconsistencies in the block numbers, file sizes, and inode formats.

- *Phase 2 - Check Pathname*

  In this phase, the directories that point to inodes that were found to contain errors in phase 1 are checked.

- *Phase 3 - Check Connectivity*

  In this phase, fsck checks that all directories have a place in the overall file system hierarchy.

- *Phase 4 - Check Reference Counts*

  In this phase, the number of links for all files and directories are checked. Each file and directory should have one link for each time it is listed in a directory. The number is usually one, but can be greater than one.

- *Phase 5 - Check Free List*

  In this phase, fsck checks the free block list for bad blocks and the total free block count. fsck also checks that all unused blocks that should be listed are listed.

In addition to these five phases, there are possible subphases, which are run or not run according to what fsck finds in the file system. There is also an optional phase 6. If an error is found in phase 5, the error and this message are displayed on the screen :

```
SALVAGE?
```

If you want fsck to repair the problem by reconstructing the free block list, enter y and press the RETURN key. A phase 6 is initiated, in which the free block list is reconstructed.

Interaction with fsck

When fsck discovers an inconsistency in the file system, it stops and displays a message on the terminal screen. Often, the message includes some possible further action. Examples are CLEAR?, CONTINUE? and REMOVE?. To allow the requested action to happen, enter y and press RETURN. If you do not want the action to occur, enter n and press RETURN. All possible fsck screen messages are listed in Appendix A.

You should usually enter y, to allow fsck to correct the file system corruption. Enter n only when fsck wants to do something that would cause you problems, as, for example, when it wants to delete a file that you need.

Caution: *Corrective action by fsck can result in loss of data.*

Running fsck Twice on a File System

If you think that a file system needs extensive repair, you can run fsck twice. The first time, assess the damage and grant permission for minor repairs, such as removing unimportant files or restoring an inode directory link. If serious repairs cannot be avoided, backup the file system before beginning the second fsck run. A likely sign of trouble is a request from fsck to remove a directory that lists many important files and directories.

Managing User Accounts

Note: *The user account management tasks can also be performed through centrEASE. See your* centrEASE Operations Reference Manual *for details.*

As system administrator, you are responsible for controlling who can use the system, and what they can do on the system. This section discusses the following user management tasks:

- Adding a user to the system.
- Barring a user's access to the system temporarily.
- Deleting a user from the system.
- Moving a user to another file system.
- Changing a user's password.

Adding a User to the System

To add a new user to the system, perform these three tasks (each task is explained in detail in the following subsections):

1 Add the user to the /etc/passwd file.

2 Assign a password to the user.

3 Create the user's home directory.

4 Add the user to the /etc/group file (this step is optional).

Adding a User to the /etc/passwd File

Each user needs a login entry in the /etc/passwd file. Use a text editor to add a new user to the file.

Each entry in the password file is a line of the form:

name:password:uid:gid:comment:home:shell

where:

- *name* is the login name of the user. The login name consists of one to eight alphanumeric characters. The first character must be a letter. All letters are lower case. Two users cannot have the same login name.

- *password* is the user password. Leave this entry blank. When you assign the user a password with the **passwd** command, the password is automatically encrypted and inserted into this field. See the subsection "Assigning a Password to the User." If you never assign the user a password, this field remains blank.
- *uid* is a unique, numeric user ID in the range of 0 through 65,535. Do not use 0 through 99; these are reserved. 0 is automatically assigned to superuser.
- *gid* is the numeric ID of the group to which you assign the user. This is the same group number listed in the /etc/group file. The default *gid* is 1. Do not use 0 through 99; these are reserved. If you do not want the user in a group, set this field to 100.

  If you assign the user to a group, you must add the user to the /etc/group file. See the subsection "Adding a User to the /etc/group File."
- *comment* is any miscellaneous information that you want to add about the user. Examples are the user's full name and the department in which the user works. You can leave this field empty.
- *home* is the full pathname of the user's home directory. See the subsection "Creating the User's Home Directory."
- *shell* is the full pathname (not the command name) of the user shell, the program executed when the user logs in. If this field is empty, the user is logged into the default shell, /bin/sh.

The following are sample entries for the /etc/passwd file.

```
root:N+x85RT7Zabc4:0:0::/:/bin/sh

jan::116:200::/usr/jan:/bin/sh

marcia::117:200::/usr/marcia:/bin/sh

gwen:X3+sMB49kxz99:118:300:gwyneth krimmel:/usr/gwen:/bin/sh

jack:DtyLB483eghIW:119:300:engineering:/usr/jack:/bin/sh
```

Maintain the entries in the /etc/passwd file in numeric order. This makes it easier to determine the last user ID that was used when you are adding a new user.

Note: *Do not change the* name, uid, *or* gid *of root. Root should be the first user in the file. The root home directory is always a slash (/), which represents the root directory.*

In /etc/passwd, the user called "root" has the numeric user ID zero (0); this identifies root as the superuser. Under no circumstances should you change the root user ID. This directory should not have any more files in it than necessary.

Assigning a Password to the User

Note: *This process is also used to assign a password to root.*

User passwords are optional. If you do not assign a password, the user can log into the system just by entering his or her login name.

To assign a password to a user, use the **passwd** command. Enter:

```
# passwd name
```

where *name* is the user login name you entered in the /etc/passwd file.

The **passwd** command prompts you for a password. Enter the user's password. The password must fit these requirements:

- The password must have at least six characters. Only the first eight characters are significant.
- The password must contain at least two alphabetic characters and at least one numeric or special character (#, $, %, ¢, &, and so on). The alphabetic characters can be upper and lower case.
- The password must be different from the login *name* and from any reverse or circular shift of that *name*. The upper and lower case of the same letter are considered by the system to be different characters.

Note: *These restrictions do not apply to the superuser password.*

When you press the RETURN key, you are prompted to reenter the password. This is to protect against typing errors.

Creating the User's Home Directory

Note: *Add the user's entry to the /etc/passwd file before you create the home directory. See "Adding a User to the /etc/passwd File," above.*

When a user logs in, he or she is automatically put into the directory defined as that user's home directory in the /etc/passwd file. To create the home directory for the user, follow these steps:

1 Select the home directory name. By convention, the home directory name is often the same as the login name.

2 Create the directory using the **mkdir** command. Enter:

```
# mkdir directory
```

Make sure that you use the full pathname of the directory. For example, if you are creating a directory named "gary," and placing it into the "/writer" file system, enter:

```
# mkdir /writer/gary
```

3 The person who creates a directory is automatically given ownership of that directory. Because you created the user directory, you must change the ownership of the user's home directory. Use the **chown** command. Enter:

```
# chown name directory
```

where *name* is the user login name you entered for the user in the /etc/passwd file and *directory* is the pathname of the user's home directory.

4 If you assigned the user to a group in the /etc/passwd file, use the **chgrp** command to give that group ownership of the directory. Enter:

```
# chgrp group directory
```

where *group* is either the name or the ID of the user's group (see "Adding a User to the /etc/group File," later in this section, for details) and *directory* is the pathname of the user's home directory.

5 Set the access privileges for the directory with the **chmod** command. See your *CENTIX Operations Guide* for a detailed explanation of protection mode. Enter:

```
# chmod accessnumber directory
```

where *accessnumber* is determined by adding numbers that represent who has access to the directory. The *directory* is the pathname of the user's home directory.

To determine the *accessnumber*, decide who can do what to the directory, then add the appropriate numbers from Table 6-1. For example, if the owner is allowed to read, write, and execute the directory, but no one else is allowed any access, you add 400 + 200 + 100 = 700. The command looks like this:

```
# chmod 700 directory
```

Or, if the owner, the group, and all users can read the file or directory, but only the owner can write to and execute it, you add 400 + 200 + 100 + 40 + 4 = 744. The command looks like this:

```
# chmod 744 directory
```

Table 6-1 **Access Numbers for the chmod Command.**

Amount to Add	Access Permission
400	owner is allowed to read the file
200	owner is allowed to write to the file
100	owner is allowed to execute the file
40	group is allowed to read the file
20	group is allowed to write to the file
10	group is allowed to execute the file
4	all users are allowed to read the file
2	all users are allowed to write to the file
1	all users are allowed to execute the file

Adding a User to the /etc/group File

The /etc/group file lists the groups in your system and each user in each group. If you assigned the user to a group in the /etc/passwd file, use a text editor to add the user into the /etc/group file. For each group, there is a line in the file that looks like this:

```
group name:null:group ID:user,user,user,user, ...
```

where:

- *group name* is the name that you have given to the group. The *group name* should be from one to eight characters long, with the first character alphabetic, and the remainder alphanumeric. Do not use uppercase letters.
- null is an empty field. This field contained the group password on earlier systems.
- *group ID* is the numeric identification code for the group. Use a number in the range of 100 through 65,535. IDs 0 through 99 are reserved.
- *user, user, user* are the login names of the users assigned to the group.

If you are creating a new group for the user, add the new line into the /etc/group file. If you are adding the user to a group that already exists, append the user's login name to the end of the list of users in the appropriate line in the file.

Barring a User's Access Temporarily

If you want to temporarily deny a user's access to the system, you can "comment out" the user's entry in the /etc/passwd file. To comment out an entry in a file, use a text editor to add a pound sign (#) in front of that entry. The system will ignore any lines that have been commented out. A line that has been commented out looks like this:

```
#jan::116:200::/usr/jan:
```

To restore Jan's access to the system, remove the # sign from her entry in the /etc/passwd file.

Deleting a User from the System

To remove a user from the system permanently:

1 Use a text editor to remove the user's entry from the /etc/passwd file.

2 Use a text editor to remove the user's name from the /etc/group file, if applicable.

Note: *You should back up the user's files before you perform the next step, in case files that you want to keep are accidentally removed.*

3 Use the **rm** command to remove the user's home directory and files. Enter:

```
# rm -r directory
```

Caution: *Using the* **rm** *command with the -r option can remove a large number of directories quickly. The -r option tells the command to remove files even if they are directories. Use this command carefully.*

Moving a User

To free up space in a file system, you can move a user to another file system. Follow these steps:

1 Create a new home directory for the user in the new file system.

2 Use the **mv** command to move the user's files to the new home directory.

3 Use a text editor to change the home directory name in the /etc/passwd file.

Changing a User's Password

There are two ways to change a user's password:

- If you know the old password, you can log in under the user's login name, then enter the **passwd** command. The system prompts you for the old password. When you enter it, the system prompts you for the new password, then prompts you again for the new password.
- If you do not know the old password, log in as superuser, use a text editor to enter /etc/passwd, and remove the encrypted password entry. Use the **passwd** command to reassign a password.

Managing System Security

As system administrator, you are responsible for maintaining security in both the CENTIX and BTOS portions of the operating system. This section covers both.

Managing CENTIX System Security

You have two basic methods of maintaining security in the CENTIX file system:

- Assigning passwords to users.
- Assigning access privileges to directories and files.

You must determine how much security and what kinds of security you need on your system. The details are in the following subsections.

Assigning Passwords

If system security is an issue, the most important password is the one you assign to superuser. Someone who knows the superuser password can access any file or directory on the system. He or she can change anyone's password, including superuser's, and lock everyone out of the system. Change your superuser password often.

***Note:** If you forget the superuser password, you must reboot the system software in the restricted mode (see Section 15 of this guide). Mount the root partition, then use a text editor to delete the encrypted password from the root entry in the /etc/passwd file. Re-enter normal mode and use the* **passwd** *command to assign a new superuser password.*

Assigning passwords to users is also important. It keeps users out of other users' files. It can also keep unauthorized users out of the system. Because many users use their own names as login names, someone could break into the system by logging in under the name of a user.

You use the **passwd** command to assign a password. Assigning and changing user passwords are explained in detail in Section 6.

Forcing Users to Change Their Passwords Periodically

Users should change their passwords periodically. If someone discovers a user's password, he or she will have access to the system for only as long as the password is in effect.

To force a user to change his or her password periodically, you can edit the user's entry in the /etc/passwd entry. Follow these steps:

1 Use a text editor to enter the /etc/passwd file. The password entry for the user looks like:

name:*password*:*uid*:*gid*:*comment*:*home*:*shell*

where *name* is the user's login name and *password* is a 13-character encrypted form of the user's password (see Section 6 for explanations of the other fields).

2 At the end of the 13-character *password* entry, add a comma, then two characters from the set shown in Table 7-1.

The first character represents the maximum number of weeks that the user can use the password before he or she is forced to provide a new password. The second character represents the minimum number of weeks that the user must keep the password before he or she can change to a new password.

For example, if you add ",N8" after an encrypted *password*, the user must change the password at least every 26 weeks (N=26) and must keep that password for at least 10 (8=10) weeks before changing it.

If the user tries to log in after his or her password has expired, the system does not accept the password.

3 Once you have added the extra fields to the *password* field, the system begins to keep track of how many weeks a user has a password. This information shows up as two encrypted characters after the three characters that you have added. This is an example:

name:X3+sMB49kxz99,N8z8:*uid*:*gid*:*accounting*:*home*:*shell*

Table 7-1 **Values Used in the /etc/passwd File to Force Users to Change Passwords.**

Character to Use	Number of Weeks It Represents	Character to Use	Number of Weeks It Represents	Character to Use	Number of Weeks It Represents
.	0	K	22	f	43
/	1	L	23	g	44
0	2	M	24	h	45
1	3	N	25	i	46
2	4	O	26	j	47
3	5	P	27	k	48
4	6	Q	28	l	49
5	7	R	29	m	50
6	8	S	30	n	51
7	9	T	31	o	52
8	10	U	32	p	53
9	11	V	33	q	54
A	12	W	34	r	55
B	13	X	35	s	56
C	14	Y	36	t	57
D	15	Z	37	u	58
E	16	a	38	v	59
F	17	b	39	w	60
G	18	c	40	x	61
H	19	d	41	y	62
I	20	e	42	z	63
J	21				

Assigning CENTIX File and Directory Access Privileges

When a file or directory is created, it is automatically assigned an owner. The owner is the person who created the file or directory. You can use the **chown** command to change the ownership of a file. Enter:

```
# chown name pathname
```

where *name* is the login name of the person to whom you are giving ownership of the file or directory and *pathname* is the file or directory that you are changing. Only superuser or the current owner can change the ownership of a file or directory.

You may also need to change the group ownership of a file (when you change the file's owner, the file's group ownership does **not** automatically change to the new owner's group). Use the **chgrp** command. Enter:

```
# chgrp groupname pathname
```

where *groupname* is the name of the new group and *pathname* is the file or directory that you are changing.

Access privileges are also assigned to a file or directory when it is created. The access privileges determine who (the owner, the owner's group, or everyone) can do what (read, write, and/or execute) to that file or directory. Access privileges are assigned according to the default creation access permission defined in the system minus any **umask** value (see "Using **umask** to Set Default File Creation Modes" below). To change the access privileges of a file or directory, use the **chmod** command. The **chmod** command and access privileges are explained in detail in your *CENTIX Operations Guide*.

The ownership and access privileges of files and directories are powerful tools. If, for example, the ownership of the /etc/passwd file is set to superuser, and the access privileges are set so that any user can read the file, but only superuser can write to or access the file, the /etc/passwd file is protected against unauthorized users.

In this system, shell commands are also files. This means that you can use the access privileges to control who uses which commands. Good examples are the commands **mount** and **umount**. If you do not want users to mount or unmount file systems, verify that the ownership of the commands are set to superuser, and that the access privileges are set so that only the owner can execute the file.

Using umask to Set Default File Creation Modes

Default file and directory creation permission values are set up when the system software is installed. You can, as explained in Section 6, use the **chmod** command to change the access permission of any file or directory. If, however, you are doing **chmod** on all of the files that a user or users are creating, you can instead use the **umask** command to change the default creation permission values.

umask is used to remove file and directory access permission that would normally be granted. The value that you assign with **umask** is subtracted from the value that would be automatically assigned to the file. For example, if a user's files would normally be created with an access permission of 777 (all users can read, write, and execute the file), you can use **umask** to remove read and write permission from everyone but the user.

The **umask** values are shown in Table 7-2. Note that they are the opposite of the **chmod** values. These values are subtracted from the default access permission value. In the example, to remove read and write permission from the group and others, you assign a **umask** value of 066. This causes the causes the file to be created with a permission of 711 (777 minus 66).

umask, unlike **chmod**, is not done for a particular file or directory, but for a creator of a file or directory. To change the default access permissions for all users, put the **umask** command into the /etc/profile file. To change the default access permission for one user, put the **umask** command into that user's .profile file.

The command has the form:

```
# umask nnn
```

where *nnn* is the **umask** value.

Table 7-2 **Access Denial Numbers for the umask Command.**

Amount to Add	Access Denial
400	owner is not allowed to read the file
200	owner is not allowed to write to the file
100	owner is not allowed to execute the file
40	group is not allowed to read the file
20	group is not allowed to write to the file
10	group is not allowed to execute the file
4	no one is allowed to read the file
2	no one is allowed to write to the file
1	no one is allowed to execute the file

Managing BTOS System Security

BTOS file system security is established in two steps:

1 You assign a protection level to each file that you want protected. The file's assigned protection level determines which passwords — volume, directory, and/or file — must be used to access the file.

2 You assign passwords to BTOS volumes, directories, and files.

Caution: *You can not assign a protection level other than 15 to any BTOS file that is accessed by CENTIX, for example, the files listed in the [sys]<sys>ConfigUfs.sys file.*

Assigning BTOS Passwords

Setting or Changing a Volume Password

A volume password can be set by using either the **MIVolume** command or the **MChange Volume Name**. If you are initializing a disk for the first time, have already backed up valid data on the volume, or do not intend to save the data on the volume, you can use **MIVolume**. If you do not intend to initialize the volume or destroy valid data, use **MChange Volume Name**.

To set a volume password for the first time, enter the password in either the [Volume password] field of **MIVolume** or in the [New volume password] field of **MChange Volume Name**.

To change a volume password, enter the new password in the [New volume password] field of **MChange Volume Name**. You must also enter the old password in the [Old volume password] field. (**MIVolume** cannot be used to change passwords because the volume password entered in the [Volume password] field must be the currently assigned password.)

Note: *If you want to set file security for files on a volume, you must assign a password to the volume.*

Files on a volume with no assigned password are unprotected, regardless of whether the volume's directories or files have passwords, or what file protection levels have been set.

Changing the BTOS System Disk Volume Name and Password

You cannot use the **MChange Volume Name** command to change the volume name or password of the BTOS system disk. This is because the system disk is used when the MChange Volume Name utility is running.

Use the following procedure to change the system disk volume name or password.

1 Edit the file [sys]<sys>BootLoadParams to include the new volume name and/or password. Use one of the CENTIX/BTOS editors, **ofed** or **ofvi**. (Refer to the *XE 500 CENTIX Installation and Implementation Guide* for an explanation of the BootLoadParams file.)

2 Put your system's Boot Load release medium on line.

3 Reboot the system by turning the base enclosure keyswitch to STOP and then to MANUAL.

   This causes the system to execute the MChange Volume Name utility, using the volume name and password specified in the BootLoadParams file.

   When a 90 is displayed, the MChange Volume Name utility has executed successfully.

5 Remove the Boot Load medium.

6 Reboot the system to normal mode by turning the base enclosure keyswitch to STOP and then to NORMAL.

Note: *To maintain proper system security, you should modify the BootLoadParams file to remove the system disk volume password once the system has returned to normal mode.*

Setting or Changing a Directory Password

A directory password can be set by using either the **MCreate Directory** command or the **MSet Directory Protection** command. To set a directory password for the first time, enter the password in either the [Password for new directory] field of **MCreate Directory** or in the [Volume or directory password] field of **MSet Directory Protection**.

To change a directory password, enter the new password in the [New password] field of **MSet Directory Protection**. You must also enter the old password in the [Volume or directory password] field.

Setting or Changing a File Password

If you want to change only the password for a file, use the **MSet File Protection** command. In the command form, specify the file's current protection level and the new password. You can determine the file's current protection level by using the **MFiles** command with the details option.

Setting a File's Protection Level

Files can be protected from being read, from being modified, or from being both read and modified. The file's assigned protection level determines which passwords can be used to access the file.

File protection levels are designated by numbers. Table 7-3 lists the file protection levels and describes the type of protection they provide.

Use the **MSet File Protection** command to assign a new protection level or a new password to files.

You can assign a new protection level or a new password to one file or, by using the wild card character (*) in the file name designation, to a group of files.

By default, the **MSet File Protection** command assigns a new protection level to the files specified but leaves the previously assigned password. Optional fields also allow you to change the password and to confirm the protection change for each file specified.

Table 7-3 **BTOS File Protection Levels.**

Caution: *You can not assign a protection level other than 15 to any BTOS file protection level that is accessed by CENTIX, for example, the files listed in the [sys]*<sys>ConfigUfs.sys file.

Protection Level	Password Needed to Access File
0	To read file: Must provide either volume or directory password. To modify file: Must provide either volume or directory password.
1	To read file: Must provide either volume, directory, or file password. To modify file: Must provide either volume or directory password.
3	To read file: Must provide either volume, directory, or file password. To modify file: Must provide either volume, directory, or file password.
5	To read file: No password required. To modify file: Must provide either volume or directory password.
7	To read file: No password required. To modify file: Must provide either volume, directory, or file password.
15	To read file: No password required. To modify file: No password required.
19	To read file: Must provide either volume, directory, or file password. To modify file: Must provide either volume or file password.
23	To read file: No password required. To modify file: Must provide either volume or file password.

Accessing Protected BTOS Files

To access a protected BTOS file with a CENTIX command, you must append a caret (^) and the file password to the file name. For example, to use **ofcopy** to copy a BTOS file that is password-protected to a CENTIX file, enter:

```
# ofcopy 'BTOSfilename^filepassword' CENTIXfilename
```

If you do not specify a password in this way, the system defaults to your user password.

Depending on the file's protection level, the specified password must match the file's assigned password, the password of the directory in which the file is located, or the password of the volume in which the file is located.

Backing Up and Restoring Files

Note: *The disk that holds your BTOS operating system cannot be backed up through the CENTIX commands that are described in this section. Use the centrEASE administrative facility to back up your BTOS system disk.*

You can also backup and restore your CENTIX files through centrEASE, rather than using the commands described in this section.

See your centrEASE Operations Reference Manual *for details.*

This section describes ways in which you can back up and restore your CENTIX files. Backing up files should be a regular part of your job as administrator. Both human and machine error can delete large amounts of data quickly. When this happens, restoring deleted data from backup media can avoid disaster.

There are several commands that you can use to do backups and restores. This section describes two of these commands in detail, and gives a brief description of other commands that are available to you. At the end of this section is a suggestion for scheduling your backups.

Note: *For instructions on using half-inch and QIC tapes, see your* CENTIX Operations Guide.

Using filesave to Do Daily Backups

The **/etc/filesave** command is a sample shellscript provided with the system software. You can modify the shellscript so that it will automatically copy the files that get changed every day. One sample is provided in the shellscript:

```
ROOT=/dev/dsk/cod02s0
BACKUP=/dev/dsk/cxdxxsx
dd if=$ROOT of=$BACKUP bs=1024
```

Use a text editor to change the script to match your system. Make these changes:

- Change "ROOT" to the name of the file system you want to back up every day.
- Change "/dev/dsk/c0d02s0" to the device on which the file system resides.

- Change "cxdxxsx" to the device to which you are backing up.
- Change "1024" to the size, in bytes, of the file system that you are backing up. The number must be a multiple of 512.
- If you want to list more than one file system for daily backup, rather than defining the terms, then using the terms in the command line, add the variables directly into the command line. Then repeat the **dd** command line for each file system you want to back up.

Once you have modified **filesave**, to run a daily backup, follow these steps:

1 If you are backing up to a QIC tape, use the BTOS **MQic Retension** command through centrEASE (see your centrEASE Operations Reference Manual) or **ofcli** (see Section 13 of this guide) to wind and rewind the QIC tape. **MQic Retension** is explained in detail in the *BTOS Operations Reference Manual*.

2 Enter:

```
# filesave
```

Your modified shellscript performs the backup.

To restore file systems that were backed up with **filesave**, use the **dd** command.

For details on **dd**, see your *CENTIX Operations Reference Manual*.

Using tapesave to Do Weekly Backups

The **/etc/tapesave** command is a sample shellscript provided with the system software. You can modify the shellscript so that it will automatically do a full backup on your system. The same sample that is provided with **/etc/filesave** is provided with **/etc/tapesave**.

Use a text editor to change the sample so that the file systems that you want backed up at the end of each week are listed. Follow the instructions given above for **/etc/filesave**.

Once you have modified **filesave**, to run a weekly backup, follow these steps:

1 If you are backing up to a QIC tape, use the BTOS **MQic Retension** command through centrEASE (see your centrEASE Operations Reference Manual) or **ofcli** (see Section 13 of this guide) to wind and rewind the QIC tape. **MQic Retension** is explained in detail in the *BTOS Operations Reference Manual*.

2 Enter:

```
# tapesave
```

Your modified shellscript performs the backup.

To restore file systems that were backed up with **tapesave**, use the **dd** command.

For detail on **dd**, see your *CENTIX Operations Reference Manual*.

Other Backup and Restore Commands

The commands listed below can also be used to backup and restore files. For details on using these commands, see your *CENTIX Operations Reference Manual*.

- **cpio** is used to copy files from one device to another in one-block portions. This command is usually used to copy small files. You must use **cpio** to restore a file that you originally backed up with **cpio**.
- **tar**, like **cpio**, is used to copy files from one device to another. With **tar**, however, you can specify a blocking factor. This blocking factor determines how many blocks are read at a time during the copy. Because using the blocking factor saves time and tape space, you usually use **tar**, rather than **cpio**, to copy large files. You must use **tar** to restore a file that you originally backed up with **tar**.
- **dd** is also used to copy files. With **dd**, you can use options to convert the files. For example, you can convert the blocking factor of the file, or change the file from EBCDIC to ASCII and vice versa.
- **volcopy** is used to copy entire file systems with label checking.

- **finc** is used to copy entire file systems, one file at a time, to backup media. You must use the **frec** command to restore file systems that you back up with **finc**. **finc** and **frec** are complicated commands and should be used only by the experienced user.

Scheduling Backup

You should make backups often enough so that if files are lost, you will not have a disaster. On the other hand, you do not want to spend all of your time backing up files. You can use the schedule shown below until you get a good sense of how often files need to be backed up on your system.

- At the end of each day, copy all user files that have been created or changed during the day. Keep these backups for three to five days before you recycle the media.
- At the end of each week, copy the entire file system. Keep these backups for eight weeks before recycling the media.
- Every other month, when you copy your entire file system, store the backup permanently. To avoid losing data because of degeneration of your media, copy the backups to new media once a year.

Administering the Printer Spoolers

A printer spooler is a system service that manages the transfer of data from disk files to printers. Spooled printing allows many users to share one or more printers. While direct printing can keep a user waiting until a printer is available, spooled printing allows the user to issue a request for printing at any time and then proceed to other activities.

The XE 500 CENTIX system contains two separate printer spoolers, the lpr spooler and the lp spooler. The lpr spooler uses a BTOS print queue. The lp spooler does not use a BTOS print queue. Both spoolers can be installed on the same system. They cannot, however, share printers.

Printers in the system can be connected to CP or TP boards. Serial printers can also be connected directly to the PT 1500 terminals. Printers connected to a PT 1500 are controlled by the printer spooler running on the CP to which the PT 1500 is connected.

Configuring and Administering the lpr Printer Spooler

You use the CENTIX **lpr** command to send files to the lpr printer spooler. For details on using **lpr**, see the *CENTIX Operations Guide*.

For instructions on configuring the lpr printer spooler, see the *CENTIX Installation and Implementation Guide*.

To check the status of the lpr spooler, use the BTOS **MSpooler Status** command. See Section 13 for instructions on using BTOS commands from CENTIX. For details on the **MSpooler Status** command, see the *BTOS Operations Reference Manual*.

Configuring and Administering the lp Printer Spooler

The lp spooler handles print requests made with the **lp** command. (See your *CENTIX Operations Guide* for details on using **lp**.) Before using **lp**, you must configure the lp spooler in both the BTOS and CENTIX operating systems to make it applicable for your installation. This section describes how to configure the lp printer spooler and how to administer the system after it has been configured and is in use.

This system also allows you to set up a terminal as an lp printer. This is described at the end of the section on configuring the printer spooler.

Configuring the lp Printer Spooler into BTOS

The procedure for configuring the lp spooler into the BTOS operating system is explained in the following paragraphs.

Creating a Port Entry in the Processor Configuration File

When serial printers are attached to a CP or TP, the processor's configuration file, [sys]<sys>*Xpnn*.cnf, must contain a line entry for the RS-232-C port to which the printer is connected. To create the port entry, use either a text editor or the MBTOS Config utility. The MBTOS Config utility is explained in the *CENTIX Installation and Implementation Guide*.

Installing the lp Spooler into the Processor Initialization File

If you are going to use the lp spooler to access a parallel printer, you must install the lp spooler into the initialization file, [sys]<sys>init*Xpnn*.jcl, for the CP or TP to which the printer is attached. The line that must be put into the initialization file is:

```
$run [sys]<sys>LpSpooler.run
```

Note: *If you are accessing only serial printers through the lp spooler, do not install the lp spooler into the processor initialization file.*

Use either a text editor or the MBTOS Config utility to add this line to the file. The MBTOS Config utility is explained in the *CENTIX Installation and Implementation Guide*.

Rebooting the System

After making changes to processor files, reboot the system to force the BTOS operating system to read the changed files.

To reboot the system, see Section 2 for directions on shutting down CENTIX. When CENTIX has entered single user mode, turn the front panel keyswitch to STOP, then back to NORMAL. When the front panel display shows a "20," you can bring CENTIX back up, as described in Section 2.

Configuring the lp Printer Spooler into CENTIX

The lp spooler system is made up of these parts:

- *Printing devices*, including parallel and serial printers, which are connected to CP or TP boards. Serial printers can also be connected directly to PT 1500 terminals. (Terminals can also be configured to act as printing devices in the lp spooler system.)
- *Device files*, one for each printing device. Device files have the form /dev/lp*nn* for parallel printing devices, /dev/tty*nnn* for serial printing devices, and /dev/tp1*nnn* for serial printing devices connected to PT 1500s. The device files represent the printing devices in the file system.
- *Logical printers*, which the administrator creates and associates with device files. When you use the **lp** command, you name the logical printer (not the printing device or the device file) as the destination for your file.
- *Interface programs*, which the administrator assigns to the logical printers. An interface program contains information about the printing device with which the logical printer is associated (through the device file).

- *Classes*, into which you can divide the logical printers on your system. A class can be used instead of a logical printer as a destination in an **lp** command. When a class is used as a destination, the file being printed is printed at the first free printer in the class.

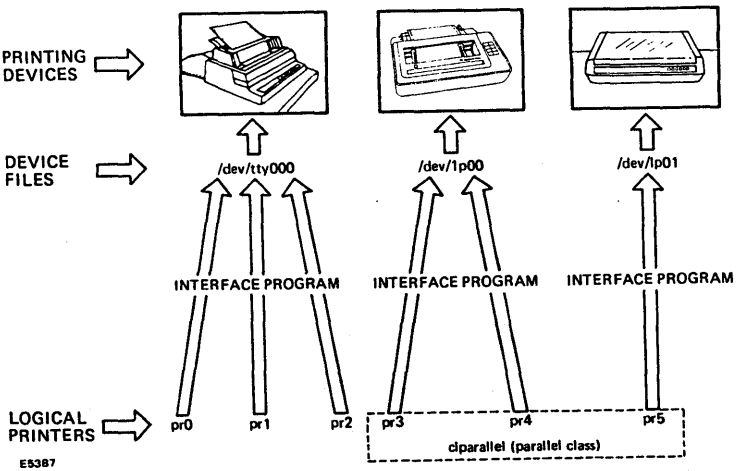
Figure 9-1 shows an example of an lp spooler system.

These are the basic steps for configuring a printer into the lp spooler system:

1. Create, if it does not already exist, the interface program appropriate for the printing device.
2. Create, if it does not already exist, the device file that will represent the printing device.
3. If you are configuring a serial printing device, establish the tty port as a printer, rather than as a terminal port by (1) giving the lp spooler exclusive access to the tty port to which the printing device is connected, and (2) ensuring that the **getty** command will not attempt to log users onto the port.
4. Use the **lpadmin** command to (1) name the logical printer; (2) link the logical printer with a device file; (3) assign the appropriate interface program to the logical printer; and (4) assign, if appropriate, the logical printer to a group—or class—of printers.
5. Use the **enable** and **accept** commands to activate the printer in the lp spooler system.
6. You can also use **lpadmin** to define a default destination for the system, if appropriate.

These steps are explained in the following subsections.

Figure 9-1 **An Example of an lp Spooler System**



Creating Interface Programs

Each logical printer must have assigned to it an interface program. The interface program describes to the lp spooler system the printing device at which the logical printer is going to print its files. Interface programs are shell procedures or some other executable program.

The interface programs supplied with the system are shell procedures. The programs are named for the printing device that they represent. They all reside in the /usr/spool/lp/model directory.

When you use **lpadmin** to associate an interface program with a logical printer, the interface program is copied into the /usr/spool/lp/interface directory and renamed. For example, to copy the /usr/spool/lp/model/dumb file to the /usr/spool/lp/interface/parallel file, enter:

```
# /usr/lib/lpadmin -pparallel -mdumb
```

If you need to revise an interface program that resides in /usr/spool/lp/model, you should not change the original. Use the **cp** command to copy the program file into the /usr/spool/lp/interface directory, then use a text editor to make the changes to the new file.

Note: *The first line of an interface program for a serial printer must be the* ***stty*** *command. This command sets the baud rate and other characteristics of the printing device. The* ***stty*** *command is needed because there are no configuration files in the BTOS system for serial printers that are run through the lp spooler. See your* CENTIX Operations Reference Manual *for details on* ***stty****.*

Verifying the Device Files

Each printing device on the lp spooler is associated with a device file. Device files do not contain information. The kernel uses device files as channels through which to communicate with peripheral I/O devices.

Device Files for Parallel Printers. Device files for parallel printers are named /dev/lp*nn*. The *nn* is a unique number based on the physical position in the enclosure of the processor board to which the printer is attached. Each TP and CP has one parallel port to support a parallel printer. The parallel ports are numbered sequentially from 00, starting at the CP or TP closest to the FP in the first enclosure. Each processor board is assigned a number, whether or not a parallel printer is assigned to the board.

A parallel printer, therefore, that is connected to the processor board closest to the FP, has a device file /dev/lp00. A printer connected to the processor board in the next slot in the enclosure has a device file /dev/lp01.

To create a device file, use the **mknod** command. Enter:

```
# mknod /dev/lpnn c 7 m
```

where *m* is equal to the number given to the device file. For /dev/lp00, *m* = 0; for /dev/lp03, *m* = 3.

Device Files for Serial Printers. Device files for serial printers connected to a CP or TP are named /dev/tty*nnn*. The *nnn* is a unique number based on two things:

- The physical position in the enclosure of the processor board to which the printer is attached.
- The number of RS-232-C serial devices and the number of PT 1500s listed in the configuration files of the system's processor boards.

Note: *For serial printers that are connected to PT 1500s, see the next subsection, "Device Files for PT 1500 Printers."*

Each CP can support three RS-232-C serial devices and 16 PT 1500s. Each TP can support ten RS-232-C serial devices. The serial devices and PT 1500 ports are assigned tty numbers sequentially from 00, starting at the CP or TP closest to the FP in the enclosure.

To determine which tty number you should assign to a serial printer, you must count how many serial devices and PT 1500s are connected between the serial printer and the start of the numbering sequence (the CP or TP closest to the FP). To do this, look at the configuration file for each CP and TP on the system. These files show how many RS-232-C serial devices are configured for each CP or TP, and how many PT 1500s a CP is supporting.

Start at the CP or TP closest to the FP and count until you reach the position of the new serial printer:

- For each CP, the number of serial devices (maximum of 3) plus the number of PT 1500s (maximum of 16) connected to the board.
- For each TP, the number of serial devices connected to the board (maximum of 10).

When you add together the results from the CPs and TPs, the next number, minus 1, is the tty number for the new serial printer. (You have to subtract 1 because the tty numbering starts at 00.)

For example, if you are adding a second printer to a TP board that comes after two CP boards in the enclosure, each of which contains 2 serial printers and 12 PT 1500s, count:

- For CP00: 2 printers plus 12 PT 1500s = 14.
- For CP01: 2 printers plus 12 PT 1500s = 14.
- For TP00: 1 printer = 1.

Because (14 + 14 + 1) -1 = 28, the second printer on the TP board will have a tty number of 28.

Note: *The tty numbers are dependent on how many serial devices and PT 1500s are* currently *configured into the system. If, for example, you remove a PT 1500 with a tty number of 12 from the first CP board and make the appropriate configuration file modifications, the devices with tty numbers of 13 and up will change.*

To create a device file for a serial printer, use the **mknod** command. Enter:

```
# mknod /dev/ttynnn c 0 m
```

where: *m* is equal to the number given to the device file. For /dev/tty000, *m* = 0; for /dev/tty003, *m* = 3.

Device Files for PT 1500 Printers. Device files for serial printers connected directly to PT 1500 terminals are named /dev/tp1*nnn*. The *nnn* is equal to the tty number currently assigned to the PT 1500 to which the printer is connected.

Use the **tty** command to determine the current tty number of the PT 1500. Log in at the terminal to which the printer is connected and enter:

```
# tty
```

The system returns:

```
/dev/ttynnn
```

where *nnn* is the current three–digit tty number of the terminal.

Use this number when creating the device file for the printer with the **mknod** command. Enter:

```
# mknod /dev/tp1nnn c 14 m
```

where:

- *nnn* is equal to the current tty number of the terminal to which the printer is connected.
- *m* is equal to the number given to the device file. For /dev/tty000, *m* = 0; for /dev/tty003, *m* = 3.

The tty number for a terminal is reassigned each time the terminal is booted up. Thus, the reassigned number can easily be different from the tp1 number first assigned to the device file. The tp1 number can be automatically changed to match the new tty number by way of the **mvtpy** command.

For this reason, each time that the user boots up a terminal to which a printer is connected, he or she must use the **mvtpy** command before using the printer (see your *CENTIX Operations Reference Manual* for details on **mvtpy**.) The **mvtpy** command automatically runs the **lpadmin** command for the printer connected to the terminal. The **lpadmin** command reassigns the tp1 number of the device file to match the new tty number of the terminal.

