PLEXUS

Plexus P/35 User´s Manual

98-05043.7 Rev.A January, 1985

Plexus P/35 User´s Manual

98-05043.7 Rev.A January, 1985

PLEXUS COMPUTERS, INC.

3833 North First Street

San Jose, CA 95134

408/943-9433

FCC COMPLIANCE NOTICE

WARNING

This equipment generates and uses radio frequency energy and if not installed and used properly, i.e., in strict accordance with the instructions manual, may cause harmful interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment.

Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense may be required to correct the interference.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

The UNIX* operating system is a large (for a microcomputer) task-oriented system that allows the user to make full use of any combination of the system's software tools to attain a desired effect. It is composed of a tree-based file structure supporting a large, unpartitioned root file system which contains literally hundreds of files, programs, and utilities that can be manipulated in combination to produce countless effects.

The power and complexity of UNIX and the Plexus P/35 dictates that using the system often requires foresight and consideration for the impact that a given procedure may have on the overall system. This means that:

- Each user should have an available reference to functions that can create an impact on the overall system and,
- Some one person must be appointed to take overall responsibility to ensure the soundness of the P/35 and its UNIX operating system. This person is known as the system administrator.

This User's Manual has been written to answer both of those needs — to provide procedures and considerations for operations that affect the soundness of the overall system, whether they are tasks performed by an ambitious user or duties performed by the system administrator. Below is a chapter-by-chapter description of what this manual contains:

Chapter One, Equipment Overview, describes the physical aspects of the P/35, basic design characteristics, and location of the controls.

Chapter Two, Startup, Shutdown, and Tape Drive Operation, tells you how to turn the system on, boot it up, and log in as a user.

Chapter Three, System Administration Considerations, lends insight and advice on the factors you must consider for your system and its specific use. Systems as complex as the P/35 generally require a system administrator.

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

Chapter Four tells how to set up a system for multi-user applications. It fills in the gap between plugging the computer in and supporting a group of interactive users.

Chapter Five provides a ready reference for moving your system back and forth among the four initialization states available as shipped from Plexus.

Chapters Six, Seven, and Eight cover the software configuration considerations when you expand your system by adding users, printers, or disk space.

Chapters Nine and Ten provide practical guidelines on using two of UNIX' more powerful tools; SCCS, the Source Code Control System that tracks multiple versions of a program or document; and UUCP, the UNIX to UNIX Communication Protocol that supports batch operations among UNIX computers, either via phone lines or hardwiring.

Chapter Eleven, The Standalone Environment, describes the programs created by and shipped from Plexus that are designed to maintain and/or modify your software and hardware interfaces.

Chapter Twelve covers the daily, weekly, and monthly routines you must observe to keep your system backed up and running smoothly.

Chapter 13, Crisis Recovery, provides procedures for minimizing the consequences of system crashes, and for identifying problems as they occur.

Finally, this manual provides an appendix, Hardware/Firmware Diagnostics, that provides you with more information as to what could specifically be wrong if your system refuses to boot properly. The information here is designed to give you a better understanding of our equipment and to make any encounters with Plexus Field Service as informative and efficient as possible.

Chapter One

EQUIPMENT OVERVIEW

The P/35 is Plexus Computers' powerful desktop mini. It features a Motorola 68000 microprocessor, distributed intelligent input/output (I/O) control and an industry-standard Multibus* backplane. It runs the UNIX** operating system and is available in 8-user and 16-user versions.

The P/35 is enclosed in a desktop-sized cabinet which contains a tape drive, a rigid-disk drive, a card cage with printed circuit (PC) boards, and power supplies. There is an I/O panel on the back, and a reset/power keyswitch on the front. The card cage contains a 68000-based processor board, one or two intelligent communication processors (ICPs), one or two memory array boards, and a disk and tape controller. These controller boards plug into a six-slot Multibus backplane.

1.1 Main Processor

The processor board contains a Motorola 68000 microprocessor supported by all the necessary buffers and transceivers. It includes a memory map and a variety of local peripheral devices, including a console port, a realtime clock and calendar, a switchpak for configuring certain features, and status LEDs. Performance features include scratch RAM, cache memory, and a variety of read and write registers.