Note: *The* ***mvtpy*** *command can only link the new tty number with the correct /dev/tp1*nnn *file if the /dev/tp1*nnn *file exists. You should, when first installing the lp spooler, use the* ***mknod*** *command to create a /dev/tp1*nnn *file to match each of the tty numbers that could be dynamically assigned to the PT 1500 terminals that have printers. See "Device Files for Serial Printers" earlier in this section for details on how tty numbers are assigned.*

For Serial Printers Only: Defining the tty Port

You must establish that the tty port to which the serial printing device is connected is a printer port, rather than a port for some other type of serial device. To do this, first establish the ownership and accessibility of the port so that only the lp spooler can access it. Enter:

```
# chown lp /dev/ttynnn
# chmod 600 /dev/ttynnn
```

where:

- /dev/tty*nnn* is the name of the device file associated with the printer. See "Device Files for Serial Printers."
- 600 is an octal number that indicates the device file accessibility. With an accessibility code of 600, the file can be read and written to only by the file's owner (lp).

See your *CENTIX Operations Reference Manual* for more information on the **chown** and **chmod** commands.

You must also ensure that the /etc/**getty** command does not attempt to log users onto the port. Use a text editor to enter the /etc/inittab00 file. Check the line entry for the printer's tty port. The third entry should be "off," not "respawn," as follows:

```
nnn:rstate:off:/etc/getty ttynnn baudrate
```

where:

- *nnn* is the tty port at which the printer is connected.
- *rstate* is an assigned number that defines the specific system processing times at which this entry is allowed to be processed. Do not change this number.
- *baudrate* is used as a pointer to the appropriate entry in the /etc/gettydefs file. By convention, the pointer is usually the line speed at which the tty line is configured.

If you do make any change to the /etc/inittab00 file, you must enter the following to put the change into effect:

```
# init q
```

See Section 3 of this guide for more information on the /etc/inittab00 file and on **getty**.

Using lpadmin to Define the Printer to the System

Note: *Many functions of* ***lpadmin*** *cannot be used while the lp scheduler, lpsched, is running. [You can use* ***lpadmin*** *while lpsched is running to change the device file associated with a logical printer (-**v** option) or to set the default system destination (-**d** option).] Before using* ***lpadmin****, make sure that lpsched is halted. To halt lpsched, execute the* ***/usr/lib/lpshut*** *command as superuser.*

You use the **lpadmin** command to define a printer to the operating system. The command takes the form:

```
# /usr/lib/lpadmin -l -Pprinter -vdevicefile -minterprog
```

where:

- *printer* is the name you assign to the printer. A printer name can be up to 14 characters long, and it must consist of alphanumeric characters and underscores.
- *devicefile* is the device file that you want to have associated with the printer. If one does not already exist, you must create the device file with the **mknod** command before you can link it to a printer with the **lpadmin** command. See "Creating the Device Files," earlier in this section.
- *interprog* is the interface program for the printing device associated with the logical printer.
- In the command line shown, the **-m** option is being used to select one of the model interface programs that resides in the /usr/spool/lp/model directory. You can also select a model interface program by using the **-e** or **-i** options of **lpadmin**. See your *CENTIX Operations Reference Manual* for details on **lpadmin**.

Note: *You must use the -l option with* ***lpadmin*** *only when the printing device you are defining is connected to a PT 1500 or when you are configuring a login terminal as a printing device. Do not use the -l option when configuring any other type of printing device.*

Enabling the Printer

Once you have defined a printer, you must introduce it to the lp spooler system. Until you have introduced a printer, it will not accept print requests from the **lp** command. Use the **enable** and **accept** commands, as follows:

```
# enable pr2
printer "pr2" now enabled
# /usr/lib/accept pr2
printer "pr2" now accepting requests
```

The system returns the messages shown on lines 2 and 4 of this example to tell you the printer is enabled and accepting requests.

Assigning a Printer to a Class

Printers controlled by the lp spooler can be assigned to groups—or *classes*—of printers. A class of printers can be used as a destination for a print request. Printers that you put together into a class, therefore, should have something in common. For example, you can assign all serial printers to the serial class (clserial). Then, when you want to print a file on a serial printer, you use clserial as the print destination. The file is printed at the first available serial printer.

Each class must contain at least one logical printer. A logical printer can be assigned to no class, or to one or more than one class.

Note that you do not create a class cl1, then assign logical printers to cl1. You create a class by assigning logical printers to it. That is, class cl1 does not exist unless a logical printer has been assigned to cl1 with the **lpadmin** command.

Use the **-c** option of **lpadmin** to assign a printer to a class. It must be used with the **-p** argument. For example,

```
# /usr/lib/lpadmin -ppr2 -cserial
```

assigns the logical printer pr2 to the serial class of printers.

Defining the System Default Destination

Use the **-d** option of **lpadmin** to set up the default destination. A destination can be a class of printers or a printer. To define a class as the default destination, enter:

```
# /usr/lib/lpadmin -dclass
```

where *class* is the name of the class.

To define a printer as the default destination, enter:

```
# /usr/lib/lpadmin -dprinter
```

where *printer* is the name of the printer.

If you want to change to no default destination, enter:

```
# /usr/lib/lpadmin -d
```

Setting Up a Terminal as an lp Printer

You can set up a terminal as an lp printer. In particular, you may have a paper terminal that you want to use in the lp spooler system. To configure the terminal, use the procedure provided for configuring a serial printing device into the spooler, with the following restrictions:

- You may have to create an interface file. Use the file provided with the system, /usr/spool/lp/model/dumb, as an example.
- Do not turn off the **getty** command, or use the **chown** and **chmod** commands, to establish port ownership and accessibility, as described in "For Serial Printers Only: Defining the tty Port."
- Use the **-l** option with **lpadmin**.

Administering the lp Printer Spooler

The following subsections describe the tasks you will perform on a day-to-day basis as administrator of the lp spooler.

Allowing and Inhibiting the lp Scheduler

The lp scheduler, **lpsched**, is the program that routes **lp** requests through the appropriate printer interface programs to the printing devices. If **lpsched** is not running, the lp spooler cannot print any files.

Each time that **lpsched** is started, a lockfile /usr/spool/lp/SCHEDLOCK is created. SCHEDLOCK prevents a second **lpsched** from being started. If the CENTIX system is shut down, then started up again, you may have to remove SCHEDLOCK before you can restart **lpsched**. To remove the file, enter:

```
# rm -f /usr/spool/lp/SCHEDLOCK
```

You must halt **lpsched** before you use **lpadmin** for any reason except to change the device file associated with a logical printer (**-v** option) or to set the default system destination (**-d** option). Use the **lpshut** command to halt **lpsched**. Enter:

```
# /usr/lib/lpshut
```

When **lpsched** is halted, SCHEDLOCK is automatically removed and any files that are printing are halted. These files are totally reprinted when **lpsched** is restarted.

To restart **lpsched** after it has been halted, enter:

```
# /usr/lib/lpsched
```

If you want **lpsched** to be automatically started by the /etc/rc program when the system is started up, use a text editor to enter the /etc/rc file. Remove the pound sign (#) from in front of the lines:

```
rm -f /usr/spool/lp/SCHEDLOCK

/usr/lib/lpsched
```

With these lines active in /etc/rc, SCHEDLOCK is automatically removed and **lpsched** is automatically started when the system is started.

You may also want **lpsched** to be automatically stopped when the **halt** or **shutdown** commands are used. To set this up, use a text editor to enter the /etc/**halt** and /etc/**shutdown** files. In each file, remove the pound sign (#) from in front of the line:

```
/usr/lib/lpshut
```

Caution: *If **lpsched** is not running when you want to **halt** or **shutdown** the CENTIX system, use a CENTIX editor to replace the pound sign in front of /usr/lib/**lpshut** line in the /etc/**halt** (if you will use **halt**) or /etc/**shutdown** (if you will use **shutdown**) file. System problems can occur if the system tries to halt a program that is not running.*

Determining the Status of the lp Spooler

You can use the **lpstat** command to have the current status of the lp spooler printed to your terminal.

If you use **lpstat** with no options, the current status of all of the current **lp** requests made under your login name is printed to the screen. The status line for each request includes:

```
printername-id    owner    size (in bytes)    date/time
```

where:

- *printername* is the printer to which the file was sent. If you selected a class of printers, this field shows the specific printer within the class to which the file was sent.
- *id* is the unique identification number assigned to your print request by the shell.

If one of the lp requests is currently printing, the printer name is displayed at the end of the status line for that request.

lpstat has several options. Among the information that you can request are the class names and their members, the system default destination that was set up with **lpadmin**, and the current status of **lpsched**. See your *CENTIX Operations Reference Manual* for a list of all of the **lpstat** options.

Preventing and Allowing lp to Route Requests to a Destination

You may need to temporarily prevent **lp** from routing requests to a certain destination. (A destination can be either a printer or a class of printers.) For example, if a printing device has been removed for repairs, you would want to prevent **lp** from routing requests to the destination that uses that printing device. Use the **reject** command to temporarily block a destination. Enter:

```
# /usr/lib/reject destination
```

where *destination* is either a printer or a class of printers.

Note that the **reject** command only prevents new requests from being routed to a destination. It does not shut down the printing device associated with the destination. Any file printing at the time that you do the **reject** will continue to print.

When you want to reopen a destination, use the **accept** command. Enter:

```
# /usr/lib/accept destination
```

where *destination* is either a printer or a class of printers.

See your *CENTIX Operations Reference Manual* for more information on **reject** and **accept**.

Enabling and Disabling a Logical Printer

Just as you use **accept** and **reject** to control lp spooler access to a destination, you use **enable** and **disable** to control spooler access to a logical printer.

Use the **disable** command to prevent an lp request to a logical printer from printing at the device associated with the logical printer. Enter:

```
# disable logical printer
```

The printing device halts, even if a file is printing when the **disable** is done. If the file was originally sent to a class of printers, the file is automatically rerouted to another printing device in the class and totally reprinted. If the file was originally sent specifically to the logical printer that has been disabled, the file is automatically reprinted when the logical printer is reenabled.

To reallow lp requests to be routed to a logical printer, use the **enable** command. Enter:

```
# enable logical printer
```

Moving Requests Between Destinations

You can use the **lpmove** command to move lp requests from one destination to another. For example, if a printing device has a paper jam, you can move all pending requests for that device to another destination. The **lpmove** command can also be used to move single lp requests.

To move all requests for one destination to another destination, enter:

```
# /usr/lib/lpmove fromdestination todestination
```

To move specific requests from one destination to another destination, enter:

```
# /usr/lib/lpmove printername-id todestination
```

where:

- *printername* is the printer to which the file was sent. If you selected a class of printers, this field will show the specific printer within the class to which the file was sent.
- *id* is the unique identification number assigned to your print request by the shell.

Note: *The printer name and the id number are printed to your screen when you use the* ***lp*** *command.*

Setting a User Default Destination

You can define a default destination for a user. Any files that the user prints will be sent to the printer or class of printers defined as the default for that user. If the user does not have a defined default destination, the files are routed to the system default destination. (See "Defining the System Default Destination," earlier in this section.)

To set a default destination for a user, use a text editor to create, if one does not exist, a file called .profile. Put .profile into the user's home directory. Insert these lines into the .profile file:

```
$ LPDEST=destination
$ export LPDEST
```

where *destination* is the user's default destination.

Setting Parallel Printer Options

To set the page format options — indent, number of columns, and lines per page — for a parallel printer, use the **lpset** command. Enter:

```
# lpset -nlpnn [-iindent] [-ccolumns] [-llines]
```

where:

- *lpnn* is the file name of the device file (/dev/lpnn) with which the parallel printer is associated.
- *indent* is the number of columns that the text is indented from the left edge of the page. The default is 4.
- *columns* is the number of columns in a line. The default is 132.
- *lines* is the number of lines on a page. The default is 66.

To make changes to the default configuration that will be in effect each time the system is booted, use a text editor to insert the **lpset** line into the /etc/rc file.

See your *CENTIX Operations Reference Manual* for more details on the **lpset** command.

The lp Spooler Log File

A log file, /usr/spool/lp/log, is kept by the lp spooler system. Each time that **lpsched** is started, the current /usr/spool/lp/log is renamed /usr/spool/lp/oldlog and a new log file is opened.

Each time that **lpsched** routes an lp request to an interface program, an entry is recorded in the log file. The entry contains:

- the logname of the user that made the request.
- the request id.
- the name of the logical printer at which the request is being printed.
- the date and time that printing first started.

Any error messages generated by the printer spooler are also recorded in /usr/spool/lp/log.

Administering the uucp Network

The uucp network is used to send and receive batch files between CENTIX systems. The systems communicate over telephone lines.

This section discusses using, configuring, and maintaining the uucp network. An example of a direct link network is given at the end of the section.

Using the uucp Network

This subsection covers the following:

- The uucp public directory
- The uucp convention for file, command, and user name
- Copying files between systems
- Remote execution of commands
- Limitations of uucp commands
- Querying and controlling uucp job and network statuses

The uucp Public Directory

The uucp public directory, /usr/spool/uucppublic, is a standard destination for uucp copies and output files. In some cases, it is the only legal destination directory. uucp commands accept the tilde-nuucp (~nuucp) sequence as an abbreviation for /usr/spool/uucppublic/.

Some uucp messages refer to the public directory as PUBDIR.

Move files from the public directory to a private directory as soon as possible. uucp purges the public directory of old files every day. On a very busy system the public directory may be purged more frequently.

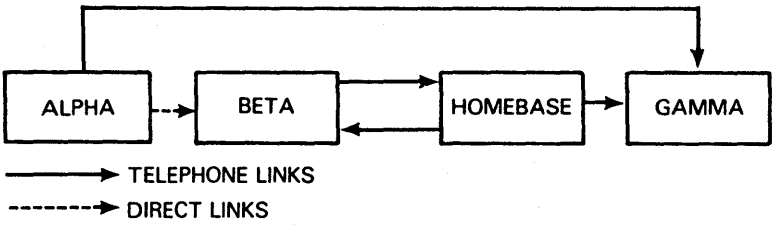
Conventions for File, Command, and User Names

A simple convention designates the computer system on which a file, command, or user is located. Using the convention requires that you know the node names of systems used to route your uucp jobs. Each system has a node name. To find the node name of a system, login to the system and type.

```
# uname -n
```

To specify a specific system, use a route of systems used to reach that system from your system. Elements of the route are separated by exclamation points (!).

Figure 10-1 **An Example of a uucp Network**



As an example, consider a simple circular network with four systems (see Figure 10-1). System alpha can communicate directly with system beta and system gamma; system beta with system alpha and system homebase; system homebase with system beta and system gamma; and system gamma with system homebase and system alpha. A user on homebase specifies alpha as beta!alpha or gamma!alpha, specifies beta as beta, and specifies gamma as gamma. A user on alpha specifies gamma as gamma or beta!homebase!gamma.

Normally the shortest route is the best way to specify a remote system, but this may depend on the hardware configuration of your network or cost of physical links such as dial-up or leased phone lines. Have users consult you for the best routings.

To specify a file, command, or user on a particular system, precede the name with the system specification and an exclamation point. Thus a user on homebase who wants /etc/passwd on alpha specifies beta!alpha!/etc/passwd.

A null system specification (for example, !/etc/passwd) or a missing system specification (/etc/passwd) specifies the local system. The **uux** command accepts null system specifications, but does not accept missing system specifications.

File names given to uucp are interpreted using file name conventions that extend those of the shell. The metacharacters asterisk (*), question mark (?), brackets ( ] and [ ) expand when used to specify systems just as they do when used to specify files to the shell. In addition, individual file names are interpreted by the following conventions:

- If the file name begins with a slant (/), uucp assumes that the name is a full path name and uses it as is.
- If the file name begins with a tilde followed by a slant and a username (~/username), the first two characters are a reference to the uucp public directory, /usr/spool/uucppublic and the username is a reference to a user's directory within the public directory. Thus beta!~/jon/docs means /usr/spool/uucppublic/jon/docs on the system beta. The users public directory is used when it is not possible to copy to a user directory (see the subsection below, "Limitations of uucp Commands").
- If the file name begins with a tilde (~) followed by a user name, the initial sequence references the specified user's home directory. For example, if jon's home directory is /a/jon, then jon/src/s.c is the same as /a/jon/src/s.c.
- If the file name does not begin with one of the three sequences given above, uucp assumes that the name is a partial path name and adds the working directory name (as printed by pwd), in front. Note that this convention includes files on remote systems, even though the working directory is a directory on the local system.

Copying Files Between Systems

There are two ways to copy files between systems:

- The **uucp** command is similar to the **cp** command (see **cp** in your *CENTIX Operations Reference Manual*). A copy requires a single **uucp** command issued anywhere in the uucp network. The **uucp** command can only access files accessible to the user "uucp".
- The **uuto** command sends files to a particular user. A user on the source system, who has access to the files, originates the copy, specifying a remote system and user. The specified user completes the copy.

Actual data transmission does not necessarily occur right away. Each copy must wait for the next establishment of a communication link (normally within 24 hours) or for other uucp jobs to finish with the link. There are two ways to prevent a file from being modified or deleted before it is transmitted:

- Specify that **uucp** immediately make its own copy of the file and transmit the copy (**-C** option of **uucp**; **-p** option of **uuto**). This method is simplest, but avoid it when copying large amounts of data at once.
- Use the uucp job status features to find out when the transmission actually takes place. See the subsection "Querying and Controlling uucp Job and Network Status."

The uucp Command

The **uucp** command takes the following form:

```
# uucp options filelist destination
```

where:

- *options* can be omitted. The following options are used by ordinary users (other options are primarily used by administrators):
  - **-C**: Copy specified files to spool directory and use copies for transmission.
  - **-m**: Send mail to sender when copy is complete
  - **-m***file*: Write message to file when transfer is complete.

- **-n***user*: Notify user on destination system, via mail, when copy is complete.
- **-j**: Print job number. This allows tracking and control of the uucp job; see the subsection "Querying and Controlling uucp Job and Network Status."

- *filelist* is a list of files to be copied, with list elements separated by spaces. Each file name can begin with a system specification. File names with *,?,and [...] metacharacters are expanded into multiple names on the specified system. Do not specify a directory without trailing file names or metacharacters; copying /x/dir/* is meaningful, but /x/dir is not.
- *destination* is the destination file for the copy. If *destination* is not a directory, *filelist* must specify exactly one name; *destination* is the name of the copy. If *destination* is a directory, *filelist* can specify any number of names; the files are copied into *destination* under the same local names.

The destination directory must be writable by the user uucp. Here are two ways to deal with this limitation:

- Copy files to the directory /usr/spool/uucppublic. uucp commands accept a tilde-nuucp sequence (~nuucp) as an abbreviation for that name. A user's public directory, /usr/spool/uucppublic/username, can be abbreviated with a tilde-slash followed by the user's login name (~/username).
- Provide a directory useable by everyone as the destination directory. Enter:

```
# chmod a+wrx directory
```

  where *directory* is the destination directory.

If you attempt to copy to a directory not writable by the user uucp, the uucp system attempts to salvage the copy by copying the files to the public directory. In this case, the copied files end up in a directory called /usr/spool/uucppublic/user, where user is the originating user's name.

The following examples are from the network described in Figure 10-1. The first example is run on homebase; it copies phonelist in the user's working directory to /usr/spool/uucppublic/phonelist.hb on alpha.

```
# uucp phonelist beta!alpha!~nuucp/phonelist.hb
```

The next example, run on homebase, copies files from homebase (a/jon/q.h and /a/jon/q.c) and from gamma (the entire contents of the directory /a/bill/dd) to the directory on beta (/b/sal/copies):

```
# uucp /a/jon/q.[hc] gamma!/abill/dd/* beta!/b/sal/copies
```

The uuto and uupick Commands

The **uuto** command originates the transmission of files. It takes the following form:

```
# uuto options filelist recipient
```

where:

- *options* can be omitted. The following options are understood:
  - **-p**: Copy the files to the uucp spooling directory and use the copies for transmission.
  - **-m**: Send mail to the sender when the copy is complete.
- *filelist* is a list of files to be sent; elements of the list are separated by spaces. The files must be on the local system; *,?, and [ ] metacharacters are expanded on the local system. Only ordinary shell conventions are understood; filelist must not contain a system designation or a tilde (~) directory designation.

  The **uuto** command interprets a directory name in filelist more usefully than does **uucp**. If a directory is specified, the entire hierarchy under the directory is copied.
- *recipient* specifies the user to whom the files are sent. The user name normally begins with a system specification.

Note that **uuto** provides no option that prints uucp system job numbers. To get job numbers for uuto requests, see the subsection "Querying and Controlling uucp Job and Network Status."

The **uuto** command uses the **uucp** command to actually send the files. If directories are specified, **uuto** may call **uucp** more than once, resulting in multiple uucp system jobs.

The following example is run on the system homebase in the network example shown in Figure 10-1. The example sends the files red and white in the working directory to user sal on system beta.

```
# uuto red white beta!sal
```

The **uupick** command picks up files sent by **uuto**. It must run on the receiving system. The command takes the following form:

```
# uupick -s system
```

where **-s** *system* can be omitted. If specified, only files from system can be picked up; otherwise, all files can be picked up.

The **uupick** command queries for an action on each file waiting to be picked up. The command then prints the file's origin (whether the file is an ordinary file or a directory) and the name of the file, then prompts for action. An empty input line means no action just now; other input lines are the following:

d	Delete the file without copying it.
m	Move the file into the current directory.
m *dir*	Move the file into the directory *dir*.
a	Move the file and all other files from the same system into the current directory.
a *dir*	Move the file and all other files from the same system into the directory *dir*.
p	Print the contents of the file.
q	Terminate **uupick**.

An end of file condition also terminates **uupick**. An unrecognized command prints a command summary.

Remote Command Execution

The **uux** command provides remote execution of shell commands. It also permits local execution of commands with remote file arguments. The **uux** command accepts simple commands and pipelines. All commands in a pipeline must execute on a single system.

The **uux** command is limited by security requirements; it can only execute commands specifically allowed by the executing system. Which commands are allowed are controlled by the file /usr/lib/uucp/l.cmds.

The **uux** command uses mail to notify the user when the command has executed. The command has this form:

```
uux options command
```

where:

- *option* can be omitted. The following options are understood:
  - **-**: Read the standard input of **uux** and provide it as the standard input of the command executed.
  - **-n**: Do not notify the user when the command has executed.
  - **-m***file*: When command has executed, write message to *file*. Do not send mail.
  - **-j**: Print uucp job number.
- *command* is one or more parameters that make up the command to be executed. Different parts of the command can be separated by spaces within the parameters or by being put in separate parameters. The **uux** command understands the same metacharacters as does the shell, in addition to uucp conventions for command and file names. Only the first program, however, in a **uux** pipeline can specify a system. The other programs are automatically executed on the same system. The shell metacharacters that are to be interpreted by **uux** must be quoted to prevent interpretation by the shell.

Limitations of uucp Commands

uucp has limitations that prevent unauthorized use of a remote system. There are four basic limitations:

- Limitations of the user uucp
- Parent directory names
- Forwarding from private directories
- Administrative restrictions on access

The User "uucp"

All uucp actions are taken by the user uucp, a user without any special or privileged status. Thus, if you prevent other users on your local system from using a file in a certain way (reading, writing, deleting), you are placing the same restriction on users on remote systems.

If the user "uucp" cannot read a file, you cannot use **uucp** to copy it, even if you own it. If a **uucp** command refuses to read a file, make sure that the file is readable by everyone. If the file is readable, check the read and search (execute) permissions of all the directories it is under. For example, suppose the uucp system cannot copy /a/jon/doc/bills. The following commands check all pertinent permissions:

```
ls -l  /a/jon/doc/bills

ls -ld  /a/jon/doc

ls -ld  /a/jon

ls -ld  /a

ls -ld  /
```

For information on file location and permission, see **chmod**, **ls**, and **pwd** in the *CENTIX Operations Guide*.

Parent Directory Names

Names that include parent directory references (for example, /usr/../xyzzy) are not permitted, because it is too difficult for uucp to determine whether such a name violates other restrictions.

Forwarding Private Directories

An extra restriction is placed on file copies that involve forwarding (communicating with systems not directly connected to the local system). When using forwarding, the remote file must not be in a private directory: the remote name must begin with ~/ or /usr/spool/uucppublic/. This restriction applies to both **uucp** and **uux**. The **uuto** command is not affected because it uses the public directory anyway.

Administrative Restrictions on Access

As administrator, you control how your system is used by other systems on the uucp network. There are three kinds of controls:

- Controls on who can access which files in the local system. You impose these controls by adding restriction information to the uucp user file.
- Controls on execution of local commands. uucp will not execute a command unless you add the command to the uucp commands file.
- Forwarding restrictions. If you create an origin file or forwarding file, there are restrictions on who can use the local system for forwarding uucp jobs or restrictions on which systems can receive forwarded jobs.

Querying and Controlling uucp Job and Network Status

The **uustat** command allows you to follow and control the progress of a uucp job until your system transmits the job to the first system in its route. It also monitors the status of your system's direct communication with other systems.

A uucp job number is required to follow and control a specific job. The **-j** option of the **uucp** and **uux** commands print the uucp job number. To make job number printing automatic, set the environment variable JOBNO to ON. To restore manual control over the job number, set JOBNO to OFF.

The **uustat** command takes the following form:

```
# uustat options
```

where *options* is a list of option parameters. The following *options* are meant for ordinary users.

- **-j***job*: Report the status of *job*, which can be a job number, to indicate a specific job; all to indicate all recent jobs; or missing, to indicate all recent jobs by the current user.
- **-u***user*: Report the status of all jobs sent by *user*.
- **-s***sys*: Report the status of all jobs that communicate with the system *sys*.

  Job status reports give the job number, the name of the user who originated the job, the next system on the job's route, the time the job was queued, the last time the job's status changed, and a description of the job's current status.
- **-k***jobno*: Kill the job whose uucp job number is *jobno*.
- **-r***jobno*: Rejuvenate the job whose job number is *jobno*. A new or rejuvenated job is safe from uucp's maintenance program for a defined period (normally a week). A job that waits through the standard period without being transmitted or rejuvenated is automatically killed.
- **-m***sys*: Give the status of the communication link with the system sys. If *sys* is all, give status for all systems known to the local system. A status report consists of the machine name, the time the status last changed, and the current status.
- **-M***sys*: Like **-m**, but also gives the time of the last successful transmission.

Configuring a uucp Network

This section tells how to configure individual uucp communication links and how to configure a system that runs uucp. Also described are uucp configuration files and demons. Note, however, that this section is not a complete reference to uucp user and administrative commands. For complete details, refer to the following entries in your *CENTIX Operations Reference Manual*: **uuclean, uucp, uustat, uusub, uuto,** and **uux**.

Basic uucp Configuration Concepts

uucp uses the same communication methods used on interactive terminals. A communication link is established between two computer systems when one computer system calls another. The call is required even when the two systems are directly connected.

A communication link is controlled from one end: uucp only permits one of the two systems on a link to exercise the link. The other system must wait to be called. The only way to permit two systems to call each other at any time is to provide two links. An alternative to a double link is to have the controlling system call the waiting system at regular intervals (polling).

uucp supports two kinds of communication links:

- *Direct link.* The two systems are directly connected, normally with two RS-232 ports connected by a null modem cable.
- *Telephone link.* Each system has a modem. The system that exercises the link must have a smart modem (a modem that can dial up another modem) or an automatic call unit. (The kernel currently supports smart modems but not automatic call units.)

If a system uses multiple telephone links, it does not need a modem for each, although multiple modems increase the number of links that can be active at one time.

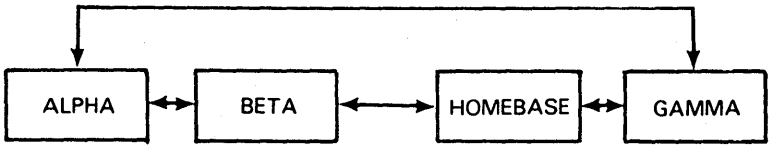
The examples in this section configure the network shown in Figure 10-2. This is the same network discussed in the previous section, but shown in more detail here. System alpha and system beta are at the same location; system homebase is at a different location in the same city; system gamma is in a different part of the country. There is one direct link: alpha to beta. There are four telephone links: alpha to gamma; beta to homebase; homebase to beta; and homebase to gamma. The direct link runs at 9600 baud, the telephone links at 1200 baud. Note that gamma controls no links; all uucp jobs that originate at gamma must wait for a call from alpha or homebase.

Actual data sending and receiving is done by the uucp copy-in/copy-out demon, **uucico**. (A demon is a program that normally runs in background). The **uucico** demon runs with its effective user ID set to **uucp**, implementing the uucp feature that all uucp commands are executed by that user. Each communication link requires an instance of **uucico** running on each system, in dialogue with the **uucico** on the other system. The **uucico** on the system that initiates communication is the master; the **uucico** on the called system is the slave.

The **uucico** command that is run as a master is the program that actually establishes a communication link. It runs in one of three modes, controlled by its **-s** parameter:

- *Clean up*: If the **-s** parameter is missing, **uucico** scans the uucp spooling directory. The **uucico** command calls each neighboring remote system that has jobs waiting on the local system. The commands, **uucp**, **uux**, and **uudemon.hr** (a maintenance demon that normally runs once an hour) run a master **uucico** without a **-s** parameter; **uucp** and **uux** do not wait for **uucico** to finish.
- *Send to a specific system*: If the **-s** parameter is present, and followed by the name of a system that uucp can call, **uucico** calls the system and does any jobs waiting for the link between the two systems to be activated.
- *Debug mode*: If you want messages that pertain to troubleshooting your configuration, **uucico** can be run in the debug mode. See the subsection "Testing with uucp" for details.

Figure 10-2 **uucp Configuration Example**



E7615

Configuring Communications Links

The basic uucp communications link starts with a hardware connection, either permanent, direct-connect, or by way of a modem (permanent or dial up). Once this is in place, the BTOS configuration files that configure the processor boards involved in the communications may need to be edited to support the modem, line driver or direct connection involved.

The software must be configured to give the necessary instructions to get communications under way—perhaps activating the auto dialer on a modem, or responding to the signal of an auto-answer modem.

Finally, the pathways and directories required for efficient use of the link by system users should be created, if they have not already been created for you. A public directory is usually created on the system as delivered. You determine such factors as when your system contacts other systems in the network, which systems you will recognize, and how long files in the public directory will be retained.

Use the following list of steps as a guide to configuring communications links with other systems in a uucp network:

1 Have your Burroughs representative install the necessary communications hardware to establish the physical link with the network.

2 Set the node name for your system. Find out the node names of the other systems on the network.

3 Create an entry in the modemcap file, if one does not already exist, for the smart modem you are using.

4 Make sure that **getty** does not monitor the caller's terminal interfaces.

5 Create an appropriate special file for the caller's lines.

6 Configure the caller's terminal interfaces for use by uucp.

7 Make sure that **getty** does monitor the called system's terminal interfaces.

8 Provide a user name on the called system for use by the caller.

9 Change the uucp system file on the caller to make calls to the other system; change the uucp system file on the called system to accept calls from the other system.

10 Configure each system to place necessary security restrictions on the other system.

Some of these steps need not be repeated for each new communication link:

- A node name is usually assigned only once. Only conflicts (such as when two networks are merged) require a new node name.
- A single terminal interface and smart modem can call more than one system.
- A single terminal interface and modem can receive calls from more than one system.
- A single user name can handle logins from more than one system.
- If the system file allows calls to be made to another system, it also allows calls from that system.

Hardware Installation

A direct link requires a null modem (cross connected) cable. Limited distance modems (line drivers) may be necessary for cables longer than fifty feet. A telephone link requires a smart modem on the calling system and a compatible autoanswer modem on the other system. See the installation instructions for your brand of modem.

See Section 4 of this manual for a discussion on how RS-232-C ports are numbered.

In following examples, we will assume the following hardware configuration. B & F is an imaginary manufacturer of autoanswer, autodial modems that run at 1200 baud.

alpha	cable to beta on tty001 B & F modem on tty002
beta	B & F modems on tty001 and tty002 cable to alpha on tty019
homebase	B & F modems on tty001
gamma	B & F modem on tty001

The examples also assume the following configuration for alpha and beta as defined in the system's cp00.cnf file, the inittab file, and the conrc file:

1 The BTOS CP00 configuration file. [sys]<sys>cp00.cnf, has RS-232-C Channels 2 and 3 set up for modem operation (i.e., there are entries for "async2" and "async3" whose parameters are defined for modem operation). The modems on these two channels are assigned the terminal names tty001 (Channel 2) and tty002 (Channel 3).

   Also, Channel 1 (entry "async1") is set up to support an ASCII dumb terminal. This dumb terminal serves as the console and is assigned as terminal tty000.

2 The inittab00 file contains entries for tty000, tty001, tty002, and tty019.

3 The conrc file shows the actual console as being tty000.

Assigning a Node Name

The node name is the system name that appears in all uucp system designations. The setuname command sets a system's node name:

```
setuname -n name
```

where *name* is the node name.

Setuname must be executed each time that the system is rebooted. Arrange for this to happen automatically by adding the command to one of the system start-up scripts. Put the command in the general purpose startup script, /etc/allrc, so that each AP will execute setuname.

Note that each AP must execute setuname: entering the command manually only sets a single AP. If your system has more than one AP, you can reboot, causing every AP to execute /etc/allrc, or you can use **spawn** (see **spawn** in your *CENTIX Operations Reference Manual*).

You can use either of the following commands to verify that the node name has been correctly set:

```
# uname -n
# uuname -l
```

In the following example, the administrator of homebase sets the node name and provides for future automatic setting. The administrator's input is bold, the system's responses are plain.

```
# setuname -n homebase
# ex /etc/rc
"/etc/rc", 33 lines, 608 characters
:/node name
:a
setuname -n homebase
.
:xit
"/etc/rc", 34 lines, 629 characters
#
```

Smart Modems and the modemcap File

The smart modems that uucp can use are described in the text file /usr/lib/uucp/modemcap. Each modem has several short names, listed at the beginning of its description. This file is documented below. New modems are defined by editing the file.

The modemcap file describes the call placing protocol of smart modems. The commands **uucp**, **cu** and **dial** accept a reference to a modemcap entry in place of an automatic call unit reference in /usr/lib/uucp/L-devices. Each entry describes a single modem in a specific configuration. Labels may not begin with the letters "cua".

The modemcap file is a text file. Lines that begin with a pound sign (#) are considered comments, and ignored. Each entry has a top line that consists of one or more labels (separated by vertical lines (|), and one or more other lines that contain capability/value pairs. A typical entry has the form:

```
label|altlabel
        :numcap#numval:strcap=strval:\
        :strcap=strval:   . . . :\
        :pl=value
```

where:

- *label*—name to be used in place of the call unit name in /usr/lib/uucp/L-devices
- *altlabel*—one or more alternate names that can be used in place of the call unit name, separated from label and each other by vertical bar (|).
- *numcap*—two character name of a numerical capability
- *numval*—the value of a numerical capability
- *strcap*—two character name of string capability
- *strval*—string value of a string capability
- *pl*—defines the control program for this entry

A numerical capability is separated from its numerical value by a pound sign (#). A string capability is separated from a string value by an equal sign (=).

Capabilities are separated from each other by colons. To extend a modemcap entry to another line, end the line with a colon (:), backslash (\) and a newline.

In a string capability the following sequences stand for the single characters noted.

xxx	(where *xxx* is one to three octal digits) the character whose octal value is *xxx*
\072	colon (:)
\200	null (do not use \000)
\E	escape (\033)
\n	newline (\012)
\r	return (\15)
\t	tab (\015)
\bl	backspace (\010)
\f	formfeed (\014)
^*x*	control-*x*

There are fifteen types of capabilities. Capabilities ending in *x* may have up to 36 instances, the *x* being a symbolic representation of any digit or lowercase letter. For example, a*x* can have an a1 instance, an a2 instance, an aq instance, etc.

a*x*=value	terminate call if comparison flag is set to EQUAL. Return value for debugging purposes.
b*x*=value	terminate call if comparison flag is set to NOT EQUAL. Return value for debugging purposes.
c*x*=value	compare value with wisk buffer (see w*x*, below) and set comparison flag to EQUAL if they are equal and NOT EQUAL if they are not.

d*x*#value	delay for the number of seconds specified in value.
eh=value	string marking the end of a phone number.
m*x*#value	skip the number of instructions specified in value, if comparison flag is set to EQUAL.
n*x*#value	skip the number of instructions specified in value, if comparison flag is set to NOT EQUAL.
pa=value	string to embed in a phone number to cause the modem to pause during dialling.
pe=value	string marking the end of a primary modem command
ph=value	send the modem a phone number. Signal the beginning of the phone number with value. Signal the end of the phone number with value of eh.
pl=value	place a phone call. The value of this capability is a string composed of names of other capabilities.
ps=value	string marking the start of a modem command.
pw=value	string to embed in phone number to cause the modem to pause during dialling and wait for another dial tone.
s*x*=value	send command *x* to the modem. Start the command with the value of ps if defined, then send value of s*x*, then the value of pe if defined.
t*x*=value	send value to modem.
w*x*=value	wisk through input from modem until value is read. Put input in wisk buffer, replacing previous contents.

Capabilities can be roughly divided into four classes:

- Placing the call
- Setting basic modem features
- Sending a phone number
- Other send/receive capabilities

Placing the Call

There is only one capability in this category. It is the pl capability. There is only one pl capability per entry. It is the master control capability for its modemcap entry.

A communications program using the modemcap entry works through the string value of pl, which is composed almost entirely of the names of other capabilites. The communications program uses each capability as it is encountered. The *cx* compare capability as well as the *mx* (skip on EQUAL) and *nx* (skip on NOT EQUAL) capabilities can be used within this value to form it into a kind of program with limited control of execution flow.

Setting Basic Features

The capabilities in this category specify strings used to activate basic features of a modem. These values never appear in the value of pl, but are implied by other capabilities.

ps — Some modems require that commands be preceded by a character sequence. This capability specifies the characters required, if any, to precede modem commands, separating them from data. The value is used when the *sx* capability is encountered in pl. See *sx*, below, under "Other Send/Receive Capabilities."

pe — This capability specifies the characters required, if any, to mark the end of a modem command. See *sx*, below, under "Other Send/Receive Capabilities."

eh — Some modems require that the end of a phone number be marked by a character sequence. This capability specifies that sequence. The value is used when the ph capability is encountered in pl.

pa — With some older telephone systems, a pause is required during the dialing of a number. When the string value specified is encountered in a phone number being sent to the modem, the modem is signalled to stop dialing momentarily.

pw — Some telephone systems require that the caller dial an introductory number, wait for a second dial tone, and then dial another number. When the string value specified here is encountered in a phone number, the modem is signalled to stop dialing and wait for another dial tone before proceding.

Sending a Phone Number

ph When this capability is encountered in pl, the communications program uses a single invocation of the **write** system call to send to the modem a string that has three parts. The three parts of the string are:

The value specified for ph. This should be the character sequence required by the modem, if any is required; that signals the beginning of a phone number.

The phone number has ASCII digits and character sequences. When the character sequence defined in the pa capability is encountered, the modem pauses; whenever the character sequence defined in the pw capability is encountered, the modem pauses, and waits for another dial tone before continuing to dial.

The value specified for eh, if defined.

Other Send/Receive Capabilities

These capabilities are different from the others in their naming convention. The first character of the capability is its name. The second character may be any digit or lowercase character and identifies the instance of the capability.

t*x* Send value to modem.

s*x* Send a command to the modem. The command has three parts and is sent in a single invocation of the write system call. The three parts are:

The value defined by ps, if any. This should let the modem know that what follows is a command, and not data.

The value specified for the s*x* capability. This should be the modem command itself.

The value defined by eh, if any. This should let the modem know that the end of the command has been sent.

d*x*	Delay for the number of seconds specified in value.
w*x*	Value must be a single character. When this capability is encountered the communication program wisks the input from the modem into the buffer up to but not including the specified character. Any previous contents of the wisk buffer are overwritten.
c*x*	Compare the value specified with the contents of the wisk buffer. (See w*x* above.) If the contents of the wisk buffer and the value specified are equal, then set the comparison flag to EQUAL. If the contents of the wisk buffer and the value specified are not equal, then set the comparison flag to NOT EQUAL.
m*x*	Skip a number of capabilities that follow in the pl control program. The number of capabilities to skip is specified by value. The capabilities are skipped only if the comparison flag is set to EQUAL.
n*x*	Skip a number of capabilities that follow in the pl control program. The number of capabilities to skip is specified by value. The capabilities are skipped only if the comparison flag is set to NOT EQUAL.
a*x*	Terminate the phone call if the comparison flag is set to EQUAL.
b*x*	Terminate the phone call if the comparison flag is set to NOT EQUAL.

In following examples, the name for the imaginary B & F smart modem is assumed to be bf.

A hypothetical modemcap entry for the B & F smart modem is shown below.

```
#B & F smart modem - option switch 9 down
bf|B and F|B and F Smart Modem
    :a1=NO ANSWER:b1=NO DIAL TONE:b2=NO ANSWER:c1=1:c2=2:\
    :c7=7:d1#1:d5#5:eh=\r:ph=\02D:ps=\02:pw=\072:\
    :sa=A:sq=Q:sv=V:sx=X:sz=Z:wp=\r:\
    :pl=szd5wpd1svwpsqwpsxwpd1phwpc7b1wpc2a1c1b2d1
```

The modem in this example has the following characteristics. With its switch number nine in the down position, commands to the modem must be preceded by an ASCII start of text (STX) character(\002), and followed by a carriage return (\r). Thus, on line four of the modemcap entry the value the ps capability is set to \002 and the value of the eh capability is set to \r.

The modem sends messages to the computer as diagnostics. These take the form of a digit followed by a carriage return.

1 carriage return	means a connection has been made.
2 carriage return	means no connection has been made or there was no answer.
7 carriage return	means that a dial tone has been detected.

Look at the value for pl, the place-a-call master control program, in the sample modemcap entry above. An analysis of this entry, a few characeristics at a time, is presented below. The characteristics are analyzed in the order in which they appear in the value of pl, that is, in the order in which the communication program would encounter them and act upon them.

sz	Send the letter Z (reset a B & F modem) as a command. That is, send the letter Z preceded by an STX character and followed by a carriage return. The sz capability is the z instance of the *sx* (send) capability. It is defined in the next to last line of the sample modemcap entry.
d5wpd1	Wait five seconds (d5). Wait for a carriage return (wp). The wp capability is the p instance of the *wx* (wisk) capability. Wait another second (d1).
svwpsqwpsxwpd1	Send the letter V (select tone dialling) as a command (sv). Wait for a carriage return (wp). Send a letter Q (toggle busy detection) as a command (sq). Wait for a carriage return (wp). Wait a second (d1).
ph	Send a phone number. Remember, the ph capability precedes the phone number with its own value (STX-D, this is set on line four of the sample modemcap file). The phone number is followed by the value set for the eh capability, in this case a carriage return (See line four of the sample modemcap file.).

wpc7b1	Wait for a carriage return (wp). If the modem does not return a seven (c7), terminate the call and return the error message "NO DIAL TONE" (b1). Remember, this modem returns a "7" if it detects a dial tone.
wpc2a1c1b2	Wait for a carriage return (wp). If the modem returns a "2" (c2), terminate the call and display message a1, "NO ANSWER". If the modem does not return a "1" terminate the call and display the message b2, "NO ANSWER".
d1	Wait a second.

Configuring getty on the Calling System

The **getty** program monitors terminal lines for attempted logins; a separate **getty** process monitors each line. The **getty** program must not monitor lines used for calling other systems.

The **who** command tells you which lines **getty** monitors. Enter one of these command lines:

```
# who -a
# who -a | fgrep ttyxxx
```

where *xxx* is the three-digit terminal number. The second form of the command restricts output to the line you are interested in. If **getty** monitors the line, (**who** prints a status for the line), you are interested in the status's first field (who or what is using the line) the second field (the terminal number) and the fifth field (the numeric ID of the process controlling the line).

To get **getty** off of the line, follow these steps:

1 If no status is shown, getty is not monitoring the line: skip the remaining steps. If the first field of the user status is "LOGIN", no one is using the line. Go to the next step. If the first field in the **who** status is a user name, that user is using the line. Have the user log off and get another **who** status.

2 Edit /etc/inittab00 and comment out the terminal configuration entries for the line (there are normally two entries). To comment out a line, insert a colon (:) at the beginning of the line. See Section 3 of this manual for the format of this file.

3 Have the init command reread /etc/inittab. Enter:

```
# init q
```

Warning: *Use the* ***init*** *command carefully and precisely. Using the wrong parameter can shut down the system, corrupting the file system, and possibly the system software.*

---

4 Make sure that the last instance of **getty** dies. Enter:

```
# kill x
```

where *x* is the process number given the last time **who** was invoked. The **kill** command will fail if init has already killed getty for you.

5 Repeat the **who** command to verify that you did the last three steps correctly.

In following examples, the systems from the previous examples use the following lines for calling other systems:

alpha	**tty001 (direct line to beta)** **tty002 (smart modem)**
beta	**tty001 (smart modem)**
homebase	**tty001 (smart modem)**

In the next example, the homebase removes the getty from tty001. Your input is shown in bold. The responses from the computer are shown in normal print.

```
# who -l | fgrep tty001
LOGIN   tty001   Apr 12 11:20   old   5309
# ex  /etc/inittab
"/etc/inittab" 54 lines, 2144 characters
:/001
001:2:respawn:/etc/getty tty001 9600
:substitute/~/#
:001:2:respawn:/etc/getty tty001 9600
:+
C001:6:respawn:/etc/getty tty001 C9600
:substitute/^/#/
:C001:6:respawn:/etc/getty tty001 C9600
:xit
"/etc/inittab" 54 lines, 2146 characters
# init q
# kill 5309
kill:  5309:  no such process
# who -l | fgrep tty001
#
```

Creating a Special File

New special files for calling lines are not strictly necessary, but avoid mistakes when using the lines. Accessing the wrong modem or direct line is apt to be less drastic a mistake than accessing the wrong terminal line. If you use an existing special file, make sure the file is executable.

Use the **ln** and **chown** commands to create a special file used to call over a direct line.

```
# ln /dev/ttyxxx /dev/d.name
# chown uucp /dev/d.name
```

where:

- *xxx* is the terminal interface's three-digit line number.
- *name* is the name of the system that the direct line calls.

Use the **ln** and **chown** commands to create a special file used to call over telephone lines:

```
# ln /dev/ttyxxx /dev/cuay
# chown uucp /dev/cuay
```

where:

- *xxx* is the terminal interface's line number
- *y* is a sequence number that identifies the particular outgoing phone line.

The last part of the special file name (d.name or cuay) is the device name. The device name is used in uucp configuration files and by interactive communications programs such as cu.

The following command verifies that a terminal line special file is equivalent to special file.

```
# ls -li /dev/ttyxxx /dev/devname
```

where:

- *xxx* is the terminal interface's line number.
- *devname* is the corresponding device name.

The **ls** command prints out two file status lines, each of which begins with an i number. The two i numbers should be the same.

In the following examples, the administrators on alpha, beta, and homebase create calling special files. The first example shows the command lines the administrator of alpha would use.

```
# ln /dev/tty001 /dev/d.beta
# ln /dev/tty002 /dev/cua0
# chown uucp /dev/d.beta /dev/cua0
```

The next example is for beta and for homebase:

```
# ln /dev/tty001 /dev/cua0
# chown uucp /dev/cua0
```

Note that this configuration is not necessary on the system being called because to that system the lines used by uucp are just login terminals.

Configuring the Caller's Terminal Interface

The uucp lines file, /usr/lib/uucp/L-devices, configures lines used to call out. This test file has two kinds of lines: comments, which are ignored, and line configurations. A comment begins with a pound sign (#). A line configuration describes a single line and is a text line of the form.

type name calldev speed protocol

where:

- *type* is the line type: DIR for direct lines, ACU for phone lines (both smart modem and automatic call unit; ACU is currently not supported)
- *name* is the device name (d.name or cuay).
- *calldev* is the calling device. For automatic call units, give the call unit's device name. For smart modems, give the smart modem name, as listed in the modemcap file. For direct lines, put any nonblank text to hold the place.
- *speed* is the normal baud rate of the line.
- *protocol* is the communication protocol. This field is usually empty.

The following examples configure the calling lines on alpha, beta, and homebase, using "bf" modems. First alpha:

```
# ex  /usr/lib/uucp/L-devices
"/usr/lib/uucp/L-devices" [Read only] 6 lines, 78 characters
:append
DIR d.bets 0 9600
ACU cua0 bf 1200
.
:xit
"/usr/lib/uucp/L-devices" File is read only
:xit!
"/usr/lib/uucp/L-devices" 8 lines, 116 characters
#
```

The "Read only" message indicates that the file lacks write permission. Thus, although **uucp** owns the file, only the superuser can modify it. Now for beta:

```
# ex /usr/lib/uucp/L-devices
"/usr/lib/uucp/L-devices" [Read only] 6 lines, 78 characters
:append
ACU cua0 bf 1200
.
:xit
"/usr/lib/uucp/L-devices" 7 lines, 100 characters
#
```

Homebase is the same as beta.

Configuring getty on the Called System

Lines for receiving uucp calls are configured precisely like lines for receiving users. A telephone dial up line can, in fact, serve both uucp and ordinary user logins. (Direct lines work both ways too, but the **cu** program is required to get access to them). To configure login lines, see Section 3 of this manual.

uucp User Names

A system that wants to accept calls from other systems must provide a user whose home directory is /usr/spool/uucppublic and whose shell is /usr/lib/uucp/uucico (the copy in/copy out demon). The called system must also mention the user name in the uucp user file.

Your system is distributed with such a user, named nuucp. Use the **passwd** command to set password for nuucp (see Section 6 of this manual). The system will work with nuucp as the only special user, but additional uucp user names provide security features:

- You can change the password for one uucp user, locking out one group of systems without affecting communications with other (presumably more trustworthy) systems.
- uucp allows you to impose file copying restrictions in addition to those imposed by regular file permissions. These restrictions divide neighboring systems into classes according to which name they use to log into the local system.

See your *centrEASE Operations Reference Manual* for information on how to create new user names.

If the nuucp user is not already in the user file, /usr/lib/uucp/USERFILE, the following line in the file properly mentions it without imposing any special restrictions:

```
nuucp, /
```

The above entry in the user file allows any remote system to log in and access any file not protected from the user uucp. To impose restrictions, see the subsection "Maintaining a uucp Network" below.

In the network example shown in Figure 10-2, the four systems modify their password and user files as follows:

- The alpha system lacks any facilities for remote logins. On the presumption that logins by other systems must be unauthorized, alpha's administrator edits nuucp's entry in /etc/passwd:

  ```
  nuucp:%::6:1:/usr/spool/uucppublic:/usr/lib/uucp/uucico
  ```

  The invalid second field renders logins by other systems impossible. Since alpha does not permit other systems to log in, no change is necessary to /usr/lib/uucp/USERFILE.

- On the beta system, the administrator decides that other systems at the same site will login as nuucp, but offsite (and potentially untrustworthy) systems will login as ouucp. Therefore, he or she edits the following lines into /etc/passwd:

  ```
  nuucp:::6:1:/usr/spool/uucppublic:/usr/lib/uucp/uucico

  ouucp:::6:1:/usr/spool/uucppublic:/usr/lib/uucp/uucico
  ```

  The administrator then uses the **passwd** command to assign the password "ourgang" to nuucp and the password "xyzzy" to ouucp. He or she tells alpha's administrator the nuucp password and homebase's administrator the ouucp passwd.

  The administrator edits the file named /usr/lib/uucp/USERFILE to contain the following lines:

  ```
  nuucp, /

  ouucp, /
  ```

- The homebase system currently wants to accept remote system logins by beta. Alpha or gamma may want access later, but if they do they will have the same status as beta. Homebase already has the user nuucp; the administrator gives nuucp the password "greekaccess".

- The gamma system treats all of the other systems the same. The administrator gives nuucp the password "compo" and adds the standard line to /usr/lib/uucp/USERFILE:

  ```
  nuucp, /
  ```

The uucp System File

The uucp system file, /usr/lib/uucp/L.sys, tells which systems neighbor the local system. If the local system can call the other system, the system file provides one or more procedures for making the call.

Each line in the system file is either a comment or a system line. A comment line begins with a pound sign (#) and is ignored. Each neighboring system requires one or more system lines. The system line has one of two forms. The first form permits the other system to call the local system:

```
name
```

where *name* is the node name of the other system.

The second form allows the local system to call the other system and the other system to call the local system. Do not provide both forms for one system. If there is more than one procedure to call the remote system (as with a system that has more than one phone number), provide one system line for each such procedure; the different procedures will be tried in the order listed. Here is the second form:

```
name when device speed number login
```

where:

- *name* is the node name of the other system.
- *when* specifies permitted calling times and minimum retry period. The simplest value is "Any," which places no restrictions. "When" takes the following form (time and retry are optional; omit the comma if you omit retry):

  ```
  dayshours,retry
  ```

  where:

  days is a concatenation of abbreviations indicating the days on which the procedure line can be used. Abbreviations are Su, Mo, Tu, We, Th, Fr, and Sa for specific days, Wk for any weekday, and Any for any day of the week.

hours specifies the time of day when the procedure can be used. If hours is omitted, the procedure can be used any time of day. The period is specified by two 24-hour clock times separated by a dash (for example: 0900-1700 for 9 am to 5 pm). The period can extend through midnight.

retry specifies the minimum waiting period, in minutes, after an unsuccessful call. If retry is omitted, the minimum waiting period is 30 minutes. The retry period applies if the last call to name was unsuccessful; the procedure cannot be used unless the minimum waiting period has expired. This feature prevents uucp from continuously trying to access a system that may be unavailable.

Here are some examples for *when*: Any0800-0700 means any time except 7 am to 8 am; SaSu means any time from 1201 am Saturday to midnight Sunday; Any, 10 means any time but not if the last try was unsuccessful and less than 10 minutes ago.

- *device*. For telephone links, this field is ACU: uucp will search the devices file for a dial-out device with the same speed. For direct links, this field must be the device name for the line.
- *speed* is the speed used. Must match the speed in the devices file.
- *number* is the phone number. The characters have the following meaning:

0-9	dial 0-9
* or :	dial *
# or ;	dial #
-	delay four seconds
w or =	wait for next dial tone
f	flash off hook for one second

On direct line, number is the same as the value specified for device.

- *login* defines the login procedure. Login consists of a list of expect/send sequences:

```
expect send expect send
```

The last item can be either an expect or send. The local system waits for the remote system to print a line containing the first expect. When it arrives, the local system sends a line consisting of the next send and waits for the next expect; and so on. When the local system sends or reads the last item, the remote system should have let it log in and be running a slave uucico.

An expect can include a subprocedure used if the remote system does not print the desired string before a certain period of time. In this case the expect consists of subexpect/subsend sequences:

```
subexpect-subsend-subexpect-subsend
```

If the remote system prints a line containing the first subexpect, the rest of the expect is skipped. If subexpect does not appear before a certain period of time, the local system sends the next subsend, waits for the next subexpect, and so on.

Here is a sample login entry:

```
ogin--ogin--ogin-- nuucp assword xyzzy
```

This login logs into a remote system as nuucp, and tries the password xyzzy. If the master **uucico** cannot get a login prompt at first, it tries twice to provoke one by sending an empty line.

Here are some system file examples. The first example is alpha's system file, which allows it to call (and be called by) gamma and beta:

```
gamma Any ACU 1200 408-5558328 ogin--ogin--ogin
nuucp assword compo

beta Any d.beta 9600 d.beta 9600
d.beta ogin--ogin--ogin-- nuucp assword ourgang
```

The next example is beta's system file, which allows it to call homebase and accept calls from alpha and homebase:

```
alpha

homebase Any ACU 1200 5550101
ogin--ogin--ogin-- nuucp assword ourgang
```

The following example is homebase's system file. It provides procedures for homebase to call beta and gamma. Calls to gamma are restricted to night hours and weekends to save long-distance costs.

```
beta Any ACU 1200 9w5559393 ogin--ogin--ogin--
ouucp assword xyzzy

gamma Wk2200 ACU 1200 9w408-5558328
ogin--ogin--ogin-- nuucp assword compo

gamma SaSu ACU 1200 9w408-5558328
ogin--ogin--ogin-- nuucp assword compo
```

Finally, we have gamma's system file, which permits calls from alpha and homebase:

```
alpha

homebase
```

Testing the Link

You can test a link using the **cu** program or using uucp itself.

Testing with cu

To test a direct link with **cu**, run the following command on the calling system:

```
# cu -l device
```

where device is the device name for the link. Attempt to log in and out of the remote system. Enter a line beginning with tilde-period (~.) when you are done.

The following command tests a telephone link:

```
# cu -l device number
```

where:

- *device* is the device name for the link.
- *number* is the telephone number of the remote system.

If this version of cu does not work, try it without the phone number; you will have to enter modem commands directly. As with the direct link, attempt to log in and out of the remote system. Enter a line beginning with tilde-period (~.) when you are done.

Testing with uucp

Follow these steps to test a link:

1 Execute a **uucp** transfer between the two systems, using the **-r** option to suppress the **uucico** demon. For example:

```
# uucp -r shortfile gamma!~/
```

2 Run the **uucico** demon, as a master, in debug mode, and in foreground. If one system must call the other, be sure to use that system.

```
# /usr/lib/uucp/uucico -r1 -x4 -sname
```

where *name* is the name of the system to be called.

The demon's debug output contains many messages meant to debug the demon itself and are not documented. But messages relating to your configuration are self-explanatory.

To find out how normal demons have done, check the log files /usr/spool/uucp/LOGFILE (today's activity) and /usr/spool/uucp/Log-WEEK.

Maintaining a uucp Network

This section discusses day to day maintenance. There are three kinds of maintenance:

- automatic maintenance
- security configuration
- emergencies

Automatic Demons

Automatic maintenance is performed by uucp background programs (demons). Certain chores are standard and are performed by standard uucp demons, which must be configured. Nonstandard demons poll slave systems.

Standard Demons

Each system running uucp should regularly execute the following standard demons: uudemon.hr, at four minutes before each hour: uudemon.day, at 4 am each day; and uudemon.wk, at 5:30 am every Sunday. To arrange this, use a text editor to remove the pound signs (#) that precede these lines in the /usr/lib/crontabs/uucp file:

```
0 4 * * * /usr/lib/uucp/uudemon.day
56 * * * * /usr/lib/uucp/uudemon.hr
30 5 * * 0 /usr/lib/uucp/uudemon.wk
```

This is what the standard demons do:

- uudemon.hr:
  - runs a master **uucico** without the **-s** option to attempt completion of uucp jobs waiting on the local system
  - uses **uulog** to merge recent status reports into the log file.
- uudemon.day:
  - uses **uuclean** to kill all uucp jobs waiting on the local system that are at least 168 hours old.
  - uses **uuclean** to kill all uux commands trying to execute on the local system that are 72 hours old.
  - moves the contents of today's log to the end of the weekly log.
  - uses **uusub** to call all systems this system knows how to call and to gather traffic statistics for the last 24 hours.
  - uses **find** to remove all ordinary files from the public directory, /usr/spool/uucppublic, that are more than 30 days old.
- uudemon.wk deletes weekly logs that are two weeks old.

Because the demons are shell scripts, you can edit them to add, change, or delete features. Systems that cannot afford the amount of space used by /usr/spool/uucp and /usr/spool/uucppublic usually shorten the time period parameters for the **uuclean** and **find** commands in uudemon.day.

Polling Demons

If a system must wait for all calls, you may want to have other systems poll it. Use the **cron** and **uusub** commands to set this up. (See your *CENTIX Operations Reference Manual*.)

The following example is a line for /usr/lib/crontabs/uucp. It attempts to poll system gamma every four hours. It is done one-half hour after the previous uudemon.hr to minimize the chance that these two demons will interfere with each other's use of the outgoing lines.

```
26 0,4,8,12,16,20 * * * /usr/lib/uucp/uusub -cgamma
```

Security Measures

You can place the following restrictions on uucp use:

- How specific remote users and systems can use your system.
- To what extent your system will forward files and commands from one remote system to another.
- Which commands will be executed by uucp.

Remote Access to the Local System

The uucp system has three kinds of restrictions on local file access:

- Files not accessible to the user uucp (a user without special status or privileges) are not accessible to the uucp system.
- Systems that access the local system through forwarding (that is, systems not directly connected to the local system) can only access files under the uucp public directory, /usr/spool/uucppublic.

- Additional restrictions imposed by the local administrator.

The first two kinds of restrictions cannot be modified without modifying uucp. The remainder of this section discusses the third kind.

The user file, /usr/lib/uucp/USERFILE, is a text file that controls the way users and systems use uucp to access the local system. It has four kinds of controls that apply to general classes of files and to specific users and systems:

- Which files local users can use uucp to copy.
- Which files remote systems can access.
- Which login name each remote system must use to talk to the local system.
- Whether a neighboring system must be called back to confirm its identity.

Controls are written into the user file with lines of the following form:

```
user, system callback prefixlist
```

where:

- *user* is a user name on the local system (either a real user or a user used for uucp logins) or a null string.
- *system* is the node name of a remote system or a null string.
- *callback* is the call-back flag, c, or a null string.
- *prefixlist* is a list of the initial parts of full path names. Elements of the list are separated by spaces.

uucp uses four rules to interpret the user file:

- When a remote **uucico** logs in, the local **uucico** searches the user file for an entry that allows the remote system to call. The user name in the entry must match the user name used by the remote **uucico**; the system name in the entry must be null or match the remote system's name. The first entry with such a match allows communication.
- If the entry matched by the previous rule has a call-back flag, that system must be called back.

- When a uucp job is stored on the local machine, uucp compares the name of the user responsible with the user names in the user file. The first name that matches yields a list of file name prefixes. If no name matches, the first line with a null user name yields a list of file name prefixes. In any case, the full name of each file to be accessed by the uucp job is compared with the list of prefixes. If the full path name does not begin with one of the prefixes, access to that file is denied.
- When a uucp job from a remote system is executed, uucp compares the name of the remote system with the system names in the user file. The first name that matches yields a list of file name prefixes. If no name matches, the first line with a null system name yields a list of file name prefixes. In any case, the full name of each file to be accessed by the uucp job is compared with the list of prefixes. If the full path name does not begin with one of the prefixes, access to that file is denied.

Here are some examples. If a user name is of the form xuucp, assume that it is a special uucp user (it has **uucico** as its shell). The first example allows any system to log in as nuucp and access any file:

```
nuucp, /
```

The next example allows system homebase to log in as ouucp and access any file whose full path name starts with /a/scott:

```
ouucp,homebase /a/scott
```

This example allows the local user bill to access files with uucp commands, but only if their full path names begin with /a/bill:

```
bill, /a/bill
```

The next example allows any remote machine to log in as nuucp and access files whose names begin with /usr/spool. In addition, alpha can access files whose full path names begin with /a/bill.

```
nuucp,alpha /usr/spool /a/bill

nuucp, /usr/spool
```

The last example allows any local user to access any file whose name begins with /usr/spool. It also allows the local user root to access any file at all.

```
root, /
, /usr/spool
```

Note that even root cannot access a file not accessible by the user uucp.

Forwarding

You can place two restrictions on uucp jobs forwarded through your system:

- Specify the systems your system will call in order to forward other systems' jobs.
- Specify the systems and users on whose behalf your system will forward jobs.

The system forward file, /usr/lib/uucp/FWDFILE, is a list of neighboring systems, one per line. If the system forward file exists, the next destination (not necessarily the final destination) of each job traveling through your system is checked; if the next system is not in your system forward file, the job is killed and the originator notified. This restriction does not apply to your own users. If the system forward file is absent, your system does not check the next destination of jobs forwarded through your system.

The system forward file is typically created to deny other systems access to an expensive communication link.

The origin file, /usr/lib/uucp/ORIGFILE, grants users on other systems permission to forward through your system. Each entry is a single line. If the entry is a node name, all users on that system can forward through your system. Users on the system whose node name is "system" can forward through your system only if their names appear in list, and the entry has this form:

```
system! list
```

The elements of list are separated by exclamation points (!).

A job allowed by the system forward file (or the absence of the system forward file) can be killed by the origin file, and vice versa.

Suppose that the administrator of system alpha decides that homebase users are specifying beta!alpha!gamma! instead of gamma! to avoid incurring costs to their own system. He or she creates /usr/lib/uucp/ORIGFILE with the following contents:

```
beta

gamma
```

The administrator of homebase makes a similar decision about alpha, beta, and gamma. To cut off all forwarding through homebase, he or she creates two empty files: /usr/lib/uucp/ORIGFILE, and /usr/lib/FWDFILE (although either would have been sufficient).

Permitted Commands

For uucp to execute a program other than its own demons, the command name for the program must be entered in the commands file, /usr/lib/uucp/L.cmds, one command name per line. Without such an entry, a command cannot be used within **uux**. If **rmail** (a restricted version of **mail**) is not mentioned in the commands file, local users cannot get mail notification of job outcomes or receive mail from users on other systems.

Add commands to the command file with care, since a sufficiently general command (such as **cat**) permits remote users to overcome your uucp security restrictions. A security-conscious system typically permits only **rmail**.

Handling uucp Emergencies

uucp creates most of its temporary files in /usr/spool/uucp. uucp users are prone to create many files in /usr/spool/uucppublic. Therefore, keep a close eye on the free space of the file system that holds /usr/spool: running out will paralyze uucp and possibly other parts of your system as well.

Note that if you restart uucp on your system after a period of time, jobs queued on neighboring systems, waiting to execute on or pass through your system, will arrive all at once. This can cause a new jam.

uucp and some other communications programs create lock files called /usr/spool/uucp/LCK *name*, where *name* is a device or remote system name. If the system or uucp crashes while a communication program was working, these files may remain, preventing you from restarting communication. Verify that the lock files do not belong to active uucico, cu, or other such program, then remove them. The safest time to remove a uucp lock file is when the system is in single-user mode. You may find it add a line using the **rm** command to remove lock files in one of the rc start-up scripts.

An Example of a Direct Link

This subsection describes the configuration of a simple direct link. The network consists of two XE 500 machines. Each system has a minimal communication hardware.

The node names chosen for the systems are "System2" and "System1". The link is controlled by System2 and runs at 9600 baud.

Each system has one CP and no TPs. They are linked by a null modem cable.

This example does not provide any security restrictions.

Configuring System1

The following communication configuration steps apply to System1.

1 Assign the node name.

2 Configure the terminal interface System2 will use to call System1.

3 Provide a user name which System2 will use to log in.

4 Create an entry for System2 in the uucp system file.

5 Specify System2's permissions for accessing System1.

The administrator executes the following command and adds it to /etc/allrc:

```
# setuname -n System1
```

The cable is connected to Channel 2 on the CP. The administrator uses Channel 1 or Channel 2 if the link is to run faster than 4800 baud. Since there is only one communication board, Channel 2 is terminal number 001, if a terminal has been configured for Channel 1.

To enable logins on 001, the administrator adds the following line to /etc/inittab00:

```
001:2:respawn:/etc/getty tty001 9600
```

Terminal 001 must not be active in Administrator Login Mode, enabled by the following line:

```
C001:6:respawn:/etc/getty tty001 /C9600
```

The administrator comments this line out:

```
:C001:6:respawn:/etc/getty tty001 C9600
```

If the administrator modified /etc/inittab00, he or she makes the modification effective by executing the following command on AP00:

```
# init q
```

The user name "nuucp" will serve for logins by System2. The administrator confirms that the password file, /etc/passwd, already defines nuucp with the following line:

```
nuucp:x:6:1::/usr/spool/uucppublic:/usr/lib/uucp/uucico
```

The second (password) field does not define a valid password, so the administrator uses the **passwd** command to specify one:

```
# passwd nuucp
```

The new password is "directalk".

The administrator adds the following line to the uucp system file, /usr/lib/uucp/L.sys:

```
System2
```

This network contains only two systems and security is not a problem. The administrator must verify that this line is in the uucp user file:

```
/usr/lib/uucp/USERFILE:./
```

Configuring System2

The following communication configuration steps apply to System2:

1 Assign the node name.

2 Make sure that getty does not monitor the line System2 uses to call System1.

3 Create a special file that uucp will use to access the line.

4 Configure the line.

5 Create an entry for System1 in the uucp system file.

6 Specify System1's permissions for accessing System2.

7 Have the link exercised at regular intervals.

The administrator executes the following command and adds it to /etc/rc:

```
# setuname -n System2
```

Configuring uucp

The line is connected to Channel B, chosen because the only other RS-232 channel is used by a user terminal. Channel B is terminal 001.

The getty command must not monitor 001, so 001 must not have any entries in /etc/inittab00. The administrator comments out the follow lines:

```
001:2:respawn:/etc/getty tty001 9600

C001:6:respawn:/etc/getty tty001 C9600
```

To make the changes effective, he or she enters:

```
# init q
```

The following commands create and check the special file used by uucp:

```
# ln /dev/tty001 /dev/d.System1

# chown uucp /dev/d.System1

# ls -li /dev/tty001
```

To specify communication configuration, the administrator adds the following line to the uucp devices file/usr/lib/uucp/L-devices:

```
DIR d.System1 x 9600
```

The administrator gives System2 the following entry in the uucp system file, /usr/lib/uucp/L.sys. Since the line is direct and costs nothing to use, there is no restriction on calling time and a short retry period.

```
System1 Any,5 d.System1 9600 d.System1
ogin--ogin--ogin nuucp password directtalk
```

The administrator verifies that the following line is in the uucp user file, /usr/lib/uucp/USERFILE:

```
./
```

The link is now established and can be used. To make sure that no job originating at System1 will wait more than thirty minutes, the administrator adds the following line to System2's cron table, /usr/lib/crontabs/uucp:

```
0,30 * * * * /usr/lib/uucp/uusub -cSystem1
```

Exercising the Link

The link will be exercised every 30 minutes plus every time a System2 user refers to System1 in a uucp copy or remote execution command. If System1 is down, then System2 will try to exercise the link no more than every 5 minutes. If an urgent uucp job is waiting on System1, any System2 user can initiate it by entering the following command:

```
# /usr/lib/uucp/uusub -cSystem1
```

Tracking System Activity

You can use the commands in the system activity package to track the following types of system activity:

- CPU utilization.
- Disk and tape I/O activities.
- Terminal device activity.
- Buffer usage.
- System calls.
- System switching and swapping.
- File-access activity.
- Queue activity.
- Message and semaphore activities.

The package includes commands that you can use to generate various types of reports. These are:

- The **sar** command, which allows a user to generate system activity reports in real time and to save system activities in a file for later usage.
- The **sadp** command, which samples disk activity once every second during a specified time interval and reports disk usage and seek distance in either tabular or histogram form.
- The **timex** command, which times a command and can also be used to report concurrent system activity and process accounting activity. **timex** is a modified form of the **time** command.

Also included as part of the system activity package is the system activity report package, which you can use to set up automatic system activity daily reports.

The system activity information reported by this package is derived from a set of system counters located in the operating system kernel. Each of these counters is described in the subsection "System Activity Counters," later in this section.

Using the System Activity Commands

The commands in the system activity package help you observe system activity during:

- A controlled stand-alone test of a large system.
- An uncontrolled run of a program that is observing the operating environment.
- Normal production operation.

Using the sar Command

Use the **sar** command to display, in tabular form, observed system activity based on a sampling interval and a number of intervals that you specify as part of the command. See your *CENTIX Operations Reference Manual* for a detailed description of using **sar**. The following is a general discussion.

You can use the **sar** command in two ways:

- When you use **sar** with the arguments *t* and *n*, it invokes the data collection program sadc. This program samples the system activity counters in the operating system every *t* seconds for *n* intervals and generates system activity reports in real time.

  You should use the **-o** option to save the sampled data in a file so that you can examine it later.

- When you use **sar** without the arguments *t* and *n*, it generates system activity reports for a time that was specified either in a previously recorded file or, by default, in the standard system activity daily data file /usr/adm/sa/sa*dd* for the current day *dd*.

Using the timex Command

The **timex** command is explained in detail in your *CENTIX Operations Reference Manual*. The following paragraphs are an overview.

The **timex** command is an extension of the **time** command. If you use **timex** without any options, it behaves exactly like **time**. In addition to giving time information, the command can print a system activity report and a process accounting report.

You will usually use timex to measure a single command. You can, however, also time multiple commands by combining them in an executable file or by entering them as follows:

```
# timex sh -c "command1; command2; command3; . . . ;"
```

This sets up the parent-child relationships needed to extract the user and system times consumed by *command1*, *command2*, *command3*, (and by the shell).

Using the sadp Command

The **sadp** command is explained in detail in your *CENTIX Operations Reference Manual*. The following paragraphs are an overview.

The **sadp** command is a user level program that can be invoked independently by any user. It requires no storage or extra code in the operating system and allows the user to specify the disks to be monitored. The program is reawakened every second. It then reads system tables from /dev/kmem, and extracts the required information.

In the operating system, there is an iobuf for each disk drive. It contains two pointers that are head and tail of the I/O active queue for the device. The actual requests in the queue can be found in three buffer header pools: system buffer headers for block I/O requests, physical buffer headers for physical I/O requests, and swap buffer headers for swap I/O. Each buffer header has a forward pointer that points to the next request in the I/O active queue and a backward pointer that points to the previous request.

The **sadp** command snapshots the iobuf of the monitored device and the three buffer header pools once every second during the monitoring period. It then traces the requests in the I/O queue, records the disk access location, and seeks distance in buckets of 8-cylinder increments. At the end of the monitoring period, it prints out the sampled data. The output of **sadp** can be used to balance load among disk drives and to rearrange the layout of a particular disk pack.

Using the System Activity Report (sar) Package

Three commands are available to use in setting up an automatic monitoring system. These are:

- **sadc**, which reads system counters from /dev/kmem and records them to a file. In addition to the file argument, you can use arguments that specify the sampling interval and the number of samples to be taken. If these arguments are not used, **sadc** writes a dummy record in the file to indicate a system restart.
- **sa1**, which is the shell procedure that invokes **sadc** to write system counters in the daily data file /usr/adm/sa*dd*, where *dd* is the day of the month. You can use sampling intervals and iterations as arguments.
- **sa2**, which is the shell procedure that invokes the **sar** command to generate daily report /usr/adm/sa/sar*dd* from the daily data file /usr/adm/sa/sa*dd*, where *dd* is the day of the month. The procedure also removes daily data files and report files after seven days. The starting and ending times and all report options of **sar** are applicable to **sa2**.

For details on using **sadc**, **sa1**, and **sa2**, see the **sar** entry in your *CENTIX Operations Reference Manual.*

You can use the **cron** system (see Section 12) to control the normal data collection and report generation operations. For example, you could have these entries in the /usr/spool/cron/crontabs/sys file:

```
0 * * * 0,6 /usr/sa/sa1

0 18-7 * * 1-5 /usr/lib/sa/sa1

0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
```

These entries cause the data collection program **sadc** to be invoked every hour on the hour. Depending on the arguments presented, **sadc** writes data to the data file one to three times at 20-minute intervals. Under the control of **cron**, therefore, the data file is written every 20 minutes between 8:00 and 18:00 on weekdays and hourly at other times.

In the example, data samples are taken most frequently during prime time on weekdays so that you will have the most detailed display when the system is being used the most. You should invoke **sa1** hourly rather than daily. This ensures that if the system crashes, data collection will resume within an hour after the system is restarted.

Because system activity counters restart from zero when the system is restarted, a special record is written on the data file. You can do this by putting the following line into one of the system startup files:

```
su adm -c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

cron also controls the invocation of **sar** to generate the daily report through shell procedure **sa2**. You can select the time period that the daily report is to cover and the groups of system activity to be reported. For example, you could add this line to the /usr/spool/cron/crontabs/sys file:

```
0 20 * * 1-5/usr/lib/sa/sa2 -s 8:00 -e 18:00 -i 3600 -uybd
```

This entry causes **sar** to generate daily reports from the daily data file at 20:00 on weekdays. The daily report reports the CPU utilization, terminal device activity, buffer usage, and device activity every hour from 8:00 to 18:00.

System Activity Counters

The operating system manages a number of counters that record various activities and provide the basis for the system activity reporting system. The data structure for most of these counters is defined in the sysinfo structure in /usr/include/sys/sysinfo.h (see the subsection "The sysinfo Structure," later in this section). The system table overflow counters are kept in the _syserr structure. The device activity counters are extracted from the device status tables. In this version, the I/O activity of the following devices is recorded: RP06, RM05, RS04, RF11, RK05, RP03, RL02, TM03, and TM11.

The following paragraphs describe the system activity counters sampled by the system activity package.

CPU Time Counters

There are four time counters that can be incremented at each clock interrupt, 60 times a second. Each counter corresponds to one of the modes in which the CPU can be at interrupt: idle, user, kernel, and wait for I/O completion. At interrupt, the counter that matches the CPU mode is incremented.

lread and lwrite

The lread and lwrite counters are used to count logical read and write requests issued by the system to block devices.

bread and bwrite

The bread and bwrite counters are used to count the number of times that data is transferred between the system buffers and the block devices. These actual I/Os are triggered by logical I/Os that cannot be satisfied by the current contents of the buffers. The ratio of block I/O to logical I/O is a common measure of the effectiveness of the system buffering.

phread and phwrite

The phread and phwrite counters count read and write requests issued by the system to raw devices.

swapin and swapout

The swapin and swapout counters are incremented for each system request that initiates a transfer from or to the swap device. Because text and data are handled separately, more than one request is usually involved in bringing a process into or out of memory. Frequently used programs are kept on the swap device and are swapped in rather than loaded from

the file system. The swapin counter reflects both these initial loading operations and resumptions of activity. The swapout counter reveals the level of actual swapping. The amount of data transferred between the swap device and memory is measured in blocks and counted by bswapin and bswapout.

pswitch and syscall

The pswitch and syscall counters are related to the management of multiprogramming. syscall is incremented each time that a system call is invoked. The numbers of invocations of the read, write, fork, and exec system calls are kept in counters sysread, syswrite, sysfork, and sysexec, respectively.

pswitch counts the times that the switcher is invoked. The switcher is invoked under these conditions:

- A system call results in a road block.
- An interrupt causes a higher priority process to wake up.
- A one-second clock interrupt occurs.

iget, namei, and dirblk

These counters apply to file-access operations. iget and namei are the names of operating system routines. The counters record the number of times the respective routines are called.

namei is the routine that performs file system path searches. It searches the various directory files to get the associated i-number of a file that corresponds to a special path.

iget is a routine called to locate the inode entry of a file (i-number). It first searches the in-core inode table. If the inode entry is not in the table, the iget routine gets the inode from the file system where the file resides and makes an entry in the in-core inode table for the file. iget returns a pointer to this entry. iget is called by namei and other file access routines.

Counter dirblk records the number of directory block reads issued by the system. The number of directory blocks read divided by the number of namei calls is an estimate of the average path length of files.

runque, runocc, swpque, and swpocc

These counters record queue activities. At each one-second interval, the clock routine examines the process table for processes in core and in ready state. If the routine finds any, the counter runocc is incremented and the number of such processes is added to the counter runque.

The clock routine also checks the process table for any processes in the swap device in the ready state. If the swap queue is occupied, the counter swpocc is incremented and the number of processes in the swap queue is added to the counter swpque.

readch and writech

The readch and writech counters record the total number of bytes (characters) transferred by the read and write system calls, respectively.

rcvint, xmtint, mdmint, rawch, canch, and outch

These counters monitor terminal device activities. rcvint, xmtint, and mdmint measure hardware interrupt occurences for receiver, transmitter, and modem, respectively. rawch, canch, and outch count the number of characters in the raw queue, canonical queue, and output queue. Characters generated by devices that operate in the "cooked" (preprocessed) mode , such as terminals, are counted in both rawch and (as edited) in canch. Characters from raw devices, such as communications processors, are counted in rawch.

msg and sema

These counters record message sending and receiving activities and semaphore operations, respectively.

io_ops, io_bcnt, io_act, and io_resp

These counters monitor I/O activity. Each disk drive and tape drive has one of each of these counters associated with it in the device status table. The io_ops counter is incremented when an I/O operation occurs on the device. It includes block I/O, swap I/O, and physical I/O.

Counter io_bcnt counts the amount of data transferred between the device and memory in 512-byte units.

Counters io_act and io_resp measure the active time and response time of a device in time ticks summed, divided by all I/O requests that have completed for each device. The device active time includes the device seeking, rotating, and data transferring times. The reponse time of an I/O operation is calculated from the time that the I/O request is queued to the device, to the time when the I/O completes.

inodeovf, fileovf, textovf, and procovf

These counters are extracted from _syserr structure. When an overflow occurs in any of the inode, file, text, and process tables, the corresponding overflow counter is incremented.

The sysinfo Structure

```
struct sysinfo    {
                  time_t          cpu[4];
#define           CPU_IDLE        0
#define           CPU_USER        1
#define           CPU_KERNEL      2
#define           CPU_WAIT        3
                   time_t         wait[3];
#define           W_IO            0
#define           W_SWAP          1
#define           W_PIO           2
                  long            bread;
                  long            bwrite;
                  long            lread;
                  long            lwrite;
                  long            phread;
                  long            swapin;
                  long            swapout;
                  long            bswapin;
                  long            bswapout;
                  long            pswtich;
                  long            syscall;
                  long            sysread;
                  long            syswrite;
                  long            sysfork;
                  long            sysexec;
                  long            runque;
                  long            runocc;
                  long            swpque;
                  long            swpocc;
                  long            iget;
                  long            namei;
                  long            dirblk;
                  long            readch;
                  long            writech;
                  long            rcvint;
                  long            xmtint;
                  long            mdmint;
                  long            rawch;
                  long            canch;
                  long            outch;
                  long            msg;
                  long            sema;
};
```

System Activity Package Formulas

The following equations show how items reported as part of the system activity package are derived. Each item is the data difference sampled at two distinct times, *t1* and *t2*.

CPU Utilization

%-of-cpu-*x* = cpu-*x* / (cpu-idle + cpu-user + cpu-kernel + cpu-wait) * 100

where cpu-x is either cpu-idle, cpu-user, cpu-kernel (cpu-sys), or cpu-wait

Cache Hit Ratio

%-of-*cache-I/O* = (logical-I/O - block-I/O) / logical-I/O * 100

where *cache I/O* is either cache read or cache write.

Disk or Tape I/O Activity

%-of-busy = I/O-active / (t2-t1) * 100

avg-queue-length = I/O-resp / I/O-active

avg-wait = (I/O-resp - I/O-active) / I/O-ops

avg-service-time = I/O-active / I/O-ops

Queue Activity

avg-*x-queue*-length = *x-queue* / *x-queue*-occupied-time

%-of-*x-queue*-occupied-time = *x-queue*-occupied-time / (t2-t1)

where *x-queue* is run queue or swap queue.

The Average of System Activity

avg-rate-of-*x* = *x* / (t2-t1)

where *x* is swap in/out, blks swapped in/out, terminal device activities, read/write characters, block read/write, logical read/write, process switch, system calls, read/write, fork/exec, iget, namei, directory blocks read, disk/tape I/O activities, message, or semaphore activities.

Running Periodic Jobs with the cron Command

The **/etc/cron** process performs selected jobs at defined times. **cron** is started up by the /etc/rc file at boot time. After boot time, **cron** wakes itself up once a minute to check if there are any more jobs to start up.

The jobs that **cron** initiates are listed in files in the /usr/spool/cron/crontabs directory. When the system software is installed, the root, sys, and adm files are provided in the directory.

Format of the Files in the crontabs Directory

Each entry in a crontabs file contains six fields. The fields are separated by spaces or tabs. The following is an example of a crontabs file entry:

```
0 1 * * * /usr/bin/calendar -
```

The first five fields define when the command in the sixth field is to be started by **cron**. The fields (in order, from left to right) define:

- Minute (from 0 through 59).
- Hour (0 through 23).
- Day of the month (1 through 31).
- Month of the year (1 through 12).
- Day of the week (0 through 6, with 0 representing Sunday).

In each of the five fields, you can enter:

- A single integer from the sets defined above.
- Two integers separated by a hyphen to represent a range of times.
- A list of integers separated by commas to represent more than one time within a field. (For example, in the first field, 1,6,10 represents minute 1, minute 6, and minute 10.)
- An asterisk (*) to represent all possible values. (For example, an asterisk in the fourth field means that the command is to run every month.)

Note that the days can be specified by two fields (day of the month and day of the week). If both fields are filled out, the command is run on the days defined in both fields. For example, the string 0 0 1,15 * 1 runs a command on the first and fifteenth of each month, and on every Monday. If you use only one of the fields to specify days, put an asterisk in the other field. For example, the string 0 0 * * 1 runs a command only on Mondays.

The sixth field of a line in a crontab file is a string executed by the shell at the specified times. A percent character in this field (unless escaped by \) is translated to a new-line character. Only the first line (up to a % or end of line) of the command field is executed by the shell. The other lines are treated as standard input to the command.

In the example shown above, the calendar command is executed at 0100 every day.

Adding User Files to the crontabs Directory

As superuser, you can use a text editor to add a user file to the crontabs directory. Regular users, however, do not have write permission to the directory. For a user to be able to add a file to the crontabs directory, you must give the user access to the **crontab** command (see "Giving Users Access to **crontab**," later in this section).

To add a file to the directory, the user follows these steps:

1 Enter:

```
$ crontab
```

2 Press the RETURN key. Enter the lines that you want in your crontabs file.

3 Press the CODE and D keys.

The file is automatically named for the user and put into the /usr/spool/cron/crontabs. For example, if the user Gary uses the **crontab** command, his file is named /usr/spool/cron/crontabs/gary. The file is given an access code of 444 (see your *CENTIX Operations Guide* for details on file access permission), and root is made the owner. This means that the user that creates a crontabs file cannot use a text editor to directly change the file. See "Making Changes to a crontabs File," below.

When the user uses the **crontab** command, the new input replaces any previous crontabs file named for that user.

Making Changes to a crontabs File

As superuser, you can enter any user's crontabs file and use a text editor to make changes and additions.

A regular user, however, is not given write permission to the crontabs files and cannot make changes directly to the file. If the user's file is simple, he or she can use the **crontab** command to create a new file. The new file will replace the old file. If the user's crontabs file is longer than a few lines, he or she probably does not want to retype the entire file. The user can follow these steps to make changes:

1 Copy the crontabs file to another file. Enter:

```
$ cp /usr/spool/cron/crontabs/username newname
```

2 Use a text editor to make any changes or additions to the *newname* file.

3 Enter:

```
$ crontab newname
```

The *newname* file is copied into the /usr/spool/cron/crontabs directory under the user's name. That is, if the user Gary enters **crontab** *newname*, the *newname* file is written to /usr/spool/cron/crontabs/gary, replacing the old gary file.

Giving Users Access to crontab

Any user that needs a file in the /usr/spool/cron/crontabs directory should be given permission to use **crontab** so that he or she can create or change the file. To give a user permission, use a text editor to add his or her login name to the /usr/lib/cron/cron.allow file. Put each name on a separate line.

If you have many users, and it is easier to list who is *not* allowed access to **crontab**, remove the /usr/lib/cron/cron.allow file, and list the users who should be denied access in the /usr/lib/cron/cron.deny file. Put each name on a separate line.

To allow access to all users, remove the /usr/lib/cron/cron.allow file and leave the /usr/lib/cron/cron.deny file empty.

To deny access to all users except superuser, remove both the /usr/lib/cron/cron.allow and /usr/lib/cron/cron.deny files.

Using the at and batch Commands

If you want a command to be executed once at a defined later time, you can use the **at** command. The command has the form:

```
$ at time
```

where *time* is the time at which you want the command to be executed (see the next paragraph for details). After you press RETURN at the end of the command line, enter the command that you want to be executed, then press the CODE and D keys.

Follow these rules to enter *time*:

- To enter a specific hour, use two digits. You can append am/pm, or use the 24-hour clock system. That is, if you want the command to run at 2:00 in the afternoon, enter 02pm or 14.

- To enter a specific minute, enter hour:minutes. Use two digits for each value. That is, if you want the command to run at three minutes after three in the afternoon, enter 03pm:03 or 15:03.

There are two optional fields. You can also enter a specific date on which you want the command run, or specify time increments: either minutes, hours, days, weeks, or months.

The **batch** command is related to the **at** command. Use the **batch** command when you want a command to run when system load permits. For more details on **at** and **batch**, see your *CENTIX Operations Reference Manual.*

Giving Users Access to at and batch

If you want a user other than superuser to use **at** and **batch**, use a text editor to add his or her login name to the /usr/lib/cron/at.allow file. Put each name on a separate line.

If you have many users, and it is easier to list who is *not* allowed access to **at** and **batch**, remove the /usr/lib/cron/at.allow file, and list the users who should be denied access in the /usr/lib/cron/at.deny file. Put each name on a separate line.

To allow access to all users, remove the /usr/lib/cron/at.allow file and leave the /usr/lib/cron/at.deny file empty.

To deny access to all users except superuser, remove both the /usr/lib/cron/at.allow and /usr/lib/cron/at.deny files.

Accessing BTOS from CENTIX

This section describes the methods you use to copy from, edit files in, and administer the BTOS operating system.

Copying Files from BTOS to CENTIX

You may need to copy a BTOS file into the CENTIX file system. You may, for example, want to use a CENTIX (rather than BTOS) text editor to modify a BTOS configuration file. You can use the **ofcopy** command to copy the BTOS file into CENTIX. Enter:

```
# ofcopy 'BTOSfilename[^password]' CENTIXfilename
```

where:

- *BTOSfilename* is the complete file name, [*volume*]<*directory*>*file*, of the file that you are copying. Make sure that you include the single quotes around *BTOSfilename*.
- ^*password* is included if the BTOS file is password-protected. Use the file's password.
- *CENTIXfilename* is the complete pathname of the CENTIX file to which you are copying. (If you are currently working in the directory in which the CENTIX file will be stored, you can use only the file name, rather than the complete pathname.)

You also use **ofcopy** to copy files from CENTIX back to BTOS. Enter:

```
# ofcopy CENTIXfilename 'BTOSfilename'
```

See the *CENTIX Operations Reference Manual* for details on **ofcopy**.

Editing BTOS Files

To edit a BTOS file, use one of the CENTIX/BTOS text editing commands, **ofed** or **ofvi**. These text editors operate exactly like their CENTIX counterparts, **ed** and **vi**.

To initiate the editors on a BTOS file, enter:

```
# ofed 'BTOSfilename[^password]'
```

```
# ofvi 'BTOSfilename[^password]'
```

where:

- *BTOSfilename* is the is the complete file name, [volume]<directory>file, of the file that you are copying.
- *^password* is included if the BTOS file is password-protected. Use the file's password.

Make sure that you include the single quotes around *BTOSfilename*. See the *CENTIX Operations Guide* for details on using the **ed** and **vi** text editors.

Listing BTOS Files and Directories

You can use the **ofls** command to list specified BTOS files and directories through CENTIX. The command has several options which allow you to select certain information about the files and directories. For example, the **-l** option lists the sizes of the selected files and directories, and the last time that the selected files where modified. See the *CENTIX Operations Reference Manual* for details on **ofls**.

Using the BTOS MCommands

The BTOS Master Commands (MCommands) are provided with your system to allow you to administer the BTOS portion of your operating system. Some examples of BTOS MCommands that you may need are **MCopy**, which copies a BTOS file from one BTOS directory to another, **MIVolume**, which initializes disks, and **MPLog**, which displays the entries that are in the system status log file. For a description of the MCommands and how to use them, see the *BTOS Operations Reference Manual*.

Note: *If you have more than one tape drive on your system, you cannot access any tape drive with the BTOS MCommands.*

Many of the MCommands issue messages telling you to select an action by pushing the appropriate key on your terminal keyboard. The messages refer to a keyboard that is different from the one on your system. Use the following to determine which key to use:

Note: *When two keys are listed, separated by a hyphen, push both keys simultaneously.*

MCommand Message Calls For:	Use PT 1500 Key(s):	Use Dumb Serial Terminal Key(s):
GO	GO	ESC or CONTROL-[
CANCEL	CODE-G	CONTROL-G
FINISH	DELETE	CONTROL-D

You have two ways to access the MCommands. These are described in the following paragraphs.

Accessing the MCommands through centrEASE

You can use the "Issue BTOS Commands through CENTIX" option of centrEASE to access the BTOS MCommands. After you select the option and the command that you want to run, centrEASE displays the MCommand menu that is shown in the *BTOS Operations Reference Manual*. You fill out the menu and run the command. See the *centrEASE Operations Reference Manual* for details on running BTOS commands through centrEASE.

Accessing the MCommands through ofcli

You can also use the CENTIX command **ofcli** to initiate the BTOS MCommands. When you invoke **ofcli**, you enter the BTOS Command Line Interpreter (CLI) mode. In this mode, the MCommands are entered as run statements, rather than from menus. For a description of the other tasks that you can perform in CLI, see "Using the BTOS Command Line Interpreter," below.

To enter the CLI mode from CENTIX , enter:

```
# /etc/ofcli
```

When you press the RETURN key, this message appears at your terminal:

```
Entering Administrator Mode
  $
```

Note: *This dollar sign ($) is* not *the shell prompt. You cannot enter shell commands while in administrator mode.*

Enter the run statement for the MCommand at the $ prompt. A run statement has the form:

```
$ run [sys]<directory>mcommand.run,parameters
```

where:

- *directory* is the BTOS directory in which the MCommand resides. The MCommands are in the <sys> and <admin> directories. The run statements for the MCommands, including the directories in which they reside, are given in the *BTOS Operations Reference Manual*.
- *mcommand* is the name of the MCommand. If there are spaces between words in the BTOS command, leave out the spaces. For example, for MCreate Directory, enter MCreateDirectory.
- *parameters* is a list of the MCommand parameters, separated by commas. For the MCommand parameters, see the command form in the *BTOS Operations Reference Manual*.

  The first parameter shown in the command form menu is the first parameter that you list after "run,". The second parameter in the menu is the second one you list, and so on. If you are not giving a value for a parameter, you must provide a blank space between commas to represent the parameter (unless there are no more parameters listed after the blank parameter).

For example, to create a BTOS directory called "Accounts", using the default values for the protection level and number of files, and assigning a directory password of "Jumbled", enter:

```
$ run [sys]<admin>MCreateDirectory.run,Accounts,,,Jumbled
```

See "CLI Command Syntax," below, for more details on entering run statements.

You can run only one MCommand each time that you enter the CLI through **ofcli**. When the MCommand has finished executing, this message appears:

```
Execution ended: date, time

Leaving Administrator Mode

#
```

To run another MCommand, enter **ofcli** at the prompt and begin again.

Using the BTOS Command Line Interpreter

The BTOS Command Line Interpreter (CLI) interprets and executes CLI commands to XE 500 BTOS. The CLI acts like the BTOS Executive for specific administrative and implementation tasks:

- When BTOS MCommands are entered from CENTIX through ofcli, the CLI interprets the run statements.
- When a processor is booted at system start-up, BTOS uses the CLI to interpret the run statements in the processor's initialization file.
- The XE 500 BTOS Debugger is run through the CLI. The Debugger is used to test application programs running on XE 500 processors and to analyze processor memory dumps. Refer to the *XE 500 BTOS Debugger Operations Guide* for more information about the Debugger.
- The CLI can be used to install and run applications on processors during normal system operations.

CLI commands are ASCII text lines. These text lines are submitted to the CLI through processor initialization files, from an RS-232-C serial terminal connected to a CP or TP CLI port, or through the **ofcli** command.

How CLI Works

The CLI runs in the primary partition of each BTOS processor. Only one program can execute in a BTOS partition at a time, so when CLI runs another program, it sets the BTOS exit run file to indicate its own run file, [sys]<sys>Cli.run. The successive instances of the CLI can be considered as a single program; they maintain a context file to maintain continuity.

To show how CLI works, it is helpful to trace what happens after a processor's operating system is loaded during system bootup.

1 Once the processor's operating system is loaded into memory, BTOS executes its initial program, the CLI, in the primary application partition.

2 The CLI executes the run statements in the processor's initialization file. If a run statement is to execute another program, the execution includes three steps:

   a The CLI sets the exit run file to be its own run file.

   b The CLI has BTOS run the other program, overlaying the CLI in the primary application partition.

   c When the other program terminates, CTOS executes the exit run file, reloading the CLI into the primary application partition.

3 If the processor is configured with a CLI port, the CLI executes any commands entered from that channel. If a command is to execute another program, the execution follows the same procedure described in step 2.

Figure 13-1 is a flow chart that represents the process just described.

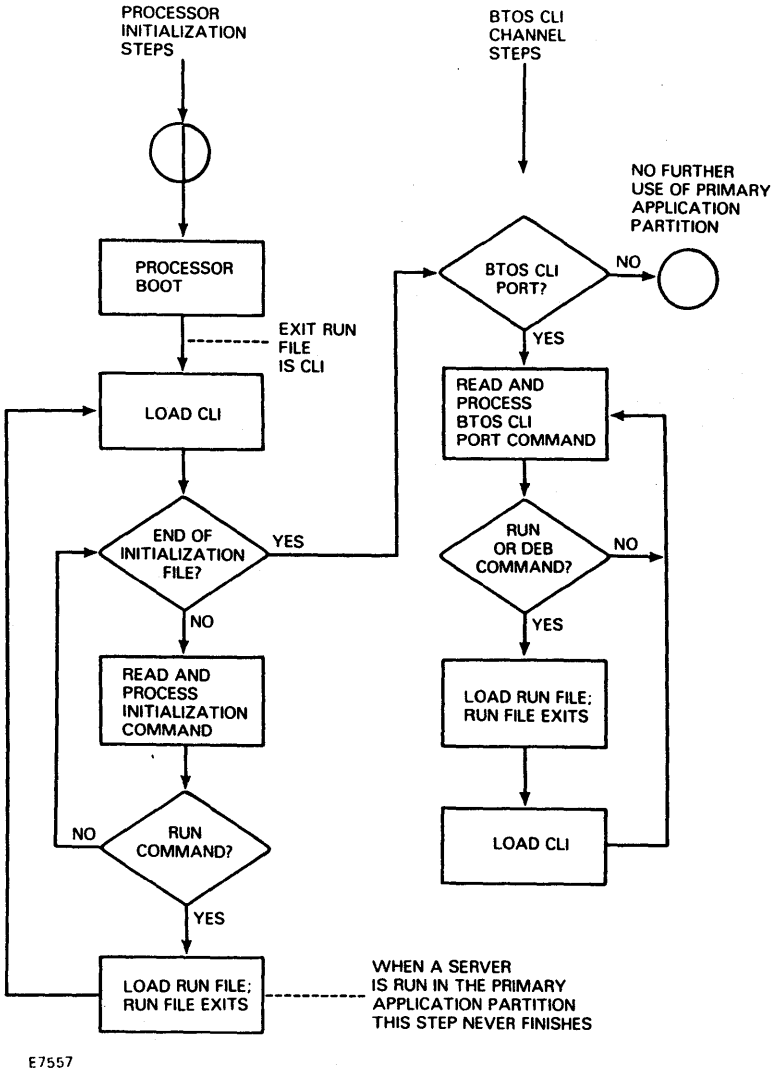
If a program is to run concurrently with the CLI, the CLI commands must install the program in a secondary application partition. Refer to the *CENTIX Installation and Implementation Guide* for more information about running applications in secondary partitions.

Communicating with the CLI

There are three ways to communicate with the CLI:

- Through command statements in a processor initialization file.
- Interactively using an RS-232-C serial terminal connected to a CP or TP CLI port. A CP or TP CLI port is defined by adding the phrase "connect=ctos" to the end of the RS-232-C channel's entry in the processor configuration file. There can only be one CLI port per CP or TP.
- Interactively, using **ofcli** from the CENTIX system console.

Figure 13-1 **Processor Initialization File Execution**



Using Processor Initialization Files

Each processor normally has an initialization file. Processor initialization file names take the form [sys]<sys>init*Xpnn*.jcl, where *Xpnn* is the processor designation (for example, FP00, CP02).

The initialization file is a text file that has a line entry for each system service to be run on the processor. Each line entry in an initialization file is a run statement that loads and activates the system service's run file on the processor.

For more information about processor initialization files and the services that can be run on them, refer to the *XE 500 CENTIX Installation and Implementation Guide*.

Using CLI Ports

Each CP or TP can have one CLI port. To designate an RS-232-C channel as the CLI port, add the phrase "connect=ctos" to that channel's entry in the processor's configuration file.

To run CLI from this channel, an RS-232-C serial terminal must be connected to the channel.

The terminal hardware must be set with the following line parameters:

- 9600 baud.
- 8 data bits.
- One stop bit.
- No parity.

When the terminal is turned on, CLI prompts for input with a dollar sign ($).

The channel is not usable if an AdminAgent is running on the processor. The AdminAgent must be terminated before you can send commands to the CLI. To terminate the AdminAgent, enter CODE-Z (or CONTROL-Z on some terminals). The BTOS status code 4, Operator Intervention, is returned.

Caution: *Terminating the AdminAgent can cause unpredictable results if the processor is currently running a program initiated by the AdminAgent. Before terminating the AdminAgent, make sure that no user is executing a master utility through that AdminAgent.*

Commands executed through the CLI port can only be run on the processor to which the CLI port is connected.

Using ofcli

You can also use the CENTIX command **ofcli** to enter the BTOS CLI mode. To enter the CLI mode from CENTIX, enter:

```
# /etc/ofcli
```

When you press the RETURN key, this message appears at your terminal:

```
Entering Administrator Mode

$
```

At this point, you can do one of these things:

- Enter one or more CLI commands. See "Using CLI Commands," below.
- Execute a BTOS run file. See "Executing a Run File," below. (Note that this is how you execute an MCommand; see "Accessing the MCommands Through **ofcli**," above.)
- Execute a BTOS JCL File. See "Calling JCL Files for Execution," below.

If you execute a run file or a JCL file, the CLI mode is automatically terminated when the file has completed. This message appears:

```
Execution ended: date, time
Leaving Administrator Mode
#
```

To run another file, enter **ofcli** at the prompt and begin again.

You can also force CLI mode to terminate by pressing the DELETE key.

CLI Command Syntax

The CLI command syntax comprises a basic command form and conventions for continuation lines, comments, and call parameters.

Command Form

A CLI command has the form

$command params

where

$	for interactive CLI, is the prompt displayed by CLI and, for JCL files, indicates that the line in the file is a CLI command.
command	is the name of the CLI command. Commands can be entered in upper or lower case. If *command* is not one of the standard CLI commands, it is assumed to be an implicit call to a JCL file (see the subsection on calling a JCL file, later in this section). Commands are always separated from parameters by spaces (not commas).
params	is a parameter list. Parameters are separated from each other by commas; spaces directly before or after parameter commas are ignored.

A parameter is identified solely by its position in the parameter list. If you are not specifying certain optional parameters, you must still include a comma in the command

for each parameter left out, unless no subsequent parameters are being specified. For example, an application has, in order, a required parameter and two optional parameters. The run statement for this application when specifying only the second optional parameter would be:

```
$run [sys]<sys>application.run,param1,,param3
```

The run statement for this application when specifyng neither of the optional parameters would be:

```
$run [sys]<sys>application.run,param1
```

Parameters are passed literally, including the case of letters.

Special Characters

The CLI attaches special meaning to the dollar sign ($), ampersand (&), percent sign (%), semicolon (;), comma (,), parentheses [( )], and backslash (\). To include a special character literally in a command, precede it with a backslash, as in the following example:

```
$run Loan.run, \$150\,000, 12\%
```

To keep spaces and commas as part of a parameter, enclose the parameter in single quotes. For example,

```
$run Newname.run, 'J. C. Ryke, Jr.'
```

To run a program with subparameters, enclose the group of parameters in parentheses [()]. The parameters within the parentheses are passed as a single parameter with subparameters. For example, in the following run statement for the MDelete utility,

```
$run [sys]<sys>MDelete.run,(p2 p3 p4),y
```

p2, p3, and p4 are files being specified for the "File list" parameter of the utility.

Continuation Lines

In a text file, if a command must be broken over to other lines, end each line of the command with an ampersand (&). When CLI is used interactively, the CLI prompts for the continuation line with an ampersand instead of a dollar sign ($).

For example, the following two run statements are functionally identical except that the second one uses a continuation line:

```
$run [sys]<sys>Example.run,p1,p2,p3,p4

$run [sys]<sys>Example.run,p1,p2,&
p3,p4
```

Comments

To insert comments that the CLI is to ignore, insert a semicolon (;) before the comment. The CLI ignores anything on that line that follows the semicolon. For example,

```
$return ; sort completed

; initcp01.jcl
$run minstallspl.run
```

Using Call Parameters

A JCL file executed by the CLI **Call** command or an implicit call can access the additional parameters specified following the JCL file name. The additional parameters in the JCL file are numbered from zero and are preceded by a percent sign (for example, %0, %1, %2, and so on).

For example, for the following JCL file called [sys]<sys>pcopy.jcl,

```
; pcopy - copy a file from one prefix to another
; format: pcopy file,prefix1,prefix2
$run [sys]<sys>mcopy.run,%1%0,%2%0
```

the following three CLI commands are equivalent (the third command line is an implicit call):

```
$run [sys]<sys>mcopy.run,[d2]<sys>dbms,[d3]<sys>dbms

$call [sys]<sys>pcopy.jcl,dbms,[d2]<sys>,[d3]<sys>

$pcopy dbms,[d2]<sys>,[d3]<sys>
```

File Name Conventions

File names in CLI commands follow the standard BTOS file name conventions. Refer to Section 14 of this guide for an overview of BT name conventions.

By default, the CLI path is set to [sys]<sys>. You can use the CLI **Path** command (described later in this section) to change the current path during a CLI session.

Using CLI Commands

The command name, the first word of a CLI command, specifies the action to be executed. Table 13-1 lists the CLI commands.

In addition to these commands, you can create your own commands that implicitly call JCL files; refer to the subsection "Calling JCL Files for Execution."

Table 13-1 **CLI Commands**

Command	Use
Cancel	Terminates execution of a called JCL file.
CancelOnError	Indicates to the CLI to terminate processing of a JCL file if an error is returned.
ContinueOnError	Indicates to the CLI to continue processing a JCL file even if an error is returned.
Deb	Loads and debugs an object code run file.
End	Signals the end of a JCL file.
Path	Sets the BTOS working path.
Prefix	Sets the prefix of a JCL file being called through an implicit call.
Return	Terminates execution of the JCL file currently being processed.
Run	Runs an object code run file.
Suffix	Sets the suffix of a JCL file being called through an implicit call.

Executing a Run File

To load and execute a run file at a processor, use the **Run** command. A run file is a loadable object file produced by BTOS Link. By convention, such files have names ending in ".run".

The **Run** command must be followed by the name of the run file to be executed. The run file can, in turn, be followed by the parameters that apply to it. For example, the following command causes the MFiles utility to be executed on the directory [d2]<acct>:

```
$run [sys]<admin>MFiles.run, [d2]<acct>*, y, [spl]
```

If the program examines its command name, the first parameter should consist of two subparameters: the run file name and the command name, in that order. For example,

```
$run ([sys]<sys>loguser.run, 'Log User'),friday
```

When the run file execution is complete, BTOS reloads the CLI and execution proceeds with the next CLI command.

Calling JCL Files for Execution

There are two ways to call a JCL file for execution by the CLI:

- You can directly call a JCL file to be executed using the **Call** command. The **Call** command must be followed by a space and the name of the JCL file. The JCL file name can then be followed by a comma and any parameters to be passed to the JCL file during execution.
- You can call a JCL file implicitly by entering the root name of the JCL file name. By default, the JCL file name must have the form [sys]<sys>*rootname*.jcl, where *rootname* is the name to be used as the CLI command. The root name can then be followed by a space and any parameters to be passed to the JCL file during execution.

  The default prefix, "[sys]<sys>", and the default suffix, ".jcl", can be changed using the Prefix and Suffix commands.

If you are passing parameters to the JCL file from a call command, their target must be defined in the JCL file using call parameters, as described previously in this section.

A called JCL file can include call commands; these are nested calls. Calls can nest to any depth, provided there is room left in the context file.

An example of a **Call** command is

```
$call [sys]<sys>Bill.jcl,x0,x1,x2
```

An example of an implicit call that is functionally identical to the previous **Call** command is

```
$Bill x0,x1,x2
```

The contents of the JCL file [sys]<sys>Bill.jcl could be

```
$run [sys]<sys>billing.run,%0,%1,%2
```

To change the default prefix or suffix of an implicit call command, use the **Prefix** or **Suffix** command, respectively. This command must be executed before the implicit call statement. The prefix or suffix defined by a **Prefix** or **Suffix** command is effective until the next instance of the **Prefix** or **Suffix** command.

To use one of these commands, enter the command name followed by a space and the new prefix or suffix to be used. The following example allows the JCL file called [d2]<acct>Bill.oldjcl to be called implicitly using the root name Bill:

```
$prefix [d2]<acct>
$suffix .oldjcl
$Bill
```

Ending a JCL File

Enter the **End** command as the last line of a JCL file to indicate where execution should end. The **End** command takes no parameters. It can be omitted without any effect, except if the file is also to be executed by the Batch system.

Terminating Execution of JCL Files

To terminate the processing of the currently executing JCL file, use the **Return** command. There are no parameters for the **Return** command. This command terminates only one level of nesting.

To terminate the processing of the currently executing JCL file and all nested calls, use the **Cancel** command. After terminating all calls, the CLI is reloaded. If CLI had been reading a processor initialization file, it then begins execution of that file again. If interactive mode was being used, CLI then prompts for another command.

By default, the execution of a JCL file continues with the next command if an error is returned while the file is being processed. You can have the execution of a JCL file be terminated when an error is received by entering the **CancelOnError** command in the file. The CancelOnError state remains effective until a **ContinueOnError** command is executed.

For example, the following JCL file would cause execution to terminate only if an error is returned while the MCopy utility is running.

```
$run [sys]<admin>MRename.run,[d2]<acct>*,[d2]<sales>*
$cancelonerror
$run [sys]<admin>MCopy.run,[d2]<sales>*,[d3]<customers>*
$continueonerror
$run [sys]<admin>MRemoveDirectory.run,[d2]<acct>
```

The **Return**, **Cancel**, **CancelOnError**, and **ContinueOnError** commands do not take parameters.

Changing the Path

To change your current working path during a CLI session, use the **Path** command. The **Path** command has the format

```
$path node,vol,dir,file,pass
```

where *node* is specified only if the system is running B-Net, *vol* is the volume or disk device name, *dir* is the directory name, *file* is the file name, and *pass* is a password, if required. Square brackets ([]) should not be used with the volume or disk device name. Angle brackets (<>) should not be used with the directory name. The caret (^) should not be used with the password.

The following example specifies a path of [d3]<bt> with the password "security" (a node and file are not specified):

```
$path ,d3,bt,,security
```

The default path is {local}[sys]<sys> with no default password.

The new path is effective until changed by another **Path** command.

Loading a Run File during a Debugger Session

To load a run file while the Debugger is activated, enter the **Deb** command. Execution, if any, is under debugger control.

Note: *To use the* ***Deb*** *command, the version of BTOS running on the processor must have been linked with the Debugger. Refer to the* XE 500 BTOS Customizer Operations Guide.

CLI Status Messages

The following is a list of CLI status messages. Explanations are provided as needed.

Bad command name or no such command file exists.

Command file name:*filename*

The command file name given in a call statement does not exist. Note that any unrecognized CLI command is assumed to be an implicit call.

Bad yes/no parameter.

The specified yes/no parameter is invalid. Specify "y" or "n".

Cannot open Context file in $ or current directory.

The Context file is used to save the state information parameters when JCL command files nest.

Cannot open specified command file.

Error in formatting date/time, error = *x*.

Error when accessing JCL file, error = *x*.

Fatal error encountered.

Termination status code:*x*

The currently running program had a fatal error. Refer to the *BTOS Status Codes Reference Manual* for the meaning of the status code.

Syntax error on current command line.

The file specification is invalid.

Unbalanced quotation marks in the Parameters field.

Executing BTOS Utilities on Preferred and Required Processors

You can improve system performance by using the **ofcli** command to direct BTOS utilities to run on certain processors. This is explained in detail in Section 17 of the *CENTIX Installation and Implementation Guide*.

Initializing and Verifying Disks

Before a disk can accept data, you must format—or initialize—it with the BTOS **MIVolume** utility. This section provides information and procedures for:

- Initializing disks to create new volumes.
- Reinitializing volumes to correct volume fragmentation or add new disk bad spot information.
- Generating a listing of a volume's bad spots.
- Verifying a disk's hardware and volume structure integrity.

Overview of Disks, BTOS Volumes, and BTOS File Systems

Each XE 500 disk must be *initialized* (also referred to as *formatted*) to be able to store data. An XE 500 disk can be a disk cartridge, a built-in disk, or a Storage Module Device (SMD) disk.

Note: *In this guide, a "built-in disk" is assumed to be a 5 1/4-inch hard disk controlled by a File Processor (FP). In some base enclosure styles, SMD disks can also be built into the XE 500; however, they will be referenced as "SMDs." All SMDs, whether they are in an XE 500 base enclosure or an MD3 enclosure, are controlled by Disk Processors (DPs).*

Once a disk drive is properly formatted to accept data, it is said to contain a volume. The term *disk drive* refers to the hardware device; *volume* refers to the complete file system unit of information stored on the disk. Each formatted disk in the system has a volume associated with it. BTOS frequently allows the device name of the disk drive and the volume name to be used interchangeably when referring to the information stored on the disk.

BTOS Disk Drive Device Names

In BTOS, disk devices are named according to their physical location in the system. The name conventions are different for disks controlled by FPs (disk cartridge drive and built-in disk drives) and for disks controlled by DPs (SMDs in an XE 500 base enclosure or an MD3 enclosure).

Disks in the first enclosure that are controlled by an FP are named d0 through d3; those in a second enclosure are named d4 through d7; and so on.

SMD disks that are controlled by the first DP in the system (DP00) are named s0 through s5. SMDs that are controlled by the second DP in the system (DP01) are named s6 through s11; and so on.

Figures in Appendix C show the various disk naming schemes.

BTOS disk drive device names and passwords can be defined in:

- The version of BTOS for the FP or DP by which they are controlled. These names and passwords are assigned when the processor's version of BTOS is generated. For more information about these parameters and operating system generation, refer to the *XE 500 BTOS Customizer Operations Guide*.
- The configuration file for the FP or DP by which they are controlled. For more information about these parameters in FP and DP configuration files, refer to the *XE 500 CENTIX Installation and Implementation Guide*.

Volume Names

You assign a BTOS volume name to a disk when you initialize it with the MIVolume utility. Follow these rules when selecting a volume name:

- Use a maximum of twelve characters, including any alphanumeric character, hyphens, and periods.
- The volume name cannot duplicate any other BTOS volume or device name.

  Do not use any of these as a volume name: d0, d1, d2, d3, and so on; s0, s1, s2, s3, and so on; Nu1; or Kb.

  Do not use a volume name that begins with any of these: Comm, Lpt, Spl, Tape, Vid, QIC.

The XE 500 BTOS File System and File names

The BTOS file system has a three-level structure:

- **Volume**. A volume is associated with each initialized disk in the system.
- **Directory**. Each volume can contain one or more directories. All directories are at the same level (that is, a directory cannot contain another directory).
- **File**. Each directory can contain one or more files.

A full BTOS filename designation has the following format:

[*volname*]<*dirname*>*filename*

where

volname	is the name of the volume on which the file is stored. Normally, the name of the disk device associated with the volume can also be used instead of the actual name of the volume.
dirname	is the name of the directory in which the file is stored. A directory name can have a maximum of 12 characters. Do not use angle brackets (<>) in the directory name.
filename	is the name assigned to the file. BTOS file names can contain a maximum of 50 characters from this set: all alphanumeric characters (both upper and lower case letters are accepted), periods, hyphens, and right angle brackets (>). Do not use these characters: [, +, =, <, &, @.

Overview of Volume Initialization

The MIVolume (Initialize Volume) utility prepares an XE 500 disk for use as a system volume. The MIVolume utility

- Formats the disk surfaces into 512-byte sectors.
- Performs read/write tests to identify surface defects.

- Writes volume control structures (the Volume Home Blocks, the Master File Directory, the Allocation Bit Map, and File Header Blocks) onto the medium.
- Creates system files.

Every disk must be initialized before it is used the first time.

Built-in disks and MD3 disks are normally initialized before they are shipped to customers or by the field representative when the system is installed.

Disk cartridges are not initialized before being shipped. It is your responsibility as the system administrator to initialize new disk cartridges.

You will have to reinitialize disks under certain conditions, including

- When system performance has been degraded due to volume fragmentation (see the subsection "Volume Fragmentation").
- If it is suspected that system files or volume control structures have become unreliable due to disk bad spots.
- Changing the size of certain volume-based files for the designated system disk, such as the system image file, the log file, the system crash file, and so on.
- Changing the maximum number of directories or files allowed on the volume.
- Setting the volume control structures to use primary headers only as opposed to primary and secondary headers.
- Setting protection for the <sys> directory.
- Entering new disk bad spots.

All disks come with a listing of known *bad spots*, which are defective areas on the disk surface. When a disk is initialized for the first time, this list is entered into a bad spot file, called badblk.sys, in the volume's <sys> directory. The volume uses the bad spot list to track the areas in which data cannot be stored.

The badblk.sys file is a coded file. The MIVolume and MVolume Report utilities generate a text listing of the known bad spots currently stored in the bad spot file.

Occasionally, disk areas that are marginally defective can cause intermittent data storage problems. The MDisk Verify utility provides for testing disk surface defects without destroying valid data, which the MIVolume utility would do. The MDisk Verify utility can also be used to test a disk cartridge's compatibility with a cartridge drive.

Volume Fragmentation

BTOS volumes can become fragmented over time. When files are created or extended, BTOS tries to allocate a single *disk extent* to store the new data. (A disk extent is one or more contiguous disk sectors that contain all or part of a file.)

If a volume has recently been initialized, the system can easily find a single disk extent that is large enough to satisfy your request. However, if the files have been created and deleted many times since volume initialization, the disk extents available for allocation may be scattered. In this case, the volume is said to be *fragmented*, because the system must allocate two or more smaller disk extents that have a total size sufficient to fulfill the request.

If a volume becomes fragmented, performance can be affected in several ways:

- The system requires more time to create or extend a file because it must access more sectors of the Bit Allocation Map to find enough disk extents to satisfy the request.
- The system requires more time to process a file sequentially because disk sectors that are logically consecutive are not necessarily physically consecutive.
- The number of files that can be open concurrently is reduced. This is because each open file requires allocation of a File Area Block in memory for each disk extent of that file.

This condition can be corrected by backing up the volume to an archive medium and reinitializing the disk. When the archived data is restored, files are located on contiguous disk extents. (For information about backing up and restoring files, refer to Section 8.)

Initialization Using ECC vs. CRC Formatting

When data is written to or read from a disk, error detection and correction operations are run on the data. These operations insure that the data is not corrupted, and that corrupted data is detected and corrected.

BTOS currently uses two methods of error detection and correction:

- Cyclical redundancy check (CRC). This is the original method of error detection and correction employed for disk data storage and retrieval in BTOS.
- Error correction code (ECC). Beginning with the 5.00.03 release of the MIVolume utility, this method is available on all versions of DPs and some newer versions of FPs. This method of error detection and correction has some performance advantages over the CRC method.

When a disk is initialized, it is formatted differently depending on whether CRC or ECC is being used. DPs and FPs that can support ECC will automatically initialize any disks under their control with an ECC format. These FPs and DPs can also still read from and write to disks formatted to support CRC.

However, the versions of FP boards that do not support ECC will not recognize an ECC-formatted disk to be a valid volume. For this reason, exercise care when initializing disks if you have a mixture of these different versions of FP boards in your system.

The disk device technical information generated by the MVolume Report utility indicates whether a disk was formatted for CRC or ECC. This utility is described later in this section.

Initializing XE 500 Volumes

To initialize XE 500 disks, use the BTOS MIVolume utility. In addition to formatting a disk, the MIVolume utility is used to

- Assign the volume name.
- Assign the volume password.
- Determine the maximum number of directories and files that can be created on the volume.
- Determine whether primary and secondary file headers, or just primary file headers are to be used.
- Assign a password to the <sys> directory.
- Determine whether the <sys> directory is to be write-protected.
- Add disk bad spots to the volume's bad spot listing.

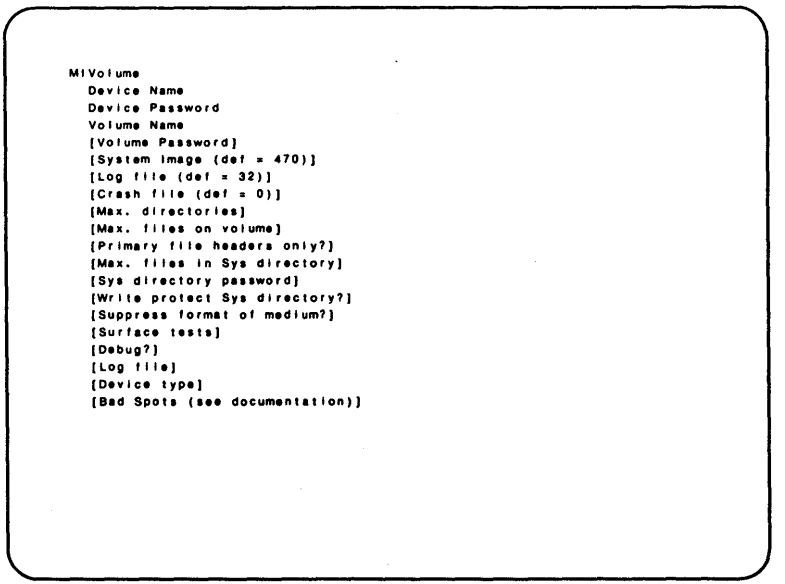
To initiate the **MIVolume** command, use the "Issuing BTOS Commands Through CENTIX" option of centrEASE (see the *centrEASE Operations Reference Manual*) or the CENTIX **ofcli** command (see Section 13).

If you are using centrEASE, the system displays the **MIVolume** command form shown in Figure 14-1. If you are using **ofcli**, you must enter these fields as part of the run statement.

You must enter parameters in three MIVolume fields:

- In the "Device name" field, enter the name of the device that contains the volume you want to format.
- In the "Device password" field, enter the password for the device. Unless modified by a user, the device password is the same as the device name.
- In the "Volume name" field, enter a name (a maximum of 12 characters) to identify the volume. This name must not duplicate any device or active volume name.

Figure 14-1 **MIVolume Command Form**



The **MIVolume** command form also has 16 optional fields (enclosed in square brackets). The defaults for these fields are set for a system volume. You can leave any or all of the fields blank to accept the defaults or enter parameters to override the defaults. Refer to Table 14-1 for information about each optional field.

When invoked, the MIVolume utility verifies the consistency of the parameters specified in the command form, opens a log file for the initialization operation (if a log file is specified), and displays the values chosen for the sizes of the volume control structures.

If MIVolume is initializing a disk cartridge, it prompts you to mount the cartridge.

Table 14-1 **MIVolume Command Form Fields**

Field	Action/Explanation
Device Name	Enter the physical device name of the disk drive to be initialized. Default names for XE 500 built-in disk drives are d0, d1, d2, and so on. Default names for SMD disk drives are s0, s1, s2, and so on.
Device Password	Enter the password (if one exists) for the drive device. When entered, it is not displayed to assure system security. The default password for a disk device is the same as its default device name.
Volume Name	Enter the name to be assigned to the volume. It can be up to 12 characters long and must not duplicate another device or volume name.
[Volume Password]	If no password is specified, the volume is unprotected. No password is necessary to access it, and none of its directories or files can have passwords. To assign a password to this volume, enter a maximum of 12 characters. Depending on the file protection level assigned, users may have to use this password when they create files or directories, or when they open files on this volume. You can assign or modify a volume's password later with the **MChange Volume Name** command.
[System Image (def = 470)]	This is the number of sectors required for an XE 500 system image [that is, the master operating system(OS) run file]. For the XE 500 system disk, the system image size should be at least 470 sectors. To initialize a nonsystem disk, enter 0. MIVolume creates an empty file, [Sys]<Sys>SysImage.sys, whose size is the specified number of sectors. You cannot install or copy the SysImage.sys file to an initialized volume without using MIVolume to allocate space for it. If the area for SysImage.sys is smaller than the SysImage.sys file being copied, the system displays an error message. You must reinitialize the disk to recover.

Table 14-1 **MIVolume Command Form Fields (Cont.)**

Field	Action/Explanation
[Log file (def = 32)]	Enter the number of sectors required for the system log file, [sys]<sys>log.sys. The default value is 32. The system log, displayed through the **MPLog** command, reports system status and error conditions. Specify a larger number if the log file tends to fill before it is convenient for you to print it. Log entries are written only to system disks. If this volume is not to contain the master OS, enter 0.
[Crash file (def = 0)]	Enter the number of sectors required for the system crash file. The system crash file is used to store a dump of the master processor's memory image when the system starts up or a system crash occurs. Allocate two sectors for each 1 kB of processor memory to be dumped. If a number is not specified, a system crash file is not written.
[Max. directories]	Leaving this field blank directs the system to set the maximum number of directories for this volume according to the volume size. To specify the maximum number of directories, enter a number. The number must be less than 65535.
[Max. files on volume]	Leaving this field blank directs the system to set the maximum number of files for this volume according to the volume size. To specify a maximum number of files, enter the number. Allow a sufficient number for future expansion; you cannot increase this number later without reinitializing the volume.
[Primary file headers only?]	The default (no) directs the system to allocate space for both primary and secondary File Header Blocks. To conserve disk space, enter "yes" to allocate space for a primary File Header Block only. This destroys the secondary file structure, but gains a large amount of disk space.
[Max. files in Sys directory]	Leaving this field blank directs the system to set the maximum number of files in the <Sys> directory, according to the volume size. To specify a maximum number of files other than the default, enter the number. Allow for more files than needed; file system performance degrades if the directory exceeds 80% capacity. Remember that, for the system disk, the system files reside in the <sys> directory.

Table 14-1 **MIVolume Command Form Fields (Cont.)**

Field	Action/Explanation
[Sys Directory password]	Enter a password that you want to assign to the <sys> directory, up to a maximum of 12 characters. If you enter a password, it will only be valid if you also specify a volume password. If you leave this field blank, no password is assigned and the files in it will be unprotected. If a password is entered, users will have to specify it when attempting to create or open files in the <sys> directory (depending on the protection level assigned to each file).
[Write protect Sys Directory?]	Enter "yes" to set the default file protection level of the files in the <sys> directory to "modify protected" (protection level 5). Leave the field blank or enter "no" to not have the default file protection level be "unprotected" (protection level 15). The protection will only be valid if you have also entered passwords for the volume and <sys> directory.
[Suppress format of medium?]	Enter "yes" to suppress formatting. Suppressing the formatting of the disk reduces the time to reinitialize a previously initialized valid disk. When initializing a new disk or reinitializing a corrupted disk, leave this field blank.
[Surface tests]	Enter the number of surface tests to be performed on the disk. A surface test writes data to and then reads the data from each sector of the disk to check for bad spots. The recommended number for disk cartridges and built-in disks is 3. The recommended number for SMD disks is 8. A surface test for a disk cartridge takes approximately 10 minutes; for a built-in disk, approximately 20-30 minutes; and for an SMD disk, approximately 20-45 minutes. If you want to shorten the time for the reinitialization of a noncorrupted disk, enter "0."
[Debug?]	No is the default. If you specify "yes," MIVolume generates information that is of interest to only Burroughs engineers. It also causes the utility to take more time to run.

Table 14-1 **MIVolume Command Form Fields (Cont.)**

Field	Action/Explanation
[Log file]	To create a print file of the MIVolume status report, enter a file name. This file can then be viewed at your terminal screen or printed. To automatically send the report to a printer and not save the print file, enter a print queue name, enclosed in square brackets. By default, the report is only displayed at the terminal screen.
[Device type]	Enter the device type of the disk that you are initializing. Valid type names are SYQUEST6 for a disk cartridge. ATASI46 for a 37.5 MB built-in disk. MICROPOLIS85 for a 71.3 MB built-in disk. TOSHIBA85 for a 72.2 MB built-in disk. MEMOREX166 for an SMD disk.
[Bad Spots (see documentation)]	For information about entering bad spots, refer to "Bad Spots" later in this section.

To avoid inadvertently destroying a volume that contains valid data, MIVolume reads the sectors in which a BTOS Volume Home Block (VHB) exists. If a VHB is found, MIVolume displays the information contained in it. This information includes the volume name, the date it was created, the date it was last modified, and the number of free pages and file headers. It also prompts for confirmation to reinitialize the volume.

Caution: *Reinitializing a valid volume causes all data stored on the volume to be lost. Before reinitializing a valid volume, always backup any files that should be saved.*

If you confirm the reinitialization and the old volume has a password, you are prompted to supply the password. MIVolume echoes a pound sign (#) for each password character entered. Terminate the password by pressing the RETURN key.

As the first step of initialization, MIVolume formats the medium and performs the number of surface tests specified in the command form. Each bad spot is recorded in the bad sector file ([sys]<sys>badblk.sys). When all surface tests have completed, the list of bad sectors is displayed. This list includes previously known bad spots, any bad spots detected during the surface tests, and bad spots entered in the command form.

After the bad spots are listed, MIVolume writes the volume control structures on the medium. The volume control structures include the files Mfd.sys, FileHeaders.sys, BadBlk.sys, DiagTest.sys; the Volume Home Block; and the Allocation Bit Map.

Note: *The DiagTest.sys file is reserved for diagnostics. It cannot be deleted, is not backed up during a backup operation, and should not be used to store data.The DiagTest.sys file is allocated on the second-to-last track of each XE 500 disk.*

Finally, MIVolume creates the system files SysImage.sys and CrashDump.sys. These two files are needed if the volume is to contain a master OS (that is, the XE 500 could boot from the volume). However, nothing is written to these two files at this time.

Guidelines for Initializing a Volume

The following list presents guidelines for using the MIVolume utility.

- MIVolume does not allow a volume currently in use to be initialized.
- If you are initializing a disk for the first time, you must enter the bad spots listed in the manufacturer's report that accompanies each disk.

  The bad spot report for disk cartridges is on a label attached to the top of the cartridge.

  For more information on how to enter bad spots, see "Entering Bad Spots" later in this section.

- Once a volume has been initialized, use the **MVolume Report** command to create a printable file of the updated bad spot listing. This file, or a printed copy of it, should be saved so that you have a current bad spot listing. You have to have this list in case the volume should become corrupted.

  You can use the information generated by **MVolume Report** to create a file whose name can be entered in the [Bad spot] field in place of entering each bad spot individually. See "Entering Bad Spots" later in this section.

- If you are initializing a disk that has become corrupted (that is, the disk can not be mounted by the system), you must enter all known bad spots.

- If you are intializing a volume for the first time or a volume that has been corrupted, you must specify a device type in the **MIVolume** command form.

  If you are reinitializing a valid volume, you do not have to specify the device type.

- If you interrupt the initialization operation, the volume must be considered invalid. The disk must be reinitialized as though it were a new disk.

- Certain **MIVolume** command form parameters should be used depending on how you want to use the volume. A volume from which you can boot the XE 500 is called a *system volume*. A disk cartridge that is to be used to archive files is called an *archive volume*. A volume that is to be used for data and program storage is called a *data volume*.

  Table 14-2 lists the applicable parameters and the values that should be used for the three types of volumes. If the default parameter value is recommended, leave the form field for that parameter blank.

Table 14-2 **Volume Types and MIVolume Parameter Values**

Parameter	System Volume	Data Volume	Archive Volume
Volume Name	sys	your choice	archive
[System Image (def = 470)]	(Default)	0	0
[Log file (default = 32)]	(Default)	0	0
[Max. directories]	(Default)	(Default)	1
[Max. files on volume]	(Default)	(Default)	10
[Max. files in Sys directory]	(Default)	(Default)	10

- If you are reinitializing a valid BTOS volume, you can suppress the medium formatting operation by specifying "yes" in the [Suppress format of medium?] field. This reduces the time required to initialize the disk.
- The value specified for the maximum number of directories is rounded up to the nearest multiple of 15 to produce the size of the file [sys]<sys>Mfd.sys. Since this file can be expanded only by reinitializing the disk, it is important to determine an appropriate value.
- MIVolume allocates File Header Block (FHB) sectors for each file. The value specified for the maximum number of files is multiplied by 1.5 to allow for a certain number of extension FHBs. This product is then rounded up to an even multiple of three times the number of sectors on each cylinder of the disk.

  MIVolume carefully locates these FHBs on the disk to ensure that the primary and secondary FHBs are located on different cylinders, at different rotational positions, and (for system volumes) on different disk surfaces.

  Since this number can be increased only by reinitializing the disk, it is important to determine an appropriate value.
- The value specified for the maximum number of files in the <sys> directory is rounded up to the nearest multiple of 14. Because system performance is degraded if the volume is a system volume and this directory is more than 80% full, allow for more files than needed.

  Since this number can be expanded only by reinitializing the disk, it is important to determine an appropriate value.

Bad Spots

If you are initializing a disk for the first time or a disk that has become corrupted (that is, the disk is no longer mountable by the system), you must enter a listing of known bad spots in the [Bad spots] field of the **MIVolume** command form.

A hardcopy listing of bad spots for each disk cartridge is attached to the top of the cartridge.

Once a disk has been initialized, an encoded list of the bad sectors is maintained in the file <sys>BadBlk.sys.

A sector is identified as defective (and therefore omitted from the Allocation Bit Map) for three reasons:

1 A bad spot in the sector was specified in the **MIVolume** command form.

2 Before initialization, the medium contained a valid BTOS volume and the sector was previously identified in the bad sector file as defective.

3 The surface tests of the MIVolume utility identified the sector as defective.

Identifying Bad Spots

Bad spots are identified by their location on the disk. A disk location is designated by cylinder/head/sector or cylinder/head/byte (a cylinder is sometimes referred to as a track).

Bad spot locations are designated by

c/h/#s or *c/h/b/*1

where

c is the cylinder number

h is the head number

s is the sector number

b is the byte number

1 is the length of the defect. You should always use the value "1" to specify the length of the defect.

All numbers used in identifying bad spots are in decimal.

The MIVolume utility converts all bad spot designations in the byte format to the sector format. When specifying bad spots in the **MIVolume** command form, you may prefer to use the sector format rather than the byte format.

Listing Current Bad Spots

To get a listing of bad spots, specify a log file or print file name in the **MIVolume, MDisk Verify,** or **MVolume Report** command form when you use either of these utilities.

The report generated by these utilities includes a listing of the bad spots currently stored in the <sys>BadBlk.sys file. You can use this listing to create a file whose name can be entered in the [Bad spot] field in place of entering each bad spot individually. The procedure for doing this is described in the next subsection.

Entering Bad Spots

To enter bad spots in the [Bad spots] field of the **MIVolume** command form, you can either enter each bad spot individually or enter the name of a file that contains a list of the bad spots.

To enter bad spots individually, type in each bad spot designation in the form field. The [Bad spots] field allows three lines for entering bad spots; to go to the next line, press the RETURN key. Do not break an individual bad spot entry over two lines.

If all the entries do not fit on the three lines, you must create a file using the BTOS Editor that lists the bad spots, separating the bad spot designations by a space or carriage return. Then, in the [Bad spots] field, enter

@BadSpotFile

where *BadSpotFile* is the file in which you have listed the bad spots.

If you have saved the print file generated when you executed the **MIVolume, MDisk Verify,** or **MVolume Report** command on the volume, you can edit this file to create a bad spot file. Use the following procedure:

1 Copy the print file to the file you will enter in the [Bad spot] field.

2 Use the BTOS Editor to delete all nonpertinent information from the print file.

  The only information that you want to keep in this file is the bad spot listing itself.

3 In the [Bad spot] form field, enter "*@BadSpotFile*," where *BadSpotFile* is the name of the file you edited.

Note: *If you are executing MIVolume through the* ***ofcli*** *run statement and are listing bad spots individually, the list of bad spot entries must be enclosed by parentheses.*

Notes about Bad Spots

- Built-in disk drives are formatted for 591 bytes per sector (512 of which are for data), 16 sectors (0-15), 7 heads (0-6), and 645 cylinders (0-644). A defect on a cylinder greater than 644 or having a byte value greater than 9455 cannot be entered. Also, do not enter bad spots for cylinder 644, head 6, because it is the last track on the disk and is reserved. Bad spot information in the form "degree *n*" should be ignored.
- Disk cartridges are formatted for 591 bytes per sector (512 of which are for data), 17 sectors (0-16), 2 heads (0-1), and 288 cylinders (0-287). A defect on a cylinder greater than 287 or having a byte value greater than 10046 cannot be entered.

Disregard any bad spots listed as being on cylinders beyond 287; these are on the inner areas of the disk, which are not used. Also, do not enter bad spots for cylinder 287, head 1, because it is the last track on the disk and is reserved.

- Two things should be noted about the defect reports attached to a disk cartridge. First, ignore any error count information appearing on the report (for example, "HD 0 ERR CNT 3"). Second, if there are numerous defects on a cylinder, the byte count is listed in the report as "misc." This means that the entire cylinder/head area of the disk should be designated as being defective. This is done by using the following bad spot designation:

  c/h

  where *c* is the cylinder number and *h* is the head number.

  For example, for the defect report in Figure 14-2, the corresponding bad spot entry is "275/1".

Figure 14-2 **Sample Disk Cartridge Defect Report**

```
Cyl      Hd      Byte
-------------------
275       1      misc.
```

Verifying Disk Integrity

The **MDisk Verify** command invokes a utility that provides testing of XE 500 disks (cartridge, built-in, and SMD disks) for soft and hard errors without destroying valid data on the volume. It can also be used to check the compatibility of disk cartridges with your cartridge drive.

The utility performs read operations of the entire disk. By default, if a defect is encountered, it is tested for three consecutive *soft* failures in 10 retries. Three consecutive soft failures constitute a *hard* failure. Known bad spots are excluded from testing. When testing has completed, the utility lists new bad spots. These bad spots are areas that caused either soft or hard errors.

You can optionally specify other values for the number of retries, how many failures constitute a hard failure, and whether the failures must be consecutive or just cumulative.

The utility generates a bad spot listing formatted such that a print file of the listing can be used to create or be added to a file for the **MIVolume** command form bad spot field.

MDisk Verify has the following additional options:

- Format the medium (like the MIVolume utility).
- Write data patterns to the disk. A default data pattern can be used or you can specify another pattern.
- If a data pattern is to be written to disk, it can be read and compared to provide extended testing.
- Test only selected areas of the disk.
- Include testing of known bad spots.

***Caution:** If the format or write options are specified, valid data on the volume will be destroyed, just as with the MIVolume utility. If you want to use these options, first backup the volume to an archive medium.*

To initiate the MDisk Verify utility, use the "Issue BTOS Commands Through CENTIX" option of centrEASE (see the *centrEASE Operations Reference Manual*) or the **ofcli** command (see Section 13).

If you are using centrEASE, the system displays the **MDisk Verify** command form shown in Figure 14-3. If you are using **ofcli**, you must enter these fields as part of the run statement.

In the Device Name field, enter the name of the device that contains the volume you want to format.

The **MDisk Verify** command form also has 14 optional fields (enclosed in square brackets). You can leave any or all the fields blank to accept the defaults or enter parameters to override the defaults. Refer to Table 14-3 for information about each optional field.

When you complete the **MDisk Verify** command form, press the GO key. The utility redisplays the command form fields, showing the values you have chosen for each field.

When you have filled out all of the fields and initiate **MDisk Verify**, the verification of the volume begins. You must respond to the prompts that appear during the initialization procedure.

If you are checking a disk cartridge, MDisk Verify first prompts you to mount the cartridge. Otherwise, the testing begins.

MDisk Verify reports its testing activity by returning a character for every 10 cylinders it checks. The character corresponds to one of the testing operations. Table 14-4 lists these characters and their corresponding test operations.

Figure 14-3 **MDisk Verify Command Form.**

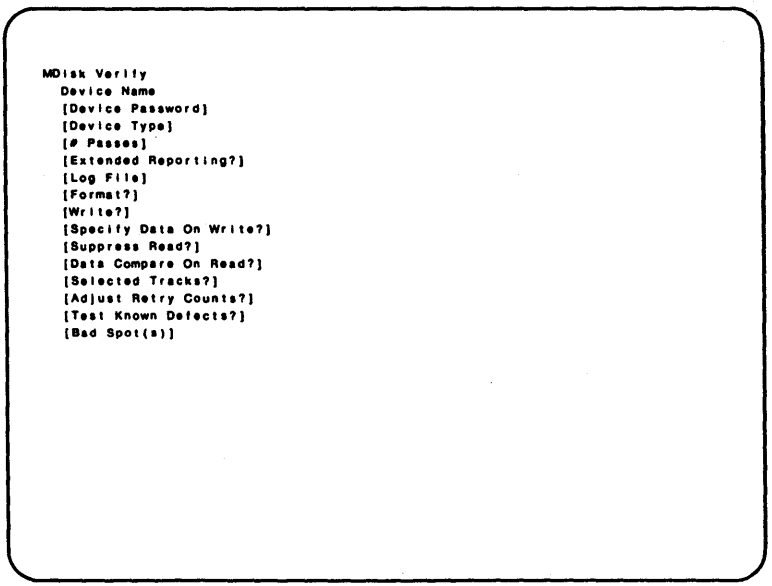


Table 14-3 **MDisk Verify Command Form Fields**

Field	Action/Explanation
Device Name	Enter the physical device name of the disk drive to be tested. Default names for XE 500 built-in disk drives are d0, d1, d2, and so on. Default names for SMD disk drives are s0 and s1 for base enclosure SMDs and s2, s3, and so on for MD3 SMDs.
[Device Password]	Enter the password (if one exists) for the drive device. When entered, it is not displayed to assure system security. If you leave this field blank, the password is assumed to be the name entered in the Device Name field.
[Device type]	Enter the device type of the disk that you are initializing. Valid type names are SYQUEST6 for a disk cartridge. ATASI46 for a 37.5 MB built-in disk. MICROPOLIS85 for a 71.3 MB built-in disk. TOSHIBA85 for a 72.2 MB built-in disk. MEMOREX166 for an SMD disk.
[# Passes]	Enter the number of Write/Read passes to be performed. The default is 1.
[Extended Reporting?]	Enter "yes" to cause all retry information to be displayed if an error is encountered and a character to be displayed for each cylinder tested. To only display test characters, leave the field blank.
[Log file]	To create a print file of the MDisk Verify status report, enter a file name. Do not specify a file on the disk being tested. This file can then be viewed at your terminal screen or printed. If the log file already exists, the new report is appended to it. If it does not exist, the log file is created. To automatically send the report to a printer and not save the print file, enter a print queue name, enclosed in square brackets. By default, the report is only displayed at the terminal screen.

Table 14-3 **MDisk Verify Command Form Fields (Cont.)**

Field	Action/Explanation
[Format?]	To format the disk before any Write or Read tests are performed enter "yes." To perform testing without formatting, leave this field blank.

Caution: *This option will erase the data on a valid volume. It cannot be used on the system disk or if the disk is in use by other users. You must initialize the disk using the MIVolume utility after the verification is complete.*

[Write?]	To cause a fixed data pattern to be written to the disk prior to being read on each pass, enter "yes." To use the default data pattern of bit-shifted variations of hex 6DB, leave this field blank.

Caution: *This option will erase the data on a valid volume. It cannot be used on the system disk or if the disk is in use by other users. You must initialize the disk using the MIVolume utility after the verification is complete.*

[Specify Data on Write?]	To be prompted before each pass for a 4 hex digit pattern to be written to the disk, enter "yes." To use the standard 6DB pattern for the Write and Data Compare operations, leave this field blank.
[Suppress Read?]	To suppress the Read operation on each pass, enter "yes." To have the Read operation performed before each pass, leave this field blank.
[Data Compare on Read?]	Enter yes to have all data read from the device to be compared to a known pattern. It is only useful if the Write option is selected.
[Selected Tracks?]	Enter "yes" to select only a portion of the disk to be tested. Before each pass, you will be prompted to enter the starting and ending cylinder and head. A single track is selected when the starting and ending locations are equal. To test the entire disk, leave this field blank.

Table 14-3 **MDisk Verify Command Form Fields (Cont.)**

Field	Action/Explanation
[Adjust Retry Counts?]	Enter "yes" to be prompted by the utility to specify the maximum number of retries to be attempted for each defect, how many failures constitute a "hard" failure, and whether the failures must be consecutive or can be cumulative. To test each defect for 3 consecutive failures in up to 10 retries, leave this field blank.
[Test Known Defects?]	Enter "yes" to include bad spots already listed in the volume's bad block list in the specified tests. To skip testing known bad spots, leave this field blank.
[Bad Spot(s)]	If you did not specify "yes" in the "[Test Known Defects?]" field, enter the known bad spots that you do not want to test. Refer to the information about entering bad spots, presented previously in this section.

Table 14-4 **MDisk Verify Report Testing Characters**

Testing Activity	Character Displayed
Formatting	F
Read only, no data compare	r
Read only, compare data	R
Write/Read, no data compare	t
Write/Read, compare data	T
Write only	W

Notes:

1 *MDisk Verify does not update the bad block list on a valid volume if new soft or hard errors are found.*

2 *MDisk Verify may be run on a write-protected disk cartridge if the device type is specified and Format and Write are not selected.*

Once the tests are completed, MDisk Verify returns a test report listing newly detected bad spots along with previously known ones.

If you are checking a disk cartridge, MDisk Verify prompts you for whether you want to test another disk cartridge using the same command form parameters.

A sample report from the MDisk Verify utility is shown in Figure 14-4. This utility was run on a disk cartridge. Default values were used for all fields except [Extended Reporting?], for which "yes" was specified. The verification tests did not detect any previously unknown bad spots on this disk.

Figure 14-4 **Sample MDisk Verify Report**

```
XE500 Disk Verify Utility B5.00.11/U.S.A.
May 23, 1986 9:53 AM                    PAGE  1


Device Name : d0
Device Type :
Number of Passes : 1
Extended Reporting? : YES
Log File : [spi]
Format? : NO
Write? : NO
Specify Data On Write? : NO
Suppress Read? : NO
Data Compare On Read? : NO
Selected Tracks? : NO
Adjust Retry Counts? : NO
Test Known Defects? : NO
0 Bad Spot(s) Entered

Please insert disk to be checked.
    (Press GO to confirm, CANCEL to deny, or FINISH to return to
     Executive)

Checking the disk on !d0

Pass:  1

rrrrrr

Rereading track at Cylinder: 5  Head: 0 by individual sectors.
 Sector: 0 is a known defect - not retried.
rrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr
rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr
rrrrrrrrrr
rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr
rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr
rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr rrrrrrrrrr
rrrrrrrrrr rrrrrrrrrr rrrrrrrr

Test result summary for valid volume:  archive
```

Figure 14-4 **Sample MDisk Verify Report (cont.)**

```
  Media Defect at Cylinder Head Sector Test Result:
  -------------- -------- ---- ------ ------------
  Known defect at      5    0      0   not tested.
  Known defect at     29    1      5   not tested.
  Known defect at     29    1      6   not tested.
  Known defect at     67    0      0   not tested.
  Known defect at     66    0      0   not tested.
  Known defect at     80    0      8   not tested.

Disk verification indicates 6 bad sectors.

Disk check complete.

Repeat test with the same parameters?
     (Press GO to confirm, CANCEL to deny, or FINISH to return
     Executive)
```

Obtaining Information about a Disk Device

To obtain technical information about a disk device and a listing of its bad spots, use the **MVolume Report** command through centrEASE or **ofcli**.

The technical information includes:

- The type of processor controlling the disk device (that is, the device class).
- The disk device unit number for that processor. The unit number is zero-relative.
- Whether the disk device is a disk cartridge drive (that is, whether it is removable media).
- Whether the disk was formatted to support the CRC or ECC data error correction and detection method.
- Cylinder, track, sector, and byte information.
- The maximum logical file address, which indicates the size of the disk in bytes.
- The step rate code, which is the step rate value used by the FP's or DP's disk controller circuit.
- The number of the write precompensation cylinder, which is the cylinder after which write precompensation is used.

The report lists bad spots in two ways:

- In tabular form, by cylinder, head, and sector.
- In the standard bad spot designation form. This listing can be used to create a bad spots file whose name can be entered on the "[Bad spot(s)]" field of the **MIVolume** command form.

You can execute the **MVolume Report** command on the current working volume if there is no volume password assigned by entering "MVolume Report" in the Executive command field and pressing the GO key. The technical information and bad spot listing for the volume's disk is displayed at the terminal screen.

To execute the **MVolume Report** command on a volume other than the current working volume, if there is volume password assigned, or if you want a readable file of the listing,

1 Enter "MVolume Report" in the Executive command field and press the RETURN key.

  The system displays the **MVolume Report** command form shown in Figure 14-5. Table 14-5 lists the fields in the **MVolume Report** command.

2 In the "[Volume or device name]" field, enter the device or volume name of the volume for which you want the bad spots listed.

3 In the "[Volume password]" field, enter the volume's password, if one exists.

4 To get an editable file of the listing, enter a file name in the "[Print file]" field.

  If you want to print out the listing automatically without creating a print file, enter a print queue name enclosed in square brackets.

5 When you have filled in the appropriate fields of the command form, press the GO key.

The system displays a listing of the volume's bad spots. If you specified a print file, an editable file of the listing is also created. If you specified a print queue, the listing is printed out.

A sample listing of bad spots is shown in Figure 14-6.

Figure 14-5 **MVolume Report Command Form**

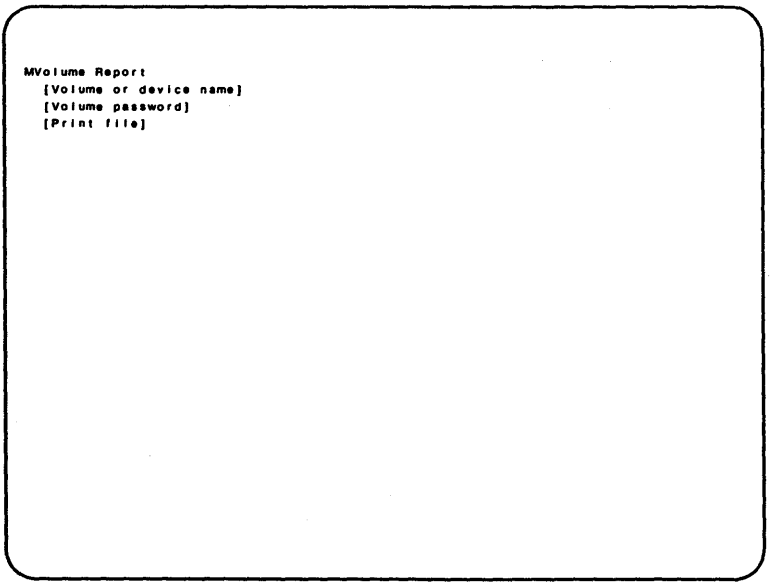


Table 14-5 **MVolume Report Command Form Fields**

Field	Action/Explanation
[Volume or device name]	Enter the name of the volume or of the disk device on which the volume is mounted. If this field is left blank, the bad spot listing for the current working volume is returned.
[Volume password]	Enter the volume password, if one exists.
[Print file]	Enter the name of the file to which the bad spot listing is to be written. Specify a print queue, enclosed in square brackets, to have the listing copied to a temporary print file and sent to the specified print queue. If you do not name a print file or print queue, the information is only sent to the terminal screen.

Figure 14-6 **MVolume Report Sample Bad Spot Listing**

```
Volume Reporter B6.00.00/U.S.A.
Aug 29, 1986 11:45 AM                PAGE  1


Device information for volume d1

   Device class                      : File Processor
   Unit number                       : 1
   Removable media                   : No
   Disk format                       : ECC mode
   Cylinders per disk                : 645
   Tracks per cylinder               : 7
   Sectors per track                 : 16
   Bytes per sector                  : 512
   Maximum logical file address      : 36986880
   Step rate code                    : 0
   Write precompensation cylinder    : 255

Bad spot information for volume: d1


  Cylinder  Head  Sector  # Sectors
  --------  ----  ------  ---------
    582       1       8          1
     41       1       0          1
     82       4       1          1
    508       3      10          1
    508       0       8          1

The MIVolume [Bad Spots] field format for these is:

  582/1/#8  41/1/#0  82/4/#1  508/3/#10  508/0/#8

Disk contains 5 bad sectors.
```

Running the System in Different Operating Modes

The XE 500 CENTIX system can be booted up using different sets of BTOS configuration files. The set of files used depends on the keyswitch position to which the key is turned when booting the system. These sets define the operating modes in which the system can run.

There can be four sets of system configuration files:

- A set used when the keyswitch is turned to MANUAL. The system configuration file names in this set include ".m" (for example, [sys]<sys>Master.m.cnf). This set is not provided in the standard software release.
- The set used when the keyswitch is turned to REMOTE. The system configuration file names in this set include ".r" (for example, [sys]<sys>Master.r.cnf). This set is provided in the standard software release. It is used to bring the system up in restricted mode.
- A set used when the keyswitch is turned to NORMAL. The system configuration files names in this set include ".n" (for example, [sys]<sys>Master.n.cnf). This set is not provided in the standard software release.
- The default set, which is the standard set of system configuration files created through the MBTOS Config utility. The files in this set do not have to include identifying character in their file names (for example, [sys]<sys>Master.cnf). These files are used if the corresponding .r, .m, or .n files do not exist.

The system files that are included in these sets are

- The master configuration file.
- Processor configuration files.
- Processor initialization files.
- The CENTIX file system configuration file.

- A CP, DP, SP, and TP version of BTOS, and the CENTIX operating system for the APs. [For master FPs and DPs, there is only one version of the master BTOS operating system (OS) used, [sys]<sys>SysImage.sys, regardless of the keyswitch position. The version of this file installed by the Boot Load utility is overwritten when system software is installed using the MSysLoad utility.]

There are also two CENTIX roots: the full root for normal mode and a modified smaller version for the restricted mode.

What Happens At Boot Time

By turning the keyswitch at the XE 500 base enclosure from STOP to MANUAL, REMOTE, or NORMAL, you initiate the XE 500 boot up process. The master FP first runs a series of self-diagnostic tests and loads the master OS, [sys]<sys>SysImage.sys. It then attempts to read the set of system files associated with the position to which the keyswitch is turned. These files include the identifying characters ".m" for MANUAL, ".r" for REMOTE, and ".n" for NORMAL. For each type of system file, if the system does not find the corresponding keyswitch-related file, it uses the default version of the file.

As part of the software installation procedure, two sets of system configuration files are installed:

- The set whose file names include ".r". This set is used to bring the system up in restricted mode.
- The default set of files created through the MBTOS Config utility.

Therefore, after following the standard software installation procedure, you can either

- Bring the system up in restricted mode by turning the keyswitch to REMOTE.
- Bring the system up in normal mode by turning the keyswitch to MANUAL or NORMAL.

Note: *Because having the keyswitch at MANUAL enables the RESET button, which could be accidentally pressed during system operation, it is recommended that you use the NORMAL keyswitch position.*

Using the Normal Mode

This manual defines the *normal mode* to be the state of the system when it is booted up using the default set of system files.

When the system is booted up in normal mode, you have use of the complete system, defined by the files installed through the MSysLoad utility and the system configuration files created or modified through the MBTOS Config utility.

If the keyswitch is turned to MANUAL or NORMAL and customized system files have been created for that keyswitch position (that is, files including the suffix ".m" or ".n," respectively), the customized files are used in place of the default files. Normal mode is not used. Refer to the subsection "Using a Customized Mode."

Using the Restricted Mode

Caution: *For you to be able to work in restricted mode, your system console must be a PT 1500 and it must be connected to the first CP in the system.*

The restricted mode is used

- During the initial system software installation.
- If you must reinitialize your system disk and/or reinstall system software in the event of system software corruption.
- To correct corrupted system files. For information about how restricted mode is used to correct corrupted system files, refer to Section 16.
- To move the root partition or swap files to improve system performance.
- To resize the root partition or swap files to improve system performance.

Capabilities of the Restricted Mode

The CENTIX restricted mode system software is installed into your system during the Boot Load utility. Once this software is installed, when you boot your system in the REMOTE keyswitch position, CENTIX boots up in the restricted mode.

The CENTIX restricted mode provides the following functions:

- Install system software.
- Restore archived system software.
- Create, move, or resize the normal mode CENTIX root.
- Create the swap area for AP00.
- Move or resize swap areas.
- Perform system repair operations to recover from system failure.

The CENTIX restricted mode has the following restrictions:

- The system comes up in single-user mode. Therefore, there can only be one person on the system in restricted mode.
- The restricted mode only supports the device assigned tty001. This device must be a PT 1500 and must be connected to the first CP (CP00).

  The standard software is configured so that a PT 1500 that is connected to CP00 is assigned tty001. (The configuration files support an lpr serial printer connected to RS–232 channel 3 of CP00. This printer is assigned tty000.)

System Configuration and Services

The restricted mode set of system configuration files (that is, the set of system configuration files that include the ".r" suffix) are listed in your *CENTIX Installation and Implementation Guide*.

The restricted mode master and processor configuration files define the following configuration:

- One master FP board (FP00).
- One CP board (CP00).
- One AP board (AP00).
- One SP board (SP00).
- One DP board (DP00).

This configuration means that, in restricted mode, only the first board of each board type boots up.

The restricted mode processor initialization files define the restricted mode BTOS system services to be as follows:

- The AdminAgent, the Temporary Directory Filter, the Queue Manager, and the CENTIX file system server running on the master FP.
- A Tape Server running on DP00 or SP00.
- An lpr printer spooler manager running on CP00.

Note: *If your system has both a DP and an SP and you are booting the system in restricted mode, you cannot predict whether the Tape Server will be installed on the DP or SP.*

To have the Tape Server installed on the processor to which the half-inch tape drive is connected, you must remove the Tape Server run statement from the restricted mode initialization file of the DP or SP to which the tape drive is not connected.

BTOS Error Logging

While in the restricted mode, all BTOS system errors and software installation status messages are logged in the file [!sys]<sys>log.sys. This is the same BTOS log file used during normal mode. These messages can be listed using the BTOS MPLog utility through centrEASE or **ofcli**.

CENTIX Shell Commands

The Centix Restricted Mode supports a subset of the CENTIX shell commands. Only those commands that are needed to support software installation and system repair are included. The following commands are present:

apnum	**find**	**ofed**
cat	**fsck**	**ofls**
CentixLoad	**grep**	**page**
chmod	**hd**	**pr**
chgrp	**init**	**pwd**
chown	**line**	**rm**
clear	**ln**	**rmdir**
console	**lpr**	**sh**
cp	**ls**	**sleep**
cpio	**mkboot**	**stty**
crup	**mkdir**	**su**
cut	**mkfs**	**sync**
date	**mklost+found**	**tee**
df	**mknod**	**true**
diff	**more**	**umount**
dircmp	**mount**	**uname**
echo	**mv**	**setmnt**
ed	**od**	**tar**
expr	**ofcli**	**test**
false	**ofcopy**	

The Shell

The full shell is available in the restricted mode. This allows you to write simple shell scripts. Please note that because the shell is included, those commands which are built into the shell are available (that is, set, shift, wait, and so on) The commands provided are sufficient for loading software and repairing a damaged root partition.

If your full root partition exists, you can mount it and make use of the commands within it by specifying the full path.

The following shell variables are defined:

```
HOME=/
PATH=/bin:/usr/bin:/etc
TERM=pt
```

Note that your PATH in the restricted mode does not include the current directory. It is set up this way so that only commands within the restricted mode root are run, unless you change the path or explicitly specify the full path. This limits the possibility of accidentally running potentially corrupted commands from within a corrupted root.

Caution: *Background processes are not supported in the restricted mode. It is impossible to kill such processes or determine their status. The shutdown function in the restricted mode Main Menu is based on the assumption that only one user is on the system and there are no background processes.*

If there are any background processes running when you shutdown from the restricted mode, the restricted mode root and/or the full root could become corrupted. In such an event, you may have to restore or reinstall your system software or restricted mode software.

Devices in the /dev Directory

Disk devices are supplied for partitions 0-7 on d0, d1, d2, d3, s0, s1, and s2. A special device, fpSAFrestore, is provided. This special device is used internally by the restricted mode in order to provide the Restore System Files function of the restricted mode Main Menu. You should not access this device.

Mount Point Directories

Three directories, /syq, /disk, and /fullroot, are supplied for use as mount points for partitions on disk cartridges or XE 500 fixed disks that you want to access from the restricted mode.

Partitions Used

The partition for the restricted mode root does not have much free memory space. There are only about 350 free blocks in this partition. It is recommended that you do not keep any files in the restricted mode root except files that are supplied and files needed during a system repair process.

The following partitions are used by the CENTIX restricted mode:

[sys]<sys>Partition.r.0	Restricted Mode root
[sys]<sys>Partition.r.1	Restricted Mode swap
[sys]<sys>Partition.r.2	Restricted Mode restore function

BTOS Utilities

Once you have installed system software, all of the BTOS utilities that are available in normal mode are also available in restricted mode.

Using the Restricted Mode

To run the system in restricted mode, use the following procedure:

1 Boot the system by turning the keyswitch from STOP to REMOTE.

A banner is displayed that tells you the system is running in the restricted mode. The system then prompts you to fill in the correct date and time.

2 Fill in the date and time and press RETURN.

The date and time should be entered in the format shown on the screen. If you make a mistake, you can use the BACKSPACE key to go back over the incorrect characters.

3 After the correct date and time have been entered, the system checks the two CENTIX file systems used by the restricted mode software, /dev/root and /dev/fpSAFrestore.

If the check fails, the system prompts you to run the CENTIX **fsck** command manually. The system then runs the superuser shell.

Enter the two **fsck** commands as follows:

```
# fsck /dev/root
# fsck /dev/fpSAFrestore
```

When these file systems have been successfully checked, enter

```
# exit
```

to exit from this shell. The system then displays the Restricted Mode Main Menu.

Caution: *Do not try to execute any other CENTIX commands from this shell. It is intended strictly for running* ***fsck*** *on the two restricted mode file systems.*

If the manual file check is not successful, reinstall the restricted mode software again.

---

4 The Restricted Mode Main Menu displayed:

```
        RESTRICTED MODE MAIN MENU
.....................................
..          0. Shutdown             ..
..          1. Restore              ..
..          2. Set up Software      ..
..          3. Go into Shell        ..
.....................................
```

The menu is followed by the prompt:

```
Please Select Option ==>
```

The Shutdown option causes the system to perform a simplified version of halt. No processes are killed and all file systems are unmounted. A message is displayed to tell you when you can stop or reboot the processor. Unlike shutting down from normal mode, the system does not return to single-user mode at the end of the shutdown.

The Restore option causes the Restore utility to be run. This utility is used to restore system files. This utility is described in the *centrEASE Operations Reference Manual.*

The Set Up Software option causes the Set Up Software Menu to be displayed. From this menu you can select the utilities that are used to install software.

The Go into Shell option runs the restricted mode shell, as previously described in this section. To leave this shell, use the **exit** command and you will be returned to the Restricted Mode Main Menu.

Using Customized Modes

This manual defines a *customized mode* to be the state of the system when it is booted up using a set of system configuration files that include ".m" or ".n".

If the keyswitch is turned to MANUAL or NORMAL and customized system files have been created for that keyswitch position (that is, the file names include ".m" or ".n," respectively), the system is booted with a configuration defined by those files.

Customized sets of system configuration files cannot be created through the MBTOS Config utility. They can be created by copying existing system configuration files and including the ".m" or ".n" suffix in the new file names. The contents of these files can then be modified with a text editor.

Customized processor operating systems can be created using the XE 500 BTOS Customizer. Refer to the *XE 500 BTOS Customizer Operations Guide*.

Analyzing and Recovering from Error Conditions

This section provides an overview of how to interpret system status, and recover from errors and failures in the XE 500 system environment. The section contains three parts:

- "Troubleshooting Tools" discusses the tools available through which status information is reported and problems can be analyzed.
- "Analyzing System Status" organizes system problems by functional groups. It includes information to help analyze problems as well as suggestions for how to recover from system errors or failures.
- "Recovery Tools" discusses the available tools and procedures to aid recovery from system errors or failure.

Caution: *Whenever you suspect that there is any damage in the CENTIX file system, you should run the CENTIX file system checking program, fsck. See Section 4 for details.*

Note: *To access the BTOS MCommands described in this section, use centrEASE (see the centrEASE Operations Reference Manual) or the CENTIX command* ***ofcli*** *(see Section 13 of this guide).*

Troubleshooting Tools

There are several troubleshooting tools that can aid in the recognition of system status, errors, and failures. These tools are listed below:

- The **front panel STATUS display** reports the status of software installation and system start-up operations.

- **BTOS status codes** are numerical codes reported by the system software through the BTOS Executive and other facilities. These codes indicate the status of system operations. The meaning of the codes are provided in the *BTOS Status Codes Reference Manual*. Also, through the CENTIX **perc** command, you can have a short description of a BTOS status code displayed at the workstation terminal.
- **System crash status words** are normally displayed at the terminal screen and entered in the system log after a system crash occurs. They can be useful in isolating the cause of a system crash.
- The **system log** provides a detailed account of system status, errors, and failures. This log can be listed by using the BTOS **MPLog** command.
- **Installation logs** provide a record of software installation operations.
- A **crash dump** is a copy of the memory image of a processor at the time of a nonrecoverable failure (crash). Such a file can be used by Burroughs personnel to analyze the cause of a processor failure.
- The **processor board rear panel LEDs** display hardware errors during system start-up and software errors that occur during normal system operation.

Front Panel STATUS Display

The XE 500 base enclosure's front panel STATUS display is the display through which the status of software installation and system start-up operations are reported.

The STATUS display shows a series of two-character codes that indicate the progress of the boot-up procedure. These codes are defined in Appendix B.

If the unit is operating correctly, the STATUS display code sequence stops at 20, indicating that the booting procedure is complete and the system is ready to run. If codes above the value of 20 are displayed, a failure preventing the system from booting up properly has occurred.

The boot-up process normally takes about a minute. While waiting for a bootable disk to become ready, the STATUS display alternates between "A1" and "04."

Display codes from 21 to 38 are listed in Appendix B.

Display codes 39 through 45 indicate that a failure has occurred during a processor booting operation. The code indicates the type of processor which failed. In order to further isolate the type of failure associated with these STATUS display error codes, observe the processor LEDs located on the processor board's rear panel panel. The interpretation of these codes is described under "Processor LED Codes," later in this section.

If code DE alternates with 41, 42, 43, 44, or 45, the master processor has not received a response from processors in expansion enclosures.

BTOS Status Codes

BTOS status codes are numerical codes reported by the system software to indicate the status of system operations. The meaning of the codes are provided in the *BTOS Status Codes Reference Manual*. Status codes can be reported to:

- The terminal screen.
- The system log.
- Software installation logs.
- Processor LEDs.

You can have a short description of a BTOS error code displayed at your terminal screen by using the CENTIX **perc** command. For details on using **perc**, see the *CENTIX Operations Reference Manual*.

System Crash Status Words

System crash status words can be useful in isolating the cause of a system crash. They are entered in the system log after a system crash occurs.

The words are numbered 1 through 8 and their values are in hexadecimal (hex). Depending on the corresponding BTOS error code, certain status words are either unused or dependent on the BTOS error code. Refer to Appendix B for a listing of the status words and their meaning.

The contents of the general status register (GSR) can also be helpful when isolating the cause of a system crash. The contents of the GSR are reported to status word 3 when a system crash occurs. Refer to Appendix B for a listing of the bits in the GSR and their meaning.

As an example, suppose the following error code and status words were reported as a result of a system crash:

```
SYSTEM CRASH - (ERC = 22)
Crash Information: 0016H 0008H DBA3H 0000H 0075H 0000H
0301H 0009H
```

The BTOS status code indicates ERC 22, a bus time-out. The information contained in the status words are as follows:

- Word 1 shows the hex equivalent of decimal 22 (that is, hex 16).
- Word 2 indicates that the failure occurred in process 8.
- Bit 10 of word 3 indicates a bus time-out.

  Bit 6 indicates that the trapped address was generated by the central processing unit (CPU) local to the processor at which the error was detected.

  Bits 1 and 0 indicate that the trapped address was greater than or equal to hex C0000.
- Word 5 indicates that the processor in slot 75 was being referenced.
- Word 6 indicates that a memory address lower than hex FFFF (64 kB) was being referenced.
- Words 7 and 8 indicate that the instruction preceding the instruction at address 03010:0009 was being executed when the time-out occurred.

The possible causes for the error could be isolated to a hardware failure of the processor in slot 74 or 75, or to a programming error.

The System Log

The system log, [sys]<sys>log.sys, provides a detailed account of system errors and failures. It can be listed by executing the BTOS **MPLog** command. The **MPLog** command provides a listing of operation status and error reports. Each instance of certain operations and of all system errors have a separate report. The form of the report depends on the type of the corresponding operation or error. The types of reports include:

- System crashes.
- Bootup (system start-up) status and errors.
- System initialization status and errors.
- Disk I/O errors.
- Cluster communication errors.
- ISAM errors.
- Tape operation status and errors.

Listing the System Log

To obtain a listing of the system log, use the **MPLog** command. The **MPLog** command form allows you to exercise the following options:

- Specify that only errors of a particular type be listed.
- Specify a print file to which the listing is copied or have the listing printed automatically.
- Specify a log file on a volume other than [sys] from which you want the log file information taken.
- Only list those error reports that occurred after a specified date and time.

To execute the **MPLog** command without exercising any of the options, enter "MPLog" in the Executive command field and press the GO key. The error reports in the log file are displayed at the terminal screen.

To execute the **MPLog** command with one or more of the options, enter "MPLog" in the Executive command field and press the RETURN key. The system displays the **MPLog** command form shown in Figure 16-1. Table 16-1 lists the fields in the **MPLog** command. When you have filled in the appropriate fields of the command form, press the GO key. The error reports in the log file are displayed at the terminal screen as specified in the command form.

Figure 16-1 **MPLog Command Form**

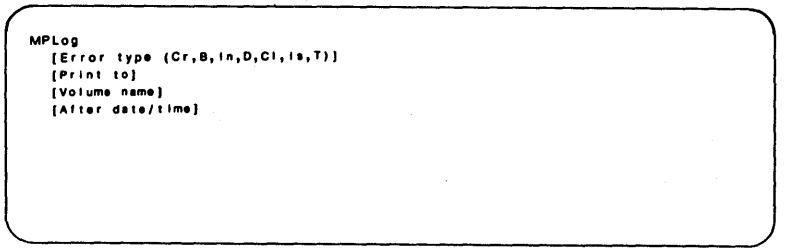


Table 16-1 **MPLog Command Fields**

Field	Action/Explanation
[Error type (Cr,B,In,D,Cl,Is,T)]	Enter the code corresponding to the type of errors that you want to list. Only one code can be entered. The codes are: Cr = System crash B = System boot In = System initialization errors D = Disk errors Cl = Cluster communication errors Is = ISAM errors T = Tape operations status and errors If you leave this field blank, all error types are reported.

Table 16-1 **MPLog Command Fields (Cont.)**

Field	Action/Explanation
[Print to]	To create a print file of the MPLog report, enter a file name. This file can then be viewed at your terminal screen or printed. To automatically send the listing to a printer and not save the print file, enter a print queue name, enclosed in square brackets. By default, the listing is only displayed at the terminal screen.
[Volume name]	Enter the name of the volume from which you want the log file information taken. The default directory and file name is <sys>log.sys. The default is [sys]. A Log file is maintained by the operating system only on the [sys] volume, and only if it was specified in the **MIVolume** utility when the [sys] volume was initialized.
[After date/time]	To only list errors that occurred from a specific date and time, enter that date and time here. If a time is not specified, the time is assumed to be 12:00 am (midnight). If you leave this field blank, all specified errors are listed.

System Crash Reports

Sample system crash error reports are shown in Figure 16-2. This error report can contain the following information:

- The name of the volume on which the log file is located is [sys].
- The date and time of the failure.
- The BTOS status code related to the failure.
- A short description of the type of failure.
- The memory location of the instruction immediately following the instruction being processed when the failure occurred.

- Eight hexadecimal system crash status words related to the failure. These status words can be used to isolate the cause of the failure. An explanation of how to interpret the status words is provided previously in the subsection "System Crash Status Words."
- Information about the processor at which a failing operation was executed. This information can include the XE 500 processor type, memory size, the user signed onto the terminal, and the operating system from which the processor was booted.

System Boot Reports

Sample system boot (start-up) reports are shown in Figure 16-3. This report identifies the following information:

- The type of XE 500 processor that was booted.
- The slot location of the processor.
- The memory size of the processor.
- The date and time that the processor was booted.
- The operating system from which the processor was booted.

Figure 16-2 **Sample MPLog System Crash Report**

```
File Processor in slot 71H
Memory Size: 768K,
SYSTEM CRASH - (ERC = 25)                    Thu Aug 28, 1986 4:48 AM
Description: Unknown non-maskable interrupt detected
Executing instruction preceding location 03B5:009C
Crash Information: 0019H 0022H 7F98H 0000H 0071H 0000H 03B5H 009CH
Os Booted: FpBtos-B5.01.00/U.S.A.
```

Figure 16-3 **Sample MPLog System Boot Reports**

```
Storage Processor with Storage Controller (DP) in slot 62H
Memory Size: 768K,
SYSTEM BOOT -                                  Fri Sep 5, 1986 4:19 PM
Os Booted: DpBtos-B5.01.00/U.S.A.

Cluster Processor in slot 73H
Memory Size: 768K,
SYSTEM BOOT -                                  Fri Sep 5, 1986 4:19 PM
Os Booted: CpBtos-B5.01.00/U.S.A.

File Processor in slot 71H
Memory Size: 768K,
SYSTEM BOOT -                                  Fri Sep 5, 1986 4:19 PM
Os Booted: FpBtos-B5.01.00/U.S.A.
```

System Initialization Error Reports

A sample system initialization error report is shown in Figure 16-4. This error report can contain the following information:

- The name and memory size of the XE 500 processor that detected the initialization failure.
- The date and time of the failure.
- The initialization status; this is technical information useful to Burroughs engineers.

Figure 16-4 **Sample MPLog System Initialization Error Report**

```
File Processor in slot 71H
Memory Size: 768K,
SYSTEM INITIALIZATION ERROR -                  Thu Aug 28, 1986 4:48 AM

Initialization Status: 2000H
```

Disk I/O Error Reports

A sample disk I/O error report is shown in Figure 16-5. This error report can contain the following information:

- The name, memory size, and slot location of the processor that controls the disk with the I/O error.
- The unit number of the disk.
- The name of the disk's volume.
- The BTOS status code related to the error.
- A short description of the type of failure.
- The date and time of the failure.
- The number of times the I/O operation was retried before the error was deemed unrecoverable.
- The disk location that was being written to or read when the I/O error was detected. This location can often indicate a previously unidentified bad spot on the disk.
- The command, main status, error status, and miscellaneous status fields contain technical hardware information useful to Burroughs engineers.

Figure 16-5 **Sample MPLog Disk I/O Error Report**

```
File Processor in slot 71H
Memory Size: 768K,
DISK ERROR - XE500 Winchester Unit 3 (ERC = 305)  Wed Aug 27, 1986 3:33 PM
Description: Sector field not found (bad Cmd or bad track)
Number of Retries: 0 (Unable to Recover),  Volume Name: c60345
Cylinder: 322,  Head: 0,  Sector: 0,  Number of Sectors: 0
Command: 70 38 01 42 00 00 00
Main Status: 51   Error Status: 10   Misc Status: FC
```

Cluster Communication Error Reports

A sample cluster communications error report is shown in Figure 16-6. This error report can contain the following information:

- The name, memory size, and slot location of the CP that detected the failure in cluster communications.
- The BTOS status code related to the error.
- A short description of the type of failure.
- The date and time of the failure.
- The CP's line number and cluster terminal identification number for which the error was detected.
- The user signed onto the cluster terminal.
- The secondary status contains technical hardware information useful to a Burroughs engineer.

Figure 16-6 **Sample MPLog Cluster Communications Error Report**

```
Cluster Processor in slot 73H
Memory Size: 768K,
XE500 CLUSTER ERROR - (ERC = 8100)          Fri Sep 5, 1986 5:42 PM
Description: Cluster workstation timed out
Line: 0,  ID: 2
Secondary Status: 01 04 04 02 91      SignOn User Name: kitsel
```

ISAM Error Reports

An Indexed Sequential Access Method (ISAM) error report includes a CENTIX ISAM error code. This error code indicates the type of problem that occurred while running ISAM. Refer to the *CENTIX ISAM Operations Reference Manual* for information about codes specific to ISAM.

Tape Operations Status and Error Reports

Tape status reports indicate the installation status of a tape server. They can contain the following information:

- The name, memory size, and slot location of the processor in which the tape server was installed.
- The date and time of the installation.
- The size, in bytes, of the tape server's tape buffer.
- The number of tape buffers allocated for the tape server.

A sample tape operation status report is shown in Figure 16-7.

Figure 16-7 **Sample MPLog Tape Operations Status Report**

```
Storage Processor with Storage Controller (DP) in slot 62H
Memory Size: 768K.
SUCCESSFUL TAPE SERVER INSTALLATION        Fri Sep 5, 1986 4:19 PM
Size in bytes of each Tape Buffer: 65534
Number of Tape Buffers allocated: 2
```

Tape error reports indicate errors that occur during tape operations. They can contain the following information:

- The processor on which the affected tape server is running.
- The half-inch or QIC tape drive unit number.
- The applicable BTOS tape error code.
- The current status of the tape drive, such as the current tape position.

A sample tape operation error report is shown in Figure 16-8.

Figure 16-8 **Sample MPLog Tape Operations Error Report**

```
File Processor in slot 71H
Memory Size: 768K.
TAPE ERROR - Qtr Inch Cartridge Unit 0 (ERC = 9024)  Fri Sep 5, 1986 4:19 PM
Description: Tape Hardware Error.
Current Error conditions / Status:
Beginning of Tape
Illegal command
Status bytes: C8H 00H
```

Crash Dumps

A system crash always writes a crash message to the system log file. The keyswitch position when a system crash occurs determines other system actions. If the keyswitch is set to NORMAL when a fatal error occurs, the default system action is to dump as much as possible of the failing processor's current memory image to a crash file. This file exists only if you create it. If you turn the keyswitch to MANUAL, you disable the automatic crash dump.

The crash dump file can be analyzed to determine the cause of a software problem. The crash dump is not printed; it is a core dump which is written to disk. You can use the **Dump** command to get a printable hex listing of the processor's memory. The **Dump** command also allows two dump files to be compared and generates a listing of their differences.

Note: *The* ***Dump*** *command is used to list the contents of a dump file; it is not used to create a dump file.*

So, for example, a dump of a suspected corrupted processor operating system can be compared to that of a known good one; the corrupted areas would be indicated by those areas that are different between the two operating systems.

The dump file for each processor has a unique name. The dump file for the master processor is [sys]<sys>crashdump.sys. The naming convention for all other crash files is:

```
[sys]<sys>xpyy.crash
```

where *xp* is the processor type (FP, CP, SP, TP, or DP) and *yy* is the processor number.

The crash file for the master processor can be created when the BTOS system disk is initialized. To create the crash file for the master processor, you must enter a value for the size of the crash file in the "[Crash file]" field of the **MIVolume** command form. The size of the crash dump file should at least be equal to the size of the master processor's memory.

All crash files other than crashdump.sys must be created manually in the [sys]<sys> directory. The file size must be at least equal to the total memory available to the processor. For example, if CP00 has 768 kB of memory, make [sys]<sys>Cp00.crash a 768-kB file.

Status Codes on Processor LEDs

If a processor detects a failure in the system start-up operations or crashes due the detection of a nonrecoverable error, the appropriate error code is displayed at the processor board's rear panel LEDs. Note that this processor is not necessarily the one that caused the failure. If the master processor is the affected board and the base enclosure keyswitch is at NORMAL, the master processor attempts to reboot the system instead of displaying the crash code.

When a processor is booted and operating normally, LED 0 (the heart beat LED) blinks on and off at a steady rate. In addition, for AP boards, the RUN LED (hardware run light) is lit, indicating that the AP's CPU is running.

To determine whether the LEDs represent a hardware boot status code or a software crash code, observe the LED lights. If the LED pattern is constant, the code displayed is a hardware boot ROM status code. If the LEDs display the walking pattern cycled with other code displays, the codes displayed between the walking pattern sequences make up a software crash hex code. For either the hardware or software crash codes, LED 0 (the heart beat LED) is not pulsing off and on.

Waiting to Boot

Once a nonmaster processor successfully completes its onboard ROM tests, it waits to be booted by the master or reset. This state is indicted on the LEDs by a "walking" pattern.

For processors not connected to another board, this pattern is one LED on walking through the other LEDs, which are off.

For processors attached to an ME board, the pattern is two nonadjacent LEDs on walking through the other LEDs, which are off.

For an SP attached to an SC board to form a DP, the pattern is two adjacent LEDs on walking through the other LEDs, which are off.

For an SP attached to an SC board and an ME board, the pattern is three adjacent LEDs on walking through the other LEDs, which are off.

Boot ROM Status Codes

Boot ROM status codes are displayed at the master processor's LEDs. These codes are listed in a table in Appendix B. Figure 16-9 shows how to derive the two-character hex code from the LEDs.

Crash Codes for BTOS Processors

After a processor crash, the affected processor enters a loop that repeats a crash display cycle on its LEDs. LED 0 (the heart beat LED) no longer pulses. The processor stays in this loop until the system is reset.

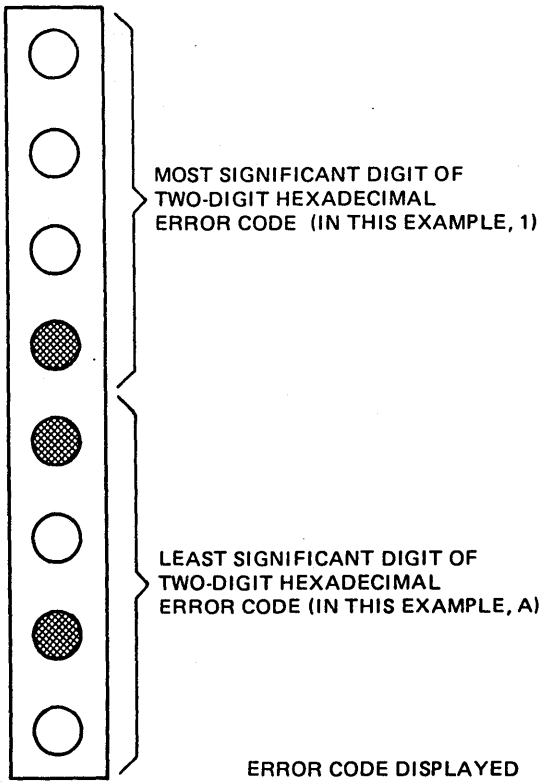
The crash display cycle begins with four cycles of a single LED, which is off, walking through the other LEDs, which are on. These cycles indicate the start of a crash code display.

The four walking cycles are followed by four error code display cycles. Each error code cycle displays one nibble (four bits) of the error code. This nibble represents one hex digit of the four-digit code.

During an error code display cycle, the top four LEDs always display binary 1000 [that is, the SE (system error) LED is on, LEDs 4, 5, and 6 are off].

The bottom four LEDs display the four hex codes, in sequence. LED 0 is the least significant bit of the nibble value. The first nibble displayed is the most significant nibble of the hex error code. At the end of each nibble display, all LEDs blink off to indicate that the next nibble will follow. After the last nibble is displayed, the entire crash display cycle is repeated.

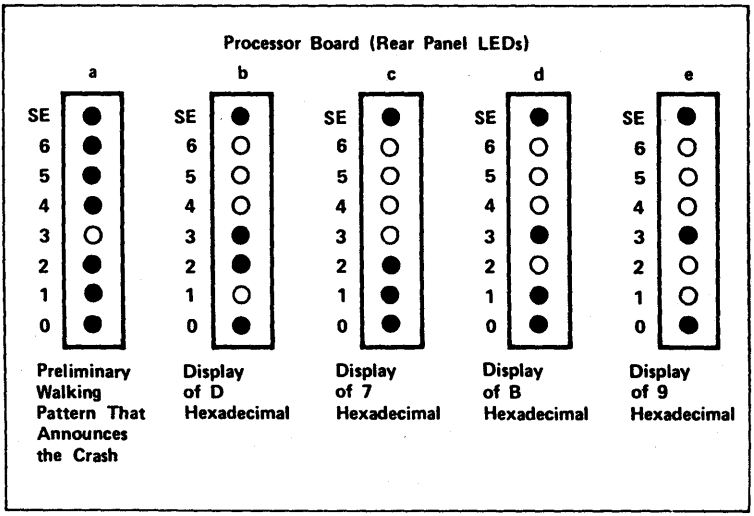
Figure 16-9 **Reading Boot ROM Error Codes on Master Processor LEDs**



To record a crash error code, wait for the beginning of the crash display cycle (the four walking cycles). Then record the value of each nibble as it is displayed. Repeat the procedure to verify that you have recorded the error code properly. For example, if a processor displays, in order, D, 7, B, and 9, the error code is hex D7B9. Figure 16-10 shows a sample sequence of this crash code display.

If the code being displayed is a BTOS crash code, you must convert the code to decimal. You can then look up the meaning of the code in the *BTOS Status Codes Reference Manual*.

Figure 16-10 **Sample Processor LED Crash Code Sequence**



E7558

Crash Codes for Applications Processors

During normal operations, the AP RUN LED is on; LED 5, the panic LED, is off; and LED 0, the heart beat LED, pulses on and off.

If a software crash occurs, the AP enters a loop that repeats a crash display cycle on its LEDs. The RUN LED is off, LED 5 is on (indicating a panic has occurred) and LED 0 (the heart beat LED) no longer pulses. The processor stays in the crash display loop until the system is reset.

The crash display cycle is displayed using LEDs 0 through 4. It begins with three cycles of a single LED, which is off, walking through the other four LEDs, which are on. These cycles indicate the start of a crash code display.

The three walking cycles are followed by five error code display cycles. Each error code cycle displays one nibble (four bits) of the error code. This nibble represents one hex digit of a five-digit CENTIX panic string code.

During an error code display cycle, LED 4 is lit, representing that a crash code nibble is being displayed.

LEDs 0 through 3 display the five hex codes, in sequence. LED 0 is the least significant bit of the nibble value. The first nibble displayed is the most significant nibble of the hex error code. At the end of each nibble display, all LEDs blink off to indicate that the next nibble will follow. After the last nibble is displayed, the entire crash display cycle is repeated.

To record a crash error code, wait for the beginning of the crash display cycle (the three walking cycles). Then record the value of each nibble as it is displayed. Repeat the procedure to verify that you have recorded the error code properly. For example, if a processor displays, in order, 2, 7, E, F, and D, the error code is hex 27EFD.

If you are able to run CENTIX on an AP after the crash, you can use the following procedure to have CENTIX return an error message that corresponds to the panic string.

Caution: *This procedure may not work on all systems. The message "bad command" may be returned when attempting to issue the "*nnnnn *?s" debugger command.*

---

1 From the CENTIX shell, enter

```
$ adb /unix
```

and press the RETURN key. This turns on the CENTIX **adb** debugger.

2 Enter

```
nnnnn ?s
```

where *nnnnn* is the five-digit panic string. Then press the RETURN key. The **adb** debugger displays the panic string's corresponding error message.

3 To exit the debugger, enter

```
$q
```

and press the RETURN key. The shell prompt is displayed again.

Analyzing System Status

This subsection organizes system problems by functional groups. It includes information to aid problem analysis and suggestions for how to recover from system errors or failures.

System Start-Up

To diagnose system start-up problems, it is helpful to know the start-up process. The next subsection discusses the operations that occur at system start-up.

What Happens during System Start-up

Upon being powered on, the XE 500's processors perform a series of self-tests. These tests are controlled by instructions in their read-only memory (ROM). The tests include:

- ROM readability.
- Memory initialization for error correcting code (ECC).
- Memory test of all accessible random access memory (RAM), including RAM on Memory Expansion (ME) boards.
- I/O tests on remote reference registers.

In addition to these tests, the master processor (that is, the FP or DP in the first processor slot of the base enclosure) performs an extensive set of tests to verify that the I/O paths to the disks it controls are working properly. All other processors except the master

- Program their interrupt systems.
- Set up a stack in high memory.
- Unmask the doorbell interrupt (Int 1).
- Mark a flag in the CPU description table that determines whether or not the processor will perform a core dump to a crash file in the event of a detected unrecoverable error.
- Enter an enabled loop waiting to have their operating systems sent by the master processor.

When the master processor has completed its self-tests, the booting process begins.

The master processor reports the status of the start-up procedure by displaying codes at the front panel STATUS display. If a hardware failure occurs, the appropriate codes are displayed in the STATUS display and on the processor LEDs of the processor that detected the failure.

The master processor looks for its operating system, [sys]<sys>SysImage.sys, sequentially through the disk units in the system. For example, in a system with a disk cartridge drive and built-in disks, it looks at the disk cartridge drive (d0) for [sys]<sys>SysImage.sys first; if it does not find the file there, it looks on disk drive d1; and so on.

Upon finding the file [sys]<sys>SysImage.sys, the file is copied into the master processor's memory. The master processor reads its configuration file to establish its disk I/O device parameters. Its initialization file is then executed to install the appropriate system services.

This master operating system then takes over the task of loading the rest of the processors with their operating systems. The master operating system reads the master configuration file, [sys]<sys>Master.cnf, to determine which processors should be booted, the operating system with which they should be booted, and, for some processors, their configuration files.

As each processor is booted, it reads its configuration file to establish its I/O device parameters. Its initialization file is then executed to install the appropriate system services.

When all processors have been booted and their system services installed, the system enters normal operation.

System Start-Up Problems

The master processor or the system disk is the most likely cause of most start-up problems. A bootable disk cartridge can be used to determine whether the master processor or the system disk is at fault. If the system starts successfully from the bootable disk cartridge, the problem is likely to be the system disk. If the system does not boot from the bootable disk cartridge, the master processor could be at fault.

If you have booted from a disk cartridge, you can review the system log, using the **MPLog** command, for ERC 301's on the system disk. ERC 301's are frequently logged in connection with boot problems. They represent disk I/O errors, which may be caused by an unreadable disk medium (bad spots), unstable input power, or improper drive termination.

Problem: Master processor freezes while attempting to load operating system to a processor. The master processor LEDs display an error code, but no error is entered in the system log. Some or all of the other processors have not booted (that is, one LED "on" cycles through the processor LEDs).

Possible causes.

- Master processor is faulty.
- Two processor boards are jumpered together by a processor-to-ME board interconnect bus.
- The backplane was damaged during the recent installation of a board.
- Power supply is malfunctioning.

Recommended action. Contact your Burroughs field service representative.

Problem: System fails to boot; 21 appears in the STATUS display.

Possible causes.

- The master processor is faulty.
- Disk drive cables (data or control) are faulty.
- The disk on which the master operating system ([sys]<sys>SysImage.sys) is stored is faulty.
- Another processor in the base enclosure is faulty.
- For a base enclosure with a QIC tape drive, the first built-in disk drive is not properly terminated.
- The master operating system is corrupted.

Recommended action. A 21 in the STATUS display means no **valid** master operating system can be found on the first disk recognized to have space allocated for the file [sys]<sys>SysImage.sys. This allocation happens when the disk is initialized. The system may be unable to find a valid master operating system for many reasons.

A subsequent successful system start-up from a bootable disk cartridge suggests that either the system disk is faulty, or the master operating system on the system disk is either corrupted or not present. If you suspect that the BTOS file system is corrupted, refer to the recovery procedure under "Determining Corrupted System Files" later in this section.

If a "No such device" (ERC 215) is returned when looking at the system disk, a fundamental hardware problem exists between the master processor and the system disk. Contact your Burroughs field service representative.

If an attempt to boot from a disk cartridge results in a display of 21 on the STATUS display, a drive cable or the master processor could be faulty. Contact your Burroughs field service representative.

Problem: System freezes during start-up; 50 appears on the STATUS display.

Possible causes.

- Syntax error in a processor initialization file, making the run statements in the file unexecutable.
- Partially corrupted BTOS file system.

Recommended action. Boot the system in restricted mode or, if possible, from a bootable disk cartridge and verify that the processor initialization files are correct. If they are in order, use the recovery procedure under "Determining Corrupted System Files" later in this section.

Problem: A processor board crashes during system boot; the board flashes a crash code on the its rear panel LEDs.

Possible causes.

- Corrupted system files.
- System disk is faulty.
- Processor board that crashed is faulty.
- For AP boards, the crashed AP board could not find the needed swap space on a disk.

Recommended action. If corrupted system files are suspected, use the recovery procedure under "Determining Corrupted System Files" later in this section. Check the operating system file that is loaded into the crashed processor.

For an AP crash, check that the swap partition for the AP exists on the disk specified in the file [sys]<sys>ConfigUFS.sys.

Problem: Code DE alternates with code 41, 42, 43, 44, or 45.

Possible causes.

- An expansion enclosure is not turned on.
- A bus repeater board or the cabling between the bus repeater boards is faulty.

Recommended action. Make sure all expansion enclosures are powered on.

If all expansion enclosures are powered on and code DE is still being displayed, contact your Burroughs field service representative.

Problem: System is unusually slow.

Possible causes.

- System disk is fragmented.
- Memory expansion is not recognized by the system.
- System disk is busy with recoverable read errors.
- Insufficient number of free blocks exist on the disk for context file use (disk full).
- X, Y, or Z blocks require better allocation to match processor I/O demands.
- A customization of BTOS processor operating systems is required to better allocate system resources.

Recommended action. Perform a backup, reinitialization, and restore of all disks believed to be fragmented (see Section 14 on volume fragmentation). If a disk is full, move or delete files from it.

Check the system log, using the MPLog command, to see that all processors with Memory Expansion (ME) boards have the proper amount of memory. If not, contact your Burroughs field service representative.

Use the CENTIX **pstat** command (see the *CENTIX Operations Reference Manual*) to determine the I/O communication performance of the processors. Adjust X, Y, or Z block allocation accordingly.

Problem: BTOS boards boot and function properly yet the STATUS display stops at 15 while booting CENTIX.

Possible causes.

- There is no run statement for the CENTIX File System Server ([sys]<sys>UFS.run) in any of the processor initialization files.
- The dollar sign ($) directory is corrupted.

- The CENTIX file system configuration file, [sys]<sys>ConfigUFS.sys, does not match the actual system configuration.

Recommended action. Boot the system in restricted mode or from a bootable medium and check that either an FP or DP initialization file contains the run statement for the CENTIX File System Server. Also check that the partition assignments in [sys]<sys>ConfigUFS.sys match the actual system configuration.

To check whether the dollar sign directory files are corrupted, use the recovery procedure under "Determining Corrupted System Files" later in this section.

Common BTOS User and System Errors

The most common errors that result from user mistakes are file, directory, and volume management errors:

1 Invalid file specifications (for example, if a user provides an invalid file name, or requests an operation on a file that does not exist)

2 Security violations (for example, if a user attempts to modify a read-only file or enter an invalid password)

3 Overflow conditions (for example, a directory or a volume fills up)

To recover from this type of error, you need only correct the entry that caused the error.

Other file errors may result from disk malfunctions that have resulted in the loss of volume control structures. You can recover from damage to volume control structures caused by disk malfunctions by backing up, reinitializing, and restoring each volume from archive files.

Certain device errors resulting from user oversights are easily recoverable. Such errors include improper initialization of the device, unit power being off, or the medium for a device not being on line (for example, a tape is not properly loaded into the tape drive).

Printer spooler errors include configuration errors (such as attempting to add a printer to a channel that is not free) and incorrect security procedures (for example, specifying a password when one is not required).

Intermittent System Crashes

The master processor monitors all processors within the XE 500 system by sampling internal locations (ordinarily active) in other processors. If a processor continues to be inactive, the error is logged, and the whole system is brought down to prevent damage. The front panel STATUS display at the XE 500 base enclosure indicates the type of failing processor.

If the problem is due to software, rebooting will normally allow you to return the system to operational status. If the problem is a hardware malfunction of one of the processor boards, you can remove that processor's entry in the master configuration file and reboot the system in order to run without the failed processor. For information about the master configuration file, see the *XE 500 CENTIX Installation and Implementation Guide*.

Intermittent crashing of the system is a wide-open subject. Causes can range from bad input power to a mix of software revisions running concurrently. Analyzing system log entries and crash dump files associated with each crash are starting points to determining the source of the problem. The following is a list of typical XE 500 crashes, accompanied by a possible cause and recommended action.

Problem: ERC 22 (bus time-out), ERC 28 (Bus error).

Possible causes.

- Poor seating of processor board with the backplane.
- Hardware not distributed across the system bus properly.
- The crashing processor or the processor involved in a remote reference is faulty.
- An attempt was made to access a nonexistent processor.

Recommended action. XE 500 processors can assume two roles, that of a bus master or a bus slave. A processor is a bus master when it writes and reads another processor's memory, a bus slave when its memory is written and read by the bus master. A bus time-out (ERC 22) will occur on a bus master if an acknowledge to a request made to a bus slave is not received in 15 microseconds.

A slave processor experiencing a crash such as an ERC 21, ERC 23, and so on, will issue the bus error signal to its master. The master will then crash with an ERC 28.

Refer to the system log, using the **MPLog** command, to identify the two boards involved. The board on which the ERC 22 or ERC 28 occurred is always the master; the slave is identified by the slot number in system crash status word 3 in the error report associated with the crash.

Note that AP boards do not return crash codes in BTOS form. An AP will instead return to NMI panic. This should be handled in the same way as the FP, CP, TP, DP, and SP if the cause of the failure is believed to be hardware. AP error description must be interpreted using the values held in its general status register (GSR).

Keep the bus length short between two processors that frequently assume a master/slave relationship. Here are some examples:

- An AP and associated I/O processors (TPs and CPs).
- An AP and the processor that controls the disk on which the AP's swap area resides.
- A CP and the processor that controls the disk for which most of the disk I/O requests from the terminals connected to the CP are directed.

Problem: System freezes (no error logging).

Possible causes. A power supply of less than the minimum revision is being used. These supplies are at times unable to source the high current demand required by a fully populated enclosure.

Recommended action. Contact your Burroughs field service representative.

RS-232-C Serial Communications Problems

Problem: Corrupted or no communications between a CP or TP and an RS-232-C serial device.

Possible causes.

- There is no entry for the RS-232-C channel in the processor's configuration file or the entry is not properly defined.
- The RS-232-C port on the processor is faulty.
- The cable is faulty or the wrong cable is being used.

Recommended action. Check that the entry for the channel in the processor's configuration file exists and is correct.

Cluster Communications Problems

Problem: Corrupted or no communications between a CP and a cluster terminal. Error codes 8100 (time-outs) or 8103 (protocol failure) are reported. PT 1500s cannot run at the high baud rate of 1.8 M.

Possible causes.

- The cluster lines are not properly terminated.
- The RS-422 port on the processor is faulty.
- The cable is faulty.
- There is no entry for PT 1500s in the CP's configuration file or the entry does not match the number of PT 1500s connected to the processor.

Recommended action. The total length from the beginning of an RS-422 channel on the CP board to the end of the cluster line may be up to 1200 feet at baud rates of either 307 k or 1.8 M. This 1200-foot maximum length is a sum total of all cable lengths between cluster terminals and consequently between cluster line terminators.

Make sure that both lines associated with a cluster port are properly terminated using an RS-422 100-ohm terminator.

A cluster port can be configured (in the CP's configuration file) to run at 1.8 M baud only if a PT 1500 is **not** in the cluster line. Otherwise, the line must be configured to run at 307 k baud.

Check that all RS-422 cables are properly connected and not defective.

Check that the entry for PT 1500s in the CP's configuration file exists and is correct.

Disk Drive Problems

Problem: Error 301 (Disk I/O error) is reported.

Possible causes.

- Previously unrecorded bad spots are appearing on the disk.
- Deteriorating disk medium producing unwriteable or unreadable areas of the disk.
- AC and DC grounds are connected.
- Unstable or inadequate input power.
- Disk drive daisy chain not terminated properly (for example, there is no terminator or there is more than one terminator in the chain).

Recommended action. Run the BTOS MDisk Verify utility to check the integrity of the disk's volume control structures and identify new bad spots. Perform a backup, reinitialization, and restore of the disk.

If ERC 301 errors continue being reported or the MDisk Verify utility does not indicate any problems, contact your Burroughs field service representative.

Problem: Disk drive cannot be accessed by BTOS.

Possible causes.

- The configuration file for the FP or DP is missing, incorrect, or not included in the master configuration file.
- The disk drive configuration file for the disk type does not exist or is incorrect.

- The FP or DP is faulty.
- Disk drive cables are faulty.
- The disk drive's power supply is malfunctioning.
- For an SMD in an MD3 enclosure, the chassis of the SMD drive is not grounded to the XE 500.

Recommended action. Make sure that the configuration file for the FP or DP exists, is correct, and is included in the master configuration file for all FPs and DPs except the master FP or DP. Also make sure that the disk drive configuration files exist and are correct. For more information about these configuration files, refer to the *XE 500 CENTIX Installation and Implementation Guide*.

Problem: A disk cartridge that was previously readable or bootable cannot be read.

Possible causes.

- The disk cartridge is faulty.
- The disk cartridge was initialized using a drive with a different revision level than the drive being used to read the cartridge.
- For a bootable cartridge, the cartridge has run out of space for dollar sign directory files.
- For a bootable cartridge, the cartridge was not initialized to accommodate the additional number of dollar sign directories created during a large system boot.

Recommended action. Insure proper disk cartridge drive operation by trying to read another cartridge.

While booted from the system disk, use the **MVolume Status** command to see how many $ directories currently exist on the cartridge. If there are more than ten, delete five and attempt a reboot.

If the cartridge cannot be read either by attempting to boot or by copying the contents of the cartridge to [nul], as described under "Determining Corrupted System Files" later in this section, the cartridge is either bad (from being dropped) or was initialized and written using a drive of a lower revision level. In the latter case, use a drive able to read the cartridge to backup the data to a built-in disk, if this is possible. Reinitialize and restore the disk cartridge using a drive of the current revision level.

Recovering from System Problems

The following subsections outline procedures for recovering from some system problems.

If corrupted system files have caused a system failure, whether due to a faulty system disk or to missing or incorrect system files, the recovery approach is to boot the system from another set of system files. This set can either be restricted mode system files, if the system disk is operating properly, or a set on a bootable disk cartridge, if your system has a disk cartridge drive.

If system files become corrupted, certain system services may fail or the system may be unable to boot up in normal mode at all. For example, if a master BTOS command's run file has become corrupted, the command will fail when you attempt to execute it. Or, if the master operating system, [sys]<sys>SysImage.sys, has become corrupted, the system may be unable to boot, even in the restricted mode.

If only certain system services fail and you suspect that corresponding system files have become corrupted, you should

- Boot up in restricted mode.
- If you know which files are corrupted, use the MRestore utility to restore those files from your system software archive media.

  If you are not sure which files are corrupted, run the MSysLoad or CentixLoad utility to reinstall the software package that contains the files you suspect are corrupted.

If the system fails to boot in either the restricted or the normal mode, you should reinstall restricted mode software. Then see if the corrupted system files can be corrected, or use the MRestore utility to restore system files.

If the system fails to boot again, the system disk probably has a hardware failure. Either

- Reinstall restricted mode software to a disk other than the faulty system disk.
- Boot up the system from a bootable disk cartridge, if possible.

Using the Restricted Mode

If the system disk is not faulty or bad spots have only corrupted normal system files, you should be able to bring the system up in restricted mode. The capabilities of restricted mode are discussed in Section 15.

Using a Bootable Disk Cartridge

If your system's base enclosure has a disk cartridge drive, you can boot the system from a bootable disk cartridge.

With the system booted from a bootable disk cartridge, you can examine the system files for the cause of the system problem.

Determining Corrupted System Files

To determine which system files are corrupted, use the following procedure:

1. Boot the system in either restricted mode or from a bootable removable medium.
2. Use an editor to review the system configuration files on the system disk (that is, the master configuration file, processor initialization files, processor configuration files, and so on).

If the system configuration files are present and correct, go to step 3.

3 Use the **MCopy** command to copy the entire system disk to "nul."

Enter "[*sysdisk*]<*>*" in the "File from" field, where *sysdisk* is the device or volume name of the system disk.

Enter "[nul]<*>*" in the "File to" field.

4 If ERC 301's (disk I/O errors) are encountered during the copy operation, take note of the files for which the ERC 301 was reported. Then check the system log using the **MPLog** command for the location of the unreadable area (bad spot) of the disk in cylinder, head, and sector format.

5 Backup the system disk using the **MBackup Volume** or **MTape Backup Volume** command.

6 Reinitialize the system disk, entering new bad spots found during the MCopy operation.

7 Restore the files to the system disk using the **MRestore** or **MTape Restore** command.

Restore the corrupted files found in step 4 from your system software archive media, or reinstall the files from your software release media.

Running the System in a Degraded Mode

If a processor board hardware failure is suspected to be the cause of system problems, you can run the system in a degraded mode.

If the failed processor is not essential for normal system operation, remove the processor's entry from the master configuration file, [sys]<sys>Master.cnf, and reboot the system. This allows you to run the system until the failed processor can be replaced.

fsck Messages

The error conditions are organized by the phase of the fsck program in which they can occur. The error conditions that can occur in more than one phase is discussed in the first subsection, "Initialization."

Initialization

Before a file system check can be performed, certain tables have to be set up and certain files opened. This section describes the opening of files and the initialization of tables. Error conditions resulting from command line options, memory requests opening of files, status of files, file system size checks, and creation of the scratch file are listed below.

C option?

C is not a legal option to fsck; legal options are **-y, -n, -s, -S, -t, -r, -q,** and **-D.** fsck terminates on this error condition. See your *CENTIX Operations Reference Manual* for details.

Bad -t option

The -t option is not followed by a file name. fsck terminates on this error condition. See your *CENTIX Operations Reference Manual* for details.

Invalid -s argument, defaults assumed

The **-s** option is not suffixed by 3, 4, or blocks-per-cylinder: blocks-to-skip. fsck assumes a default value of 400 blocks-per-cylinder and 9 blocks-to-skip. See your *CENTIX Operations Reference Manual* for details.

Incompatible options: -n and -s

It is not possible to salvage the free-block list without modifying the file system. fsck terminates on this error condition. See your *CENTIX Operations Reference Manual* for details.

Cannot fstat standard input

The fsck program's attempt to fstat standard input failed. The occurrence of this error condition indicates a serious problem which may require additional assistance. fsck terminates on this error condition.

Cannot get memory

The fsck program's request for memory for its virtual memory tables failed. The occurrence of this error condition indicates a serious problem which may require additional assistance. fsck terminates on this error condition.

Cannot open checklist file: F

The default file system checklist file *F* (usually */etc/checklist*) cannot be opened for reading. fsck terminates on this error condition. Check access modes of *F*.

Cannot stat root

The fsck programs's request for statistics about the root directory (/) failed. The occurrence of this error condition indicates a serious problem which may require additional assistance. fsck terminates on this error condition.

Cannot stat F

The fsck program's request for statistics about the file system *F* failed. It ignores this file system and continues checking the next file system given. Check access modes of *F*.

F is not a block or character device

fsck has been given a regular file name by mistake. It ignores this file system and continues checking the next file system given. Check file type of *F*.

Cannot open F

The file system *F* cannot be opened for reading. It ignores this file system and continues checking the next file system given. Check access modes of *F*.

Size check: fsize X isize Y

More blocks are used for the inode list *Y* than there are blocks in the file system *X*, or there are more than 65,535 inodes in the file system. It ignores this file system and continues checking the next file system given.

Cannot create F

The fsck program's request to create a scratch file *F* failed. It ignores this file system and continues checking the next file system given. Check access modes of *F*.

CANNOT SEEK: BLK B (CONTINUE)

The fsck program's request for moving to a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem which may require additional assistance.

Possible responses to CONTINUE prompt are:

YES	Attempt to continue to run file system check. Often, however, the problem persists. This error condition does not allow a complete check of the file system. A second run of fsck should be made to recheck this file system. If block was part of the virtual memory buffer cache, fsck terminates with the message "Fatal I/O error".
NO	Terminate program.

Phase 1: Check Blocks and Sizes

This phase checks the inode list. This part lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format.

UNKNOWN FILE TYPE I = I (CLEAR)

The mode word of the inode *I* indicates that the inode is not a special character inode, regular inode, or directory inode.

Possible responses to CLEAR prompt are:

YES	Deallocate inode *I* by zeroing its contents. This always invokes the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode.
NO	Ignore this error condition.

LINK COUNT TABLE OVERFLOW (CONTINUE)

An internal table for fsck containing allocated inodes with a link count of zero has no more room. Recompile fsck with a larger value of MAXLNCNT.

Possible responses to CONTINUE prompt are:

YES	Continue with program. This error condition does not allow a complete check of the file system. A second run of fsck should be made to recheck this file system. If another allocated inode with a zero link count is found, this error condition is repeated.
NO	Terminate program.

B BAD I = I

Inode *I* contains block number *B* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the EXCESSIVE BAD BLKS error condition in Phase 1 if inode *I* has too many block numbers outside the file system range. This error condition always invokes the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE BAD BLKS I = I (CONTINUE)

There is more than a tolerable number (usually 10) of blocks with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system associated with inode *I*.

Possible responses to CONTINUE prompt are:

YES	Ignore the rest of the blocks in this inode and continue checking with next inode in the file system. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system.
NO	Terminate program.

B DUP I = I

Inode *I* contains block number *B* which is already claimed by another inode. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if inode *I* has too many block numbers claimed by other inodes. This error condition always invokes Phase 1b and the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE DUP BLKS I = I (CONTINUE)

There is more than a tolerable number (usually 10) of blocks claimed by other inodes.

Possible responses to CONTINUE prompt are:

YES	Ignore the rest of the blocks in this inode and continue checking with next inode in the file system. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system.
NO	Terminate program.

DUP TABLE OVERFLOW (CONTINUE)

An internal table in fsck containing duplicate block numbers has no more room. Recompile fsck with a larger value of DUPTBLSIZE.

Possible responses to CONTINUE prompt are:

YES	Continue with program. This error condition does not allow a complete check of the file system. A second run of fsck should be made to recheck this file system. If another duplicate block is found, this error condition is repeated.
NO	Terminate program.

POSSIBLE FILE SIZE ERROR I = I

The inode *I* size does not match the actual number of blocks used by the inode. This is only a warning. If the **-q** option is used, this message is not printed.

DIRECTORY MISALIGNED I = I

This size of a directory inode is not a multiple of the size of a directory entry (usually 16). This is only a warning. If the **-q** option is used, this message is not printed.

PARTIALLY ALLOCATED INODE I = I (CLEAR)

Inode *I* is neither allocated nor unallocated.

Possible responses to CLEAR prompt are:

YES	Deallocate inode *I* by zeroing its contents.
NO	Ignore this error condition.

Phase 1B: Rescan for More Dupes

When a duplicate block is found in the file system, the file system is rescanned to find the inode which previously claimed that block. This part lists the error condition when the duplicate block is found.

B DUP I = I

Inode *I* contains block number *B*, which is already claimed by another inode. This error condition always invokes the BAD/DUP error condition in Phase 2. Inodes with overlapping blocks may be determined by examining this error condition and the DUP error condition in Phase 1.

Phase 2: Check Pathnames

In this phase, directory entries that point to error conditioned inodes from Phase 1 and Phase 1b are removed. This part lists error conditions resulting from root inode mode and status, directory inode pointers in range, and directory entries pointing to bad inodes.

ROOT INODE UNALLOCATED. TERMINATING

The root inode (always inode number 2) has no allocated mode bits. The occurrence of this error condition indicates a serious problem which may require additional assistance. The program terminates.

ROOT INODE NOT DIRECTORY (FIX)

The root inode (usually inode number 2) is not directory inode type.

Possible responses to FIX prompt are:

YES	Make the root inode's type to be a directory. If the root inode's data blocks are not directory blocks, a very large number of error conditions are produced.
NO	Terminate program.

DUPS/BAD IN ROOT INODE (CONTINUE)

Phase 1 or phase 1b have found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system.

Possible responses to CONTINUE prompt are:

YES	Ignore DUPS/BAD error condition in root inode and attempt to continue to run the file system check. If root inode is not correct, a large number of other error conditions can occur.
NO	Terminate program.

I OUT OF RANGE I=I NAME=F (REMOVE)

A directory entry *F* has an inode number *I* which is greater than the end of the inode list.

Possible responses to REMOVE prompt are:

YES	The directory entry *F* is removed.
NO	Ignore this error condition.

UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F (REMOVE)

A directory entry *F* has an inode *I* without allocate mode bits. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed. If the file system is not mounted and the **-n** option was not specified, the entry is removed automatically if the inode it points to is character size *O*.

Possible responses to REMOVE prompt are:

YES	The directory entry *F* is removed.
NO	Ignore this error condition.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory entry *F*, directory inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to REMOVE prompt are:

YES	The directory entry *F* is removed.
NO	Ignore this error condition.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T FILE=F (REMOVE)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed.

Possible responses to REMOVE prompt are:

YES	The directory entry *F* is removed.
NO	Ignore this error condition.

BAD BLK B IN DIR I=I OWNER=O MODE=M SIZE=S MTIME=T

This message occurs only when the **-q** option is used. A bad block was found in DIR inode *I*. Error conditions looked for in directory blocks are nonzero padded entries, inconsistent "." and ".." entries, and imbedded slashes in the name field. This error message indicates that the user should later either remove the directory inode if the entire block looks bad or change (or remove) those directory entries that look bad.

Phase 3: Check Connectivity

This phase checks the directory connectivity seen in Phase 2. This part lists error conditions resulting from unreferenced directories and missing or full lost+found directories.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)

The directory inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. fsck forces the reconnection of a nonempty directory.

Possible responses to RECONNECT prompt are:

YES	Reconnect directory inode *I* to the file system in directory for lost files (usually lost+found). This may invoke lost+found error condition in Phase 3 if there are problems connecting directory inode *I* to lost+found. This may also invoke CONNECTED error condition in Phase 3 if link was successful.
NO	Ignore this error condition. This always invokes UNREF error condition in Phase 4.

SORRY. NO lost+found DIRECTORY

There is no lost+found directory in the root directory of the file system; fsck ignores the request to link a directory in lost+found. This always invokes the UNREF error condition in Phase 4. Check access modes of lost+found. See your *CENTIX Operations Reference Manual* for details.

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the lost+found directory in the root directory of the file system; **fsck** ignores the request to link a directory in lost+found. This always invokes the UNREF error condition in Phase 4. Clean out unnecessary entries in lost+found or make lost+found larger. See your *CENTIX Operations Reference Manual* for details.

DIR I=I1 CONNECTED. PARENT WAS I=I2

This is an advisory message indicating a directory inode *I1* was successfully connected to the lost+found directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the lost+found directory.

Phase 4: Check Reference Counts

This phase checks the link count information seen in Phase 2 and Phase 3. This part lists error conditions resulting from unreferenced files; missing or full lost+found directory; incorrect link counts for files, directories, or special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free-inode counts.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the **-n** option is not set and the file system is not mounted, empty files are not reconnected and are cleared automatically.

Possible responses to RECONNECT prompt are:

YES	Reconnect inode *I* to file system in the directory for lost files (usually lost+found). This may invoke lost+found error condition in Phase 4 if there are problems connecting inode *I* to lost+found.
NO	Ignore this error condition. This always invokes CLEAR error condition in Phase 4.

SORRY. NO lost+found DIRECTORY

There is no lost+found directory in the root directory of the file system; **fsck** ignores the request to link a file in lost+found. This always invokes CLEAR error condition in Phase 4. Check access modes of lost+found.

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the lost+found directory in the root directory of the file system; fsck ignores the request to link a file in lost+found. This always invokes the CLEAR error condition in Phase 4. Check size and contents of lost+found.

(CLEAR)

The inode mentioned in the immediately previous error condition cannot be reconnected.

Possible responses to CLEAR prompt are:

YES	Deallocate inode mentioned in the immediately previous error condition by zeroing its content.
NO	Ignore this error condition.

LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for inode *I*, which is a file, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed.

Possible responses to ADJUST prompt are:

YES	Replace link count of file inode *I* with *Y*.
NO	Ignore this error condition.

LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for inode *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed.

Possible responses to ADJUST prompt are:

YES	Replace link count of directory inode *I* with *Y*.
NO	Ignore this error condition.

LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for *F* inode *I* is *X* but should be *Y*. The file name *F*, owner *O*, mode *M*, size *S*, and modify time *T* are printed.

Possible responses to ADJUST prompt are:

YES	Replaces link count of inode *I* with *Y*.
NO	Ignore this error condition.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Inode *I*, which is a file, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the **-n** option is not set and the file system is not mounted, empty files are cleared automatically.

Possible responses to CLEAR prompt are:

YES	Deallocate inode *I* by zeroing its contents.
NO	Ignore this error condition.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Inode *I*, which is a directory, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the **-n** option is not set and the file system is not mounted, empty directories are cleared automatically. Non-empty directories are not cleared.

Possible responses to CLEAR prompt are:

YES	Deallocate inode *I* by zeroing its contents.
NO	Ignore this error condition.

BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with file inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed.

Possible responses to CLEAR prompt are:

YES	Deallocate inode *I* by zeroing its contents.
NO	Ignore this error condition.

BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed.

Possible responses to CLEAR prompt are:

YES	Deallocate inode *I* by zeroing its content.
NO	Ignore this error condition.

FREE INODE COUNT WRONG IN SUPERBLK (FIX)

The actual count of the free inodes does not match the count in the superblock of the file system. If the **-q** option is specified, the count are fixed automatically in the superblock.

Possible responses to FIX prompt are:

YES	Replace count in superblock by actual count.
NO	Ignore this error condition.

Phase 5: Check Free List (Non-Pilf File Systems)

This phase concerns itself with the free-block list. This part lists error conditions resulting from bad blocks in the free-block list, bad free-blocks count, duplicate blocks in the free-block list, unused blocks from the file system not in the free-block list, and the total free-block count incorrect.

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE)

The free-block list contains more than a tolerable number (usually 10) of blocks with a value less than the first data block in the file system or greater than the last block in the file system.

Possible responses to CONTINUE prompt are:

YES	Ignore rest of the free-block list and continue execution of **fsck**. This error condition always invokes "BAD BLKS IN FREE LIST" error condition in Phase 5.
NO	Terminate program.

EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE)

The free-block list contains more than a tolerable number (usually 10) of blocks claimed by inodes or earlier parts of the free-block list.

Possible responses to CONTINUE prompt are:

YES	Ignore the rest of the free-block list and continue execution of fsck. This error condition always invokes "DUP BLKS IN FREE LIST" error condition in Phase 5.
NO	Terminate program.

BAD FREEBLK COUNT

The count of free blocks in a free-list block is greater than 50 or less than 0. This error condition always invokes the "BAD FREE LIST" condition in Phase 5.

X BAD BLKS IN FREE LIST

X blocks in the free-block list have a block number lower than the first data block in the file system or greater than the last block in the file system. This error condition always invokes the "BAD FREE LIST" condition in Phase 5.

X DUP BLKS IN FREE LIST

X blocks claimed by inodes or earlier parts of the free-list block were found in the free-block list. This error condition always invokes the "BAD FREE LIST" condition in Phase 5.

X BLK(S) MISSING

X blocks unused by the file system were not found in the free-block list. This error condition always invokes the "BAD FREE LIST" condition in Phase 5.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)

The actual count of free blocks does not match the count in the superblock of the file system.

Possible responses to FIX prompt are:

YES	Replace count in superblock by actual count.
NO	Ignore this error condition.

BAD FREE LIST (SALVAGE)

Phase 5 has found bad blocks in the free-block list, duplicate blocks in the free-block list, or blocks missing from the file system. If the **-q** option is specified, the free-block list is salvaged automatically.

Possible responses to SALVAGE prompt are:

YES	Replace actual free-block list with a new free-block list. The new free-block list is ordered to reduce time spent by the disk waiting for the disk to rotate into position.
NO	Ignore this error condition.

Phase 5: Check Bit Map (Pilf File Systems)

The bit map is checked in this phase. Error conditions that result from bad or duplicate blocks in the bit map are listed.

X BAD BLOCKS IN BIT MAP

X number of blocks in the bit map have a block number lower than the first data block in the file system or greater than the last block in the file system. This error condition always invokes the "BAD BIT MAP" condition in phase 5.

X DUP BLKS IN BIT MAP

X number of blocks claimed by inodes or earlier parts of the bit map were found in the bit map. This error condition always invokes the "BAD BIT MAP" condition in phase 5.

BAD BIT MAP (SALVAGE)

Bad blocks or duplicate blocks have been found in the bit map. If the -q option is specified, the bit map is salvaged automatically.

Possible responses to SALVAGE prompt are:

YES	Replace the bit map with an initialized bit map.
NO	Ignore this error condition

EXCESSIVE DUP BLKS IN BIT MAP (CONTINUE)

The bit map contains more than a tolerable number (usually 10) of blocks claimed by inodes or earlier parts of the bit map.

Possible responses to CONTINUE prompt are:

YES	Ignore the rest of the bit map and continue execution of fsck. This error condition always invokes "DUP BLKS IN BIT MAP" error condition in phase 5.
NO	Terminate program.

Phase 6: Salvage Free List (Non-Pilf File Systems)

This phase checks the free-block list reconstruction. This part lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

Default free-block list spacing assumed

This is an advisory message indicating the blocks-to-skip is greater than the blocks-per-cylinder, the blocks-to-skip is less than 1, the blocks-per-cylinder is less than 1, or the blocks-per-clinder is greater than 500. The default values of 9 blocks-to-skip and 400 blocks-per cylinder are used. See your *CENTIX Operations Reference Manual* for details.

Phase 6: Salvage Bit Map (Pilf File Systems)

I inodes had their cluster factor reset to 0

When a file system is created, a cluster size is specified. If an inode has been corrupted, fsck resets the cluster factor for that inode to 0.

Cleanup

Once a file system has been checked, a few cleanup functions are performed. This part lists advisory messages about the file system and modify status of the file system.

X files Y blocks Z free

This is an advisory message indicating that the file system checked contained *X* files using *Y* blocks leaving *Z* blocks free in the file system.

******* BOOT CENTIX (NO SYNC!) *******

This is an advisory message indicating that a mounted file system or the root file system has been modified by fsck. If the CENTIX system is not rebooted immediately without sync, the work done by fsck may be undone by the in-core copies of tables the CENTIX system keeps.

******* FILE SYSTEM WAS MODIFIED *******

This is an advisory message indicating that the current file system was modified by fsck.

Status Code Tables

Note: For other XE 500 status codes, see the BTOS Status Codes Reference Manual.

Table B-1 **XE 500-Specific BTOS Status Codes**

Decimal Value	Meaning
Kernel	
21	Memory reference fault. System crash status word 3 is the nonmaskable interrupt (NMI) status code. Status word 4 is the high 2 bits of the memory address at the time of the NMI.
22	Bus time-out. A bus time-out usually results from a reference to a nonexistent remote processor memory address. If system crash status word 4 is a 3, a remote reference was being performed when the crash occurred. Status word 5 is the contents of the remote slot number register (port hex 40). Word 6 is the contents of the base register zero (port hex 48).
23	Double-bit error detected by error correction code (ECC) circuitry. System crash status word 3 is the NMI status code. Status word 4 is the high 2 bits of the memory address at the time of the NMI. Status word 6 contains the ECC syndrome register contents.
24	Power failure. System crash status word 3 is the NMI status code. Status word 4 is the high 2 bits of the memory address at the time of the NMI.
25	Unknown NMI. System crash status word 3 is the NMI status code. Status word 4 is the high 2 bits of the memory address at the time of the NMI.
26	Stray interrupt. System crash status word 3 is the interrupt type multiplied by 6. Status word 4 is the interrupt service register, word 6 is the interrupt request register, and word 7 is the interrupt mask register.
27	Divide overflow. This is usually caused by a divide by zero. System crash status words 7 and 8 indicate the address of the instruction following the instruction being executed when the crash occurred.

Table B-1 **XE 500-Specific BTOS Status Codes (Cont.)**

Decimal Value	Meaning
28	Bus error. This error results from a reference to a nonexistent remote processor memory address or a remote double-bit error. If system crash status word 4 is a 3, a remote reference was being performed when the crash occurred. Status word 5 is the contents of the remote slot number register (port hex 40). Word 6 is the contents of the base register zero (port hex 48).
39	Cannot resign as a server of a request you are not serving or attempt to serve a request that is already being served.
Initialization	
104	Insufficient Y blocks or Z blocks specified at system generation. Increase the number of Y blocks or Z blocks of the failing processor by either recustomizing its operating system or modifying the Y or Z block allocation entry in the processor's configuration file. Refer to the *XE 500 BTOS Customizer Operations Guide* or the *XE 500 CENTIX Installation and Implementation Guide* for more information about Y and Z blocks.
105	Insufficient hardware configuration detected. There are no CPs running or there are too many FPs, DPs, or SPs running.
106	A processor initialization file contains an error. Correct the processor initialization file for the failing processor. The *XE 500 CENTIX Installation and Implementation Guide* contains information about the proper format for processor initialization files.
107	Watchdog process on master processor detected failure of another processor on the bus. This error will only be returned if a processor other than the master processor crashes and the "NoWatchDog" entry has been removed from the XE 500 master configuration file, [sys]<sys>Master.cnf.
150	A processor attempted to route a request to itself, which cannot be done.

Table B-1 **XE 500-Specific BTOS Status Codes (Cont.)**

Decimal Value	Meaning
151	Remote request too large for local buffer. Try increasing the number of Y or Z blocks for the failing processor. Refer to the *XE 500 BTOS Customizer Operations Guide* or the *XE 500 CENTIX Installation and Implementation Guide* for more information about Y and Z blocks.
152	Cannot remote boot a processor that has not been initialized.
153	Target processor did not respond to remote boot.
154	Illegal option used in GetSlotInfo or GetProcInfo or illegal code in the Cdt issued from RemoteBoot.
155	Too many disk devices declared in the [sys]<sys>Master.cnf file. Increase the size of %nmounted disks in the system generation prefix file table for the master processor and regenerate the master processor's operating system. For more information about customizing XE 500 processor operating systems, refer to the *XE 500 BTOS Customizer Operations Guide*.
Device Management	
305	Attempt to reference unformatted disk. Any attempt to use an unformatted disk by any I/O request, except by the format request, returns this error.
306	Recall failed.
307	Hard disk controller write fault detected.
350	Hard disk controller local direct memory access (DMA) fault.
352	Hard disk controller remote DMA fault.
Allocation (BTOS)	
403	Parameter in a BTOS memory management request is not aligned on a paragraph boundary (that is, the offset part of the segment:offset memory address must be zero).

Table B-1 **XE 500-Specific BTOS Status Codes (Cont.)**

Decimal Value	Meaning
Printer Spooler	
709	Current print request cancelled.
710	Printer restarted.
711	Printer is freed and needs to be reconfigured.
720	Too many printers specified in spooler configuration file. The limit is 100.
721	Attempt to install a printer spooler manager on a processor that already has a printer spooler manager installed. Check that the failing processor's initialization file has only one printer spooler manager entry.
722	Attempt to install a printer spooler manager on a processor that does not have printer hardware. It can be installed only on a CP or TP. Check your FP, DP, or SP initialization files to make sure that they do not contain an entry for a printer spooler manager.
723	Attempt to install a printer spooler manager when there is no Queue Manager installed. (The spooler tries for two minutes, and then gives up.) Check your processor initialization files to make sure that they contain an entry for the Queue Manager.
Command Line Interpreter	
2752	The FINISH or CANCEL key has been pressed instead of the CODE-D or CODE-G keys, respectively, when attempting to execute a master utility from CENTIX through the **ofcli** command.
Tape Management (BTOS)	
9002	Tape operation timed out. This error is usually caused by an operator error or a hardware malfunction.
9009	End-of-tape encountered. For the QIC server, this is the logical end-of-tape (in write mode, the estimate of the remaining tape space has reached zero; in read mode, blank tape has been reached).

Table B-1 **XE 500-Specific BTOS Status Codes (Cont.)**

Decimal Value	Meaning
9010	Unrecoverable I/O error occurred during a read or write operation.
9016	Tape drive is not ready.
	Tape drive is either off-line or busy.
9021	A file mark was encountered during a read operation.
9024	A hardware error occurred in the tape drive or the controller.
9032	A tape service request was made with an invalid tape handle or user number.
9033	A tape service was made with invalid parameters.
9034	The half-inch tape server ran out of device control blocks.
9035	An open tape service was requested with an invalid tape name.
9036	An open tape service was requested on a drive currently being used by another user.
9037	An attempt was made to write on a write-protected tape or cartridge.
9038	An unrecognized command was issued as a tape operation request.
9039	A read or write request was issued to a half-inch tape drive that had tape errors outstanding.
9040	An open tape service was requested with an unrecognized open mode.
9041	A close tape service request was issued to a half-inch tape drive that had operations outstanding.
9050	During read mode, a tape was detected as being corrupted.
9051	During write mode with a QIC tape, the physical end-of-tape was reached before the logical end-of-tape; therefore, information was lost.
9052	Invalid tape position.
	A half-inch tape is neither at load point nor at the start of a file.
9053	Invalid tape reel.
	The tape is not part of the archive tape set currently being used for a restore operation.

Table B-1 **XE 500-Specific BTOS Status Codes (Cont.)**

Decimal Value	Meaning
9054	Invalid tape sequence.
	The tape is not the next archive tape in the current sequence being used for a restore operation.
9055	Invalid tape configuration file for the BTOS tape utility.
9056	Missing tape configuration file for the BTOS tape utility.
9057	Invalid tape record size specified for buffer in tape read/write request.
9058	Missing tape controller hardware.
9070	No internal buffers are available for remote processor copy-in/copy-out. This will happen only if the read/write request is from a remote processor.
	Reissue the request.
9071	Bad alignment of user-supplied data buffer.
	Be sure data buffers are aligned on word boundaries.
9072	The user request specified a data buffer size larger than that specified at tape server installation time.
9073	QIC tape was removed from the drive after being opened by a user.
9074	QIC server was given an illegal command.
9075	QIC tape is not a bootable tape.
9076	QIC tape is a corrupted bootable tape.
9080	The user of tape access methods supplied a buffer that was too small. The buffer must be at least three times the record size specified in the tape configuration file used by the access method.
9081	Too many tape servers installed in the system.
9099	An attempt was made to install a tape server on a processor that already had one.

Table B-1 **XE 500-Specific BTOS Status Codes (Cont.)**

Decimal Value	Meaning
BTOS System Requests	
11800	Internal waiting-on-Yblk error.
11801	Time-out waiting for Data Set Ready (DSR) after an open terminal request.
11802	The target processor does not respond. The processor has crashed or has a hardware failure.
11803	Routing area too small. When adding request codes, the size of the routing area in the master processor's CDT was not increased. Increase the value of the sRouteArea parameter in the master processor's Prefix.asm file to accommodate the additional requests. Refer to the *XE 500 BTOS Customizer Operations Guide*.
11804	Illegal parity option. A mark or space parity was specified with eight data bits; the hardware does not allow this combination.
11805	The user number table used to map global user numbers to local user numbers is too small. Increase the value of the %nRemoteUser parameter in the SysGen.mdf file. Refer to the *XE 500 BTOS Customizer Operations Guide*.
AdminAgent	
11900	The default path supplied to MfAdminAgent.run from WsAdminAgent.run is not a valid XE 500 path.
11901	The AdminAgent is busy and cannot service the request.
11902	The internal request block sent from the WsAdminAgent to the MfAdminAgent has an improper format. This error can occur if the WsAdminAgent and MfAdminAgent run files are not at the same version level.

Table B-2 **Front Panel STATUS Display Codes**

Code	Meaning
System Start-up/Initialization Sequence	
0	The system is in the reset mode (keyswitch is in the STOP position).
01	Starting boot procedure; performing boot ROM diagnostics.
02	Boot ROM writing memory to [sys]<sys>crashdump.sys.
03	Boot ROM performing full memory read/write.
04	ROM booting the master OS, [sys]<sys>SysImage.sys, on the master processor. Display alternates between 04 and A1 while waiting for system disk to become ready.
05	Boot ROM sequence successfully completed.
06	BTOS initialization begun.
07	Hardware initialization complete. First time master processor communicates across system bus.
08	Master processor initialization complete.
09	Master processor initializes disk hardware (reads [sys]<sys>Fp00.cnf or [sys]<sys>Dp00.cnf).
10	Master processor reads master configuration file, [sys]<sys>Master.cnf, and initializes other processors. Processor BTOS download is in progress.
CENTIX Initialization Sequence	
11	CENTIX memory test pass 1; CENTIX kernel debugger initialized.
12	CENTIX memory test pass 2.
13	CENTIX kernel trap vectors and initializes memory map.
14	CENTIX char, block, swap device initialized. Heap set up; ICC initialized.
15	CENTIX connections established to the CENTIX file system through [sys]<sys>ConfigUFS.sys and the CENTIX File System server, [sys]<sys>UFS.run.
16	CENTIX terminal output, semaphores, and clock initialized.
17	CENTIX starts swapper, init, and pager processes.

Table B-2 **Front Panel STATUS Display Codes (Cont.)**

Code	Meaning
18	CENTIX starts /etc/init.
19	/etc/init creates /etc/utmp, opens /etc/inittab files.
20	System is operating normally.
Boot Medium Error Codes	
21	The master processor ROM was unable to find a ready disk.
23	The master processor ROM found no valid home block on the first ready disk drive.
24	The master processor ROM found a bad run file signature on the first ready disk drive.
25	The run file checksum of the master OS, [sys]<sys>SysImage.sys, on the first ready disk is bad, indicating the file is corrupted.
Master DP Specific Medium Error Codes (These codes also appear on the rear panel.)	
26	Error when trying to reset.
27	Error when trying to rewind the tape.
28	Error when reading the tape.
29	Tape was written with incorrect or unsupported blocking factor.
Disk Hardware Error Codes (These codes also appear on the rear panel.)	
30	Disk error on drive 0.
31	Disk error on drive 1.
32	Disk error on drive 2.
33	Disk error on drive 3.
36	Bad disk controller; ROM could not read the controller register.
37	Bad disk controller; not ready in command phase.
38	Bad disk controller; no response.

Table B-2 **Front Panel STATUS Display Codes (Cont.)**

Code	Meaning
Processor Error Codes	
39	General board hardware error. A processor has reported a hardware error; see processor's rear panel LEDs for specific error.
40	Master processor detected FP time-out, indicating processor crash.
41	Master processor detected TP time-out, indicating processor crash.
42	Master processor detected CP time-out, indicating processor crash.
43	Master processor detected SP time-out, indicating processor crash.
44	Master processor detected DP time-out, indicating processor crash.
45	Master processor detected processor time-out, indicating processor crash. Processor type cannot be determined.
Software Error Codes	
50	Fatal software error detected. See processor rear panel LEDs for error code.
51	CENTIX file system has been corrupted. The fsck utility requires operator intervention.
Boot Load from Disk Cartridge Status Codes	
80	Start Boot Load utility and create BootLoad.log file.
81	Pause for MIVolume. User must turn the keyswitch to NORMAL to initialize the system disk.
82	Initializing the system disk.
86	Creating the directories listed in ProductContent.
87	Copying the system software from the release disk cartridge to the system disk.
88	Appending text to the appropriate file as listed in the ProductContent file.
89	Creating the files specified in ProductContent.
90	Successful completion of the software installation from the currently loaded release disk cartridge.
91	Cannot create the log file.
92	MIVolume has failed.
96	Failed to create a directory.

Table B-2 **Front Panel STATUS Display Codes (Cont.)**

Code	Meaning
97	Check the log file. An error has occurred. The following are examples: - A ProductContent file is missing. - Cannot create a specified file. - Cannot append BTOS commands to BootLoad.sub. - A copy has failed.
98	Cannot access SysContent.
99	Fatal error—keyswitch in MANUAL position.
Boot Load from Tape Status Codes	
80	Start Boot Load utility, create BootLoad.log file, and execute MChange Volume Name.
81	Pause for MIVolume. User must turn the keyswitch to NORMAL to initialize the system disk.
82	Initializing the system disk. If STATUS display shows 82 for more than 45 minutes, the release tape, tape drive, or system disk is probably faulty.
87	Restoring the system software from archive files on the release tape to the system disk. If the STATUS display shows 87 for more than 20 minutes, the release tape or tape drive is probably faulty.
90	Successful completion of the software installation from the currently loaded release tape.
Expansion Enclosure Error Codes	
DE	There is a break in the system bus between enclosures due to one of the following conditions: (1) An expansion enclosure is not turned on; (2) there is a faulty bus repeater board; (3) bus repeater cables are improperly installed or damaged.

Table B-3 **Processor Rear Panel LED Status Codes**

Number	Meaning
Codes Displayed While STATUS Display Is at 01	
01	Starting board hardware initialization.
02	Performing memory test of first 64 kB of memory.
03	Setting up initial memory area.
04	Determining memory size.
05	Relocating BTOS buffer. Setting up CDT.
06	Checking I/O ports.
07	Performing ECC "wash" to set up check bits.
10	Hardware initialization successfully completed.
Codes Displayed When STATUS Display Is at 39	
0A	Bad ROM checksum.
0B	Bad I/O port.
0C	Memory error in local memory while writing 0's.
0D	Memory error in local memory while writing 1's.
0E	Memory error in local memory while writing address pattern.
0F	Memory error in local memory while reading address pattern.
Bad Media Error Codes Displayed When STATUS Display Is at 21	
21	Cannot boot from any disk.
22	No tape or disk device contains a valid master OS, [sys]<sys>SysImage.sys.
23	No volume home block found on first ready drive.
24	Bad run file signature on first ready drive.
25	Bad run file checksum on first ready drive.

Table B-3 **Processor Rear Panel LED Status Codes (Cont.)**

Number	Meaning
Master DP Specific Medium Error Codes (These codes also appear on the front panel.)	
26	Error when trying to reset
27	Error when trying to rewind the tape.
28	Error when reading the tape.
29	Tape was written with incorrect or unsupported blocking factor.
Disk Hardware Error Codes (These codes also appear on the front panel.)	
30	Disk error on drive 0.
31	Disk error on drive 1.
32	Disk error on drive 2.
33	Disk error on drive 3.
Disk Hardware Error Codes Displayed When STATUS Display Is at 39	
36	Bad disk controller; cannot read controller registers.
37	Bad disk controller; never ready in command phase.
38	Bad disk controller; no response.
Master DP Specific Disk Hardware Error Codes Displayed When STATUS Display is at 39	
3A	Storage Processor (SP) of master DP does not have a Storage Controller (SC) attached.
3B	Inconsistent values on SMD controller interface.
3C	SMD Controller Data I/O (DIO) bit set incorrectly.
3D	A fatal SMD I/O error has occurred. (This is a generic error.)

Table B-3 **Processor Rear Panel LED Status Codes (Cont.)**

Number	Meaning
ME Board Error Codes Displayed When STATUS Display Is at 39	
4C	Memory error on ME board while reading 0's.
4D	Memory error on ME board while reading 1's.
4E	Memory error on ME board while writing address pattern.
4F	Memory error on ME board while reading address pattern.
General Status Codes	
1A	Error during hardware initialization.
1B	Error during data communications (data comm) peripheral initialization.
1C	Starting data comm dump or boot.
1D	Writing data comm dump.
1E	Reading data comm boot.
1F	Waiting for data comm line ACKnowledge.
81	Starting hardware initialization.
82	Starting checksum of ROM.
84	ECC logic and memory are initialized nondestructively (data is not lost).
FF	Processor is in reset state.

Table B-4 **System Crash Status Words**

Word	BTOS Error Codes	Meaning
1	All	The hexadecimal value for the corresponding BTOS decimal error code.
2	All	The process number that was running when the error occurred.
3	21, 22, 23, 24, 25, 26, 28	The 16-bit nonmaskable interrupt (NMI) status (read from the General Status Register, port hex 68; see Table B-5).

Table B-4 **System Crash Status Words (Cont.)**

Word	BTOS Error Codes	Meaning
4	21, 22, 23, 24, 25, 28	The high two bits of the memory address at the time of the NMI. If the value is 3, a remote memory reference was being performed when the crash occurred.
	26	The interrupt service register.
5	22, 28	The contents of the remote slot number register (port hex 40).
6	22	The contents of the base register zero (port hex 48). The base register zero holds bits 16 through 31 of the off-board memory address.
	23, 28	The error correction code (ECC) syndrome register contents.
	26	The interrupt request register.
7	All (except 26)	The code segment (CS) of the location following the call to the BTOS fatal error handler.
	26	The interrupt mask register.
8	All	The instruction pointer (IP) of the location following the call to the BTOS fatal error handler. Status words 7 and 8 indicate the memory location of the instruction immediately following the instruction that was being executed when the error occurred. The value in the CS register, shifted left one bit, plus the value in the IP register, gives the address.

Table B-5 **General Status Register Contents**

Bit	Name	Bit Setting	Meaning
15	ME	0	Memory Expansion board is installed.
14	SE	0	At least one processor in the system is driving the system error line on the system bus.
13	LR	0	Last reset by power up or RESET switch.
		1	Last reset by software.
12	NEM	0	If an NMI was caused by a reference to a nonexistent memory location, the reference could have been initiated locally, or by a processor or direct memory access (DMA) controller on another board.
11	PF	0	Power failure.
10	TO	0	Bus time-out. No acknowledge was received from a memory or I/O timing circuit.
9	BE	0	Bus error. A remote double-bit error or a reference to a nonexistent memory location occurred.
8	DBE	0	Double-bit error.
7	REM	1	The trapped address was generated by another processor.
6	CAD	1	The trapped address was generated by the local CPU.
5	DMA	1	The trapped address was generated by the local DMA controller.
3	DFP	1	This board is the master processor.
4	REM	1	Front panel keyswitch is in the REMOTE position. Only applicable for the master processor.
2	NOR	1	Front panel keyswitch is in the NORMAL position. Only applicable for the master processor.
1	A19		Bit 19 of the trapped address. Used with bit 0 (A18) to determine if trapped address is located in base memory, expansion memory, or on another processor.

Table B-5 **General Status Register Contents (Cont.)**

Bit	Name	Bit Setting	Meaning
0	A18		Bit 18 of the trapped address. If bits 19 and 18 are both 0, the trapped address is located in base memory, expansion memory, or on another processor. If bits 19 and 18 are both 1, the trapped address is located on another processor. Otherwise, the trapped address is located in expansion memory.

Table B-6 **Boot ROM Codes on an AP**

Code (hex)	Meaning
40	Running normally
80 or C0	Bootstrap ROM is initializing
83 or C3	Starting memory test
8C or CC	Memory error while writing zeros
8D or CD	Memory error while writing ones
8E or CE	Memory error while writing addresses
8F or CF	Memory error while verifying addresses
BF	System is in reset mode; the keyswitch is at STOP or the RESET button has been pushed

Table B-7 **Boot ROM Codes on an APII**

Code (hex)	Meaning
40	Running normally
21 or 61	Register testing and map initialization
22 or 62	Memory testing
23 or 63	Memory wash

Table B-7 **Boot ROM Codes on an APII (Cont.)**

Code (hex)	Meaning
Error Patterns:	
81 or C1	Error during memory test; wrote a zero, but read something else
82 or C2	Error during memory test; wrote FFs, but read something else
83 or C3	Error during memory test; wrote address, but read something else
84 or C4	Error during memory test; address verification
85 or C5	Error during context register
86 or C6	Error testing remote slot register
87 or C7	Error testing base register 0
88 or C8	Error testing base register 1
89 or C9	Error testing base register high
8A or CA	Error testing and initializing segment table
8B or CB	Error testing and initializing page table
8C or CC	Error testing and initializing GRC
BF	System is in reset mode; the keyswitch is at STOP or the RESET button has been pushed

Hardware Configuration Information

Figure C-1 **Built-in Disk Device Naming Conventions**

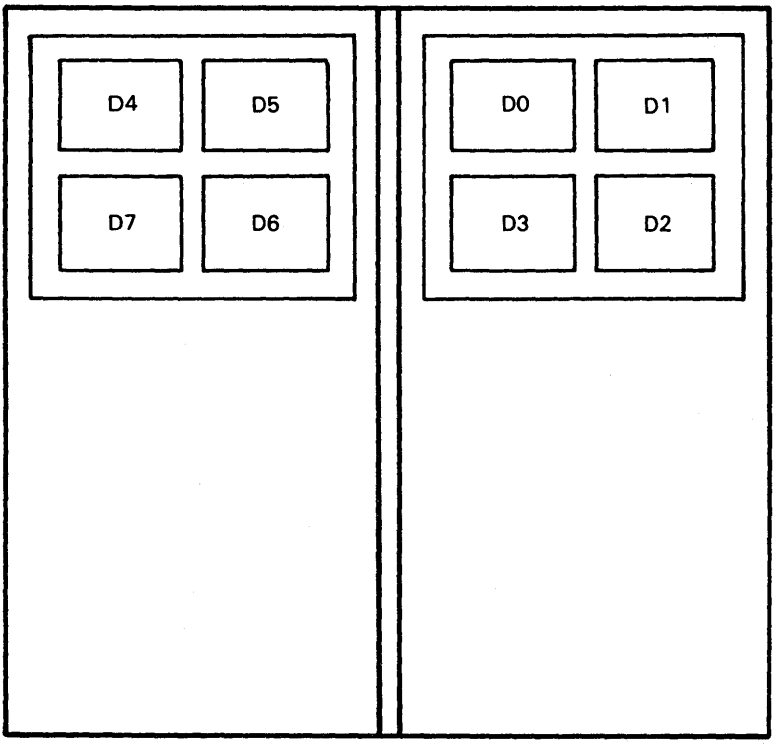
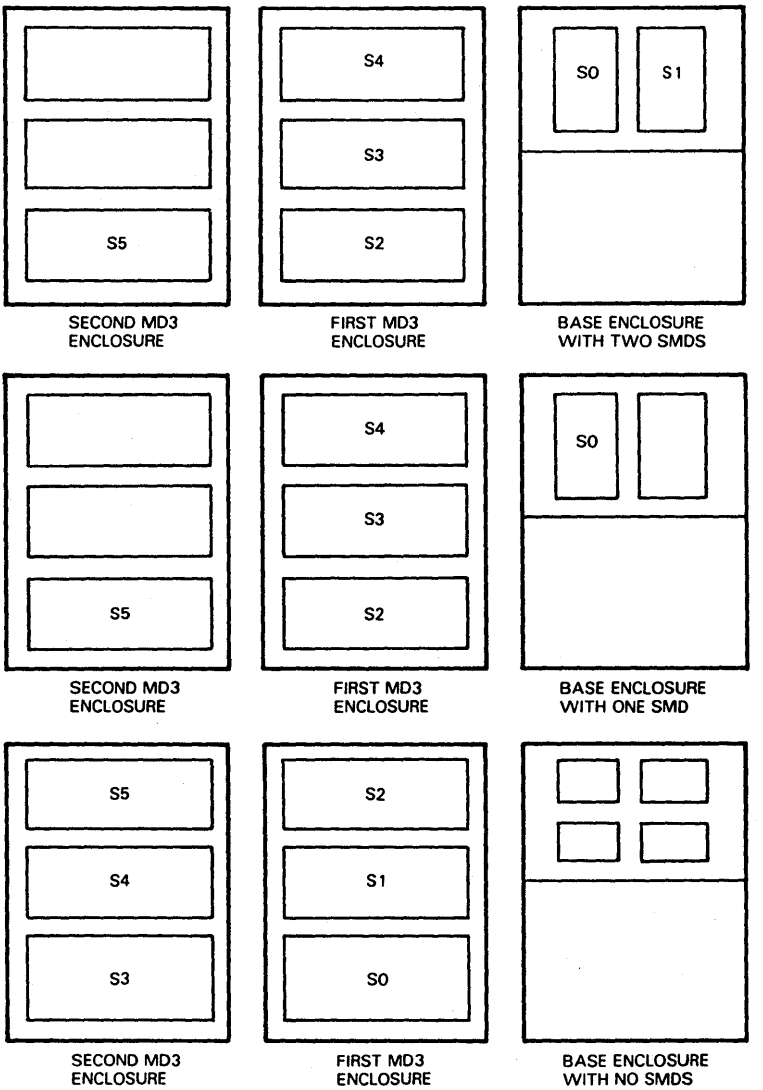


Figure C-2 **SMD Disk Device Naming Conventions for DP00**



ALL VIEWS ARE FROM THE REAR OF THE ENCLOSURES.

E7560

Figure C-3 **Processor Board Numbering Scheme**

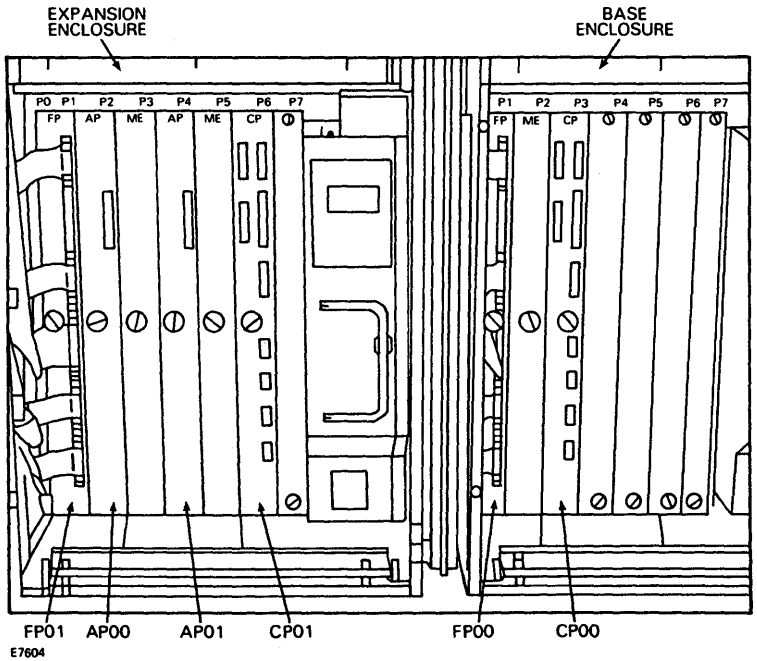


Table C-1 **Processor Slot Numbering Scheme**

Enclosure	P0	P1	P2	P3	P4	P5	P6	P7
Base	70	71	72	73	74	75	76	77
Expansion 1	60	61	62	63	64	65	66	67
Expansion 2	50	51	52	53	54	55	56	57
Expansion 3	40	41	42	43	44	45	46	47
Expansion 4	30	31	32	33	34	35	36	37
Expansion 5	20	21	22	23	24	25	26	27

Figure C-4 **Counting SP and DP Boards when identifying a Half-Inch Tape Driver**

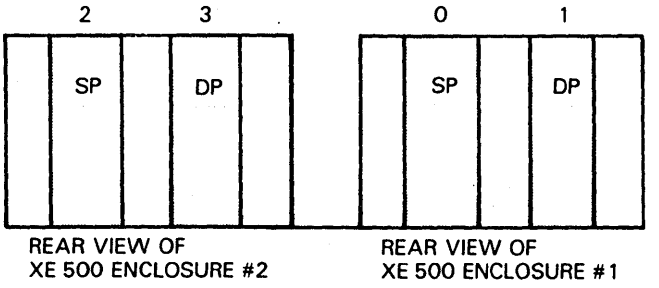
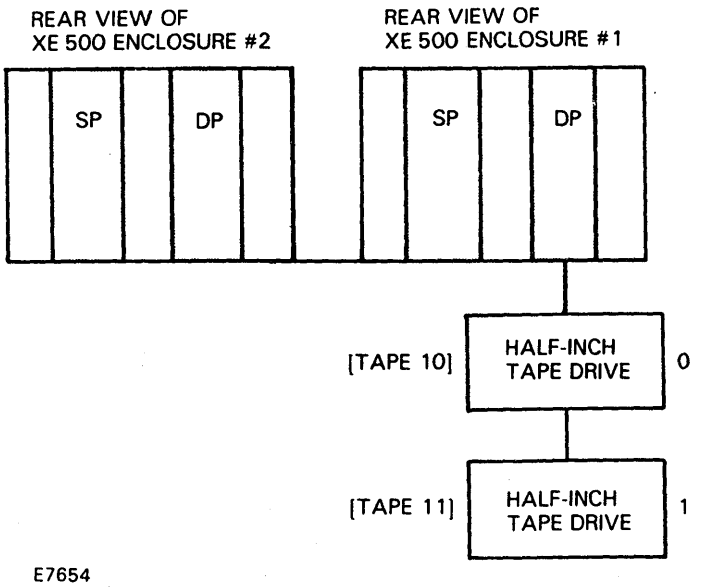


Figure C-5 **Identifying Half-Inch Tape Drives**



Reporting Problems to Burroughs

All software problems, software deficiencies, or suggestions for improvements should be reported by submitting a Field Communication Form (FCF), form number 3027057. The FCF should be reviewed for completeness and forwarded to the following address:

Burroughs Corporation
Product Assurance and Support
PO Box 235
Downingtown, PA 19335

Attn: FCF Coordinator

All information that would be helpful in understanding and analyzing the reported situation should be included with the FCF. This can be in the form of descriptions, computer listings, media containing programs and data, and so on. Please include a listing of the file [sys]<sys>Sys.Version with **all** FCFs submitted.

Certain problem situations are given below, together with appropriate information to be submitted for each.

For a hang or crash condition:

- A printout from an MPlog or the system log file itself ([sys]<sys>log.sys).
- Any observations of abnormal operation prior to the problem.
- Applicable crash files [for the master processor,[sys]<sys>crashdump.sys; for all other processors, [sys]<sys>*Xpnn*.crash, where *Xpnn* is the processor name (for example FP01, CP00, AP00)].

  If you have customized a processor's operating system, include the symbol file that was created when you linked the operating system with the **Link** command.
- An indication of front panel STATUS display code.
- An indication of LED lights on the back of each processor board.

- A description of what operation was being performed when the problem occurred.
- Any data and/or programs (source and run files) necessary to demonstrate the problem. These should be copied to a portable medium and sent along with instructions on how to recreate the problem.

For execution conditions:

- Any data and/or programs (source and run files) necessary to demonstrate the problem. These should be copied to a portable medium and sent along with instructions on how to recreate the problem.
- A list of erroneous output with the errors indicated and corrections noted.

For problems that reoccur but cannot be reproduced on demand:

- A clear and detailed description of the situation.
- A listing of the system log file (where applicable).
- Any data that may be available for this type of problem, including listing, and so on.

When completing the FCF, please observe the following guidelines:

- Only one problem should be reported per FCF.
- The FCF form is to be used for reporting system software problems, errors in documentation, or suggestions for new software features. Do not use it to ask questions or request information.
- The information should be as clear and legible as possible.
- All attachments submitted with the FCF should be clearly marked with the FCF number. If you wish anything to be returned to you, including any media you may have submitted, you MUST indicate so on the FCF form and the piece of media. If you do not, your media will not be returned.

Glossary

Applications Processor (AP). Processor board in the XE 500 system that runs the CENTIX operating system.

AP. See **Applications Processor.**

ASCII (American National Standard Code for Information Interchange). Control and graphic character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange between data communications systems.

background process. Process that, once started up, runs underneath any active processes, with no interaction with the user through the terminal.

B 20. Burroughs microcomputer from the clustered workstation series.

block. On a disk device, a 512-byte subdivision of data on the disk. Also referred to as a *sector.*

block device. A hardware device that handles I/O data in 512-byte blocks. Disks are block devices. Block devices can be configured to handle raw data streams. In other words, block devices can be configured to transfer data as though they were character devices.

Bourne Shell. A command-oriented shell used to search for programs in specific places on the CENTIX file system.

BTOS. B 20 Operating System. All XE 500 boards except the Applications Processor run a version of BTOS. BTOS-based processors handle all of the actual data transfers for CENTIX between the XE 500 and I/O devices (such as disk drives, tape drives, terminals, and modems).

CENTIX. Burroughs version of the System V UNIX operating system.

CENTIX BASIC. ANSI-standard BASIC adapted for use with CENTIX.

CENTIX C. Standard C Language adapted for use with CENTIX operating system.

CENTIX COBOL. High-level programming language conforming to ANSI X3.33-1974.

CENTIX FORTRAN. CENTIX adaptation of high-level programming language conforming to ANSI standard X3.9-1978.

CENTIX Pascal. CENTIX adaptation of high-level programming language suited for large projects.

CENTIX shell. Command interpreter; program acting as interface between operating system and users.

centrEASE. A menu-driven, interactive facility that you can use to perform many administrative tasks on the CENTIX system.

centreCAP. A programming tool that can be used to create function-key driven user interfaces.

centreSCREEN. The CENTIX forms programming tool through which forms can be designed and retrieved.

centreSNA. Burroughs software products; provide an interface for CENTIX application products to SNA networks.

centreSPHERE. Environmental and application software.

centreWINDOW. Allows multiple windows at a PT 1500, with each window running its own program.

centrOFFICE. An office information management system for efficient communication and information accessing, including word processing, spreadsheet, and mail exchange.

character device. A hardware device that handles raw data streams. The size of I/O transfers for a raw data stream is determined by the design of the device itself or by the program controlling the device. Terminals and printers are character devices.

Cluster Processor (CP). Board in XE 500 system; runs communications software and supports PT 1500 terminals, B 20 workstations, a parallel printer, and up to three RS-232-C serial devices.

CMS. See **Computer Management System.**

COBOL Animator. Source level debugging tool that allows the programmer to interact with an executing COBOL program.

Computer Management System (CMS). Operating system; run-time system and post-compilation system.

console. The terminal designated by the system software for use by the system administrator.

CP. See **Cluster Processor.**

daemon. A system background process.

data communication (data comm). The transfer of data between a data source and data link using one or more data links according to the designated protocol.

demand paging. A form of memory management that keeps in on-board memory only those parts of the program code and data required for execution.

device. A terminal, printer, disk, tape, or other input/output medium that can be attached to the system. A device can be physical or logical.

device file. In the CENTIX file system, a file in the /dev directory that represents a terminal, printer, disk, tape, or other input/output device.

directory. In CENTIX, a directory is a list of files that are assigned to the directory. A directory can also contain other directories.

disk cartridge. Magnetic disk storage medium utilizing a hard disk enclosed in a portable cartridge. Disk cartridges are used with a disk cartridge drive of an XE 500 base enclosure.

disk extent. One or more contiguous disk sectors that contain all or part of a file.

Disk Processor (DP). Processor board in an XE 500 system that is formed by connecting SC to SP. The DP supports I/O to half-inch magnetic tape drives and MD3 disks.

DP. See **Disk Processor.**

dumb terminal. Unprogrammable terminal that uses ASCII code.

/etc/checklist. File that lists the CENTIX file systems that are checked by **fsck**, the file system checking program, at boot time.

/etc/rc.mounts. File that lists the file systems that are mounted at boot time.

/etc/getty. Process that readies terminal connections for login.

/etc/gettydefs. File that contains line communications information on terminals. It is read by the /etc/getty process.

/etc/inittab*nn*. CENTIX files that defines the CENTIX terminal assignments and the CENTIX processes that are started at boot time.

/etc/passwd. File that lists information about each user on the system, including the user's login name, password, home directory, and so on.

/etc/profile. A shell script that defines the users' terminal environment when the system is booted.

/etc/rc. File used as a general purpose start-up file for various background processes such as the lp spooler scheduler and cron.

File Processor (FP). Processor board in an XE 500 system that supports I/O operations to disk devices.

file system. In CENTIX, a collection of files that are all stored on the same logical disk device. A file system must be attached to, or is "subordinate to," a directory. The file system physically contains the files that are logically contained in that directory. The term can also be used, as in "the CENTIX file system," to describe the entire hierarchy of directories, specific file systems, and files in a CENTIX system.

FP. See **File Processor.**

home directory. For a user, the directory into which the user is automatically placed when he or she logs onto the system.

Indexed Sequential Access Method (ISAM). Programming tool that uses an index to sequence file records on disk and to access those records directly.

INGRES. Relational data base management system.

inode. In a CENTIX file system, there is one inode for each file and directory in the file system. The inode contains status information for its file or directory, such as the size, its owner and permissions, its disk address list, and whether it is a directory, an ordinary file, or a special file.

I/O. Input/output.

ISAM. See **Indexed Sequential Access Method.**

kernel. Portion of the CENTIX operating system that controls system processes and allocates system resources.

Large-Scale Integration (LSI). Monolithic integrated circuits of very high density.

LSI. See **Large-Scale Integration.**

Master Commands. The BTOS commands that can be accessed through CENTIX to administer the BTOS portion of the operating system.

Master Utilities. Utilities that are invoked when the BTOS Master Commands are used.

MCommands. See **Master Commands.**

MD3. Enclosure containing up to three Memorex 166 SMD disk drives.

ME. See **Memory Expansion Board.**

Memory Expansion Board (ME). Board attached to a processor board to supply 1/2 M-byte or 1 M-byte additional memory capacity.

Memory Management Unit (MMU). Part of the AP that supports multiprogramming and demand paging.

mixed system. An XE 500 system that contains a complete BTOS operating system and a complete CENTIX operating system.

MMU. See **Memory Management Unit**.

multiuser mode. An operating state defined in the /etc/inittab*nn* files. In multiuser mode, user terminals are readied for login.

MUtilities. See **Master Utilities**.

ofcli. CENTIX command used to access the BTOS Command Line Interpreter (CLI) mode, from which BTOS MCommands can be initiated.

partition. The name of a BTOS file that is associated with a CENTIX logical disk device.

path name. For a CENTIX file, the name that identifies the file's position in the CENTIX file system. A complete (absolute) path name always begins with /, which stands for the root directory.

Pilf factor. A value that can be specified when you create a file system to control the size of the blocks of data that are moved in and out of the file system in I/O operations.

piping. Linking of programs so that the output of one program becomes the input for another program.

port. The part of a data processor dedicated to a single data channel for receiving data from, or transmitting data to, one or more external remote devices.

printer spooler. A system service that manages the transfer of data from disk files to printers.

QIC. See **Quarter-Inch Cartridge Tape**.

Quarter-Inch Cartridge (QIC) Tape. Magnetic tape storage medium that utilizes quarter-inch-wide tape enclosed in a portable cartridge. QIC tapes are used with a disk cartridge drive of an XE 500 base enclosure.

raw device. A block device configured to accept raw data. The size of I/O transfers for a raw data stream is determined by the design of the device itself or by the program controlling the device.

root. The base directory of the CENTIX file system. Every CENTIX directory must either be subordinate to root, or subordinate to a directory that is subordinate to root, or subordinate to a directory that is subordinate to a directory that is subordinate to root, and so on. In a file path name, root is represented by a slash (/).

Run-Time System. CENTIX shell commands and software to support an office environment running office application programs.

saf. Command entered that initiates the centrEASE administrative facility.

SCCS. See **Source Code Control System.**

sector. See **block.**

shell. The portion of the CENTIX operating system that provides a user interface to the kernel.

shell script. An executable CENTIX file that contains a program comprised of shell commands.

single user mode. An operating state defined in the /etc/inittab*nn* files. In single user mode, only the system console can access the system.

SMD. See **Storage Module Device.**

SNA. See **Systems Network Architecture.**

Sort/Merge. Programming tool that provides sequencing of file records and merging of sorted records from more than one file.

Source Code Control System (SCCS). A group of software commands that control and account for changes to text files.

SP. See **Storage Processor.**

special file. See **device file.**

Storage Module Device (SMD). 132-byte (formatted) Memorex 166 Disk Drive.

Storage Processor (SP). Processor board in XE 500 system; controls half-inch magnetic tape.

superblock. The portion of a CENTIX file system that contains descriptions of the file system, including the file system name, its size in blocks, the number of blocks reserved for inodes, the free inode list, and the free block list.

superuser. The name by which the system administrator is called in CENTIX documentation. To become superuser, the administrator signs onto the system as "root".

system bus. Path over which the system processors communicate.

Systems Network Architecture (SNA). A formal set of rules for designing, building, and operating the components of a data communications network.

terminal. A device, usually equipped with a keyboard and a display, which is capable of sending and receiving information over a communication channel.

Terminal Processor (TP). Processor board in XE 500 system that supports a parallel printer and up to ten RS-232-C serial devices.

TP. See **Terminal Processor**.

UNIX. AT&T Bell Laboratories operating system designed for application program development on various computer systems.

volume. In BTOS, the complete file system unit of information stored on a formatted disk.

XE 550 System. Burroughs multiprocessor computer; runs CENTIX, a UNIX-based operating system.

Index

D

E

L

M

N

O

P

Q

R

S

T

U

V

Title: _____

Form Number: _____ Date: _____

Burroughs Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information.

Please check type of suggestion: ☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments: _____

_____

_____

_____

Name _____

Title _____

Company _____

Address _____
Street City State Zip

Telephone Number ( ) _____
Area Code

Title: _____

Form Number: _____ Date: _____

Burroughs Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information.

Please check type of suggestion: ☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments: _____

_____

_____

_____

Name _____

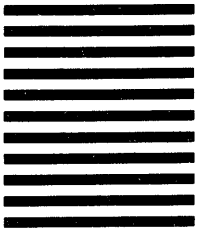
Title _____

Company _____

Address _____
Street City State Zip

Telephone Number ( ) _____
Area Code

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY CARD

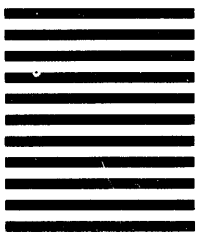
FIRST CLASS PERMIT NO. 817 DETROIT, MI 48232

POSTAGE WILL BE PAID BY ADDRESSEE

Burroughs Corporation
Production Services – East
209 W. Lancaster Avenue
Paoli, Pa 19301 USA

ATTN: Corporate Product Information

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 817 DETROIT, MI 48232

POSTAGE WILL BE PAID BY ADDRESSEE

Burroughs Corporation
Production Services – East
209 W. Lancaster Avenue
Paoli, Pa 19301 USA

ATTN: Corporate Product Information