The 68000 microprocessor has a 16-bit data path and a 24-bit address path. This allows it to directly address 16M bytes of memory space. In this system, the space is divided so that the lower 8Mbytes access main memory through the memory map and the upper 8Mbytes addresses processor system resources (i.e. PROM, local registers, I/O, Multibus, etc.). This selection is made by a decode of the high-order address bit (A23).

* Multibus is a trademark of Intel Corporation.

** UNIX is a trademark of Bell Laboratories. Plexus Computers Inc is licensed to distribute UNIX under authority of AT&T.

1.1.1 PROM

The processor board's PROM contains the instructions and exception vectors used during self-test and initialization. It is accessed after reset and until the BOOT self-test is cleared.

1.2 Memory Array

The P/35 dynamic RAM is contained on one or two memory boards which go in the top two slots of the card cage. Standard memory boards provide 1M byte of memory; memory boards are also available in either 256K or 512K increments. The P/35 needs a minimum of 1M byte of memory to operate well.

All memory access is over connector P2, which provides a 32-bit data path. Power and ground are the only connections to the Multibus.

Memory boards provide their own refresh, and error detection and correction (EDC). Single-bit errors are corrected and double-bit errors cause a buss error.

1.3 Input/Output Controller

Intelligent Communications Processors (ICPs) handle data movement between terminals and printers and system memory. They use direct memory access (DMA) to bypass the CPU, freeing it from handling all I/O interrupts. Each ICP controls eight serial ports (terminals or modems) and one parallel port (printer). Plexus configures each P/35 for one or two ICPs at the factory.

1.3.1 Serial Ports

Eight RS232C serial ports support baud rates from 50 to 19.2Khz by programming a Zilog CTC Counter/Timer. Each port has its own DMA channel to move data from local memory to the transmit line.

1.3.2 Parallel Port

The ICP has a single parallel port used to connect an industry-standard (Centronics-type) printer. Control logic associated with the port transfers data to the printer without intervention from the CPU.

1.3.3 Local Memory

The ICP is equipped with 32K bytes of dynamic RAM that stores operating instructions for the ICP and buffers data between transfers.

1.4 Intelligent Mass Storage Controller (IMSP)

IMSP is an intelligent disk and tape controller that contains its own Z8001 microprocessor.

It receives commands from the CPU to move blocks of data between system memory (RAM) and the disk drives or tape drive. It has 128K bytes of local RAM which it uses to buffer a number of sectors, to decrease the number of disk accesses when the system experiences a heavy processing load. These buffers store the information from the disk and pass it to each process as if it were the only process using disk.

The IMSP uses an industry-standard SMD type disk interface. The intelligent tape drive performs many of the functions normally required of the tape controller. It communicates with the IMSP over eight data lines and eight control lines.

1.5 Standard Peripheral Configuration

The IMSP supports a number of industry-standard disk and tape drives, so it is possible to configure a wide variety of systems. However, the Plexus-standard combination of a fixed Winchester disk and streaming cartridge tape drive has proven to be very flexible and cost-effective.

1.5.1 Disk Drive

Plexus' P/35 can be equipped with one of four 8-inch fixed Winchester disk drives in 22, 36, 72, or 142.6 Mb capacities. Additional disk drives may be added in a separate cabinet.

1.5.2 Tape Drive

The P/35 comes equipped with a 45M-byte streaming cartridge tape drive. It has 8000 bpi storage density on 9 tracks, and can store 45M-bytes in approximately 9 minutes. P/35s shipped before May, 1984 have 22 Mb cartridge drives of a 4-track design. The 45 Mb tape drive can read tapes produced on the 22 Mb drive.

1.6 Physical Description

This section describes the physical characteristics of the Plexus computer system, introduces its primary controls, lists the required operating power and environmental conditions, and describes the optional expansion configurations available.

1.6.1 Physical Characteristics

The Plexus P/35 computer is a compact, single-unit, self-contained (except for a terminal) computer system (see Figure 1-1). This unit can be mounted on a table, a shelf, into a standard 19-inch RETMA rack, into a custom Plexus rack, or on top of a Plexus-designed System Expansion Module (SEM).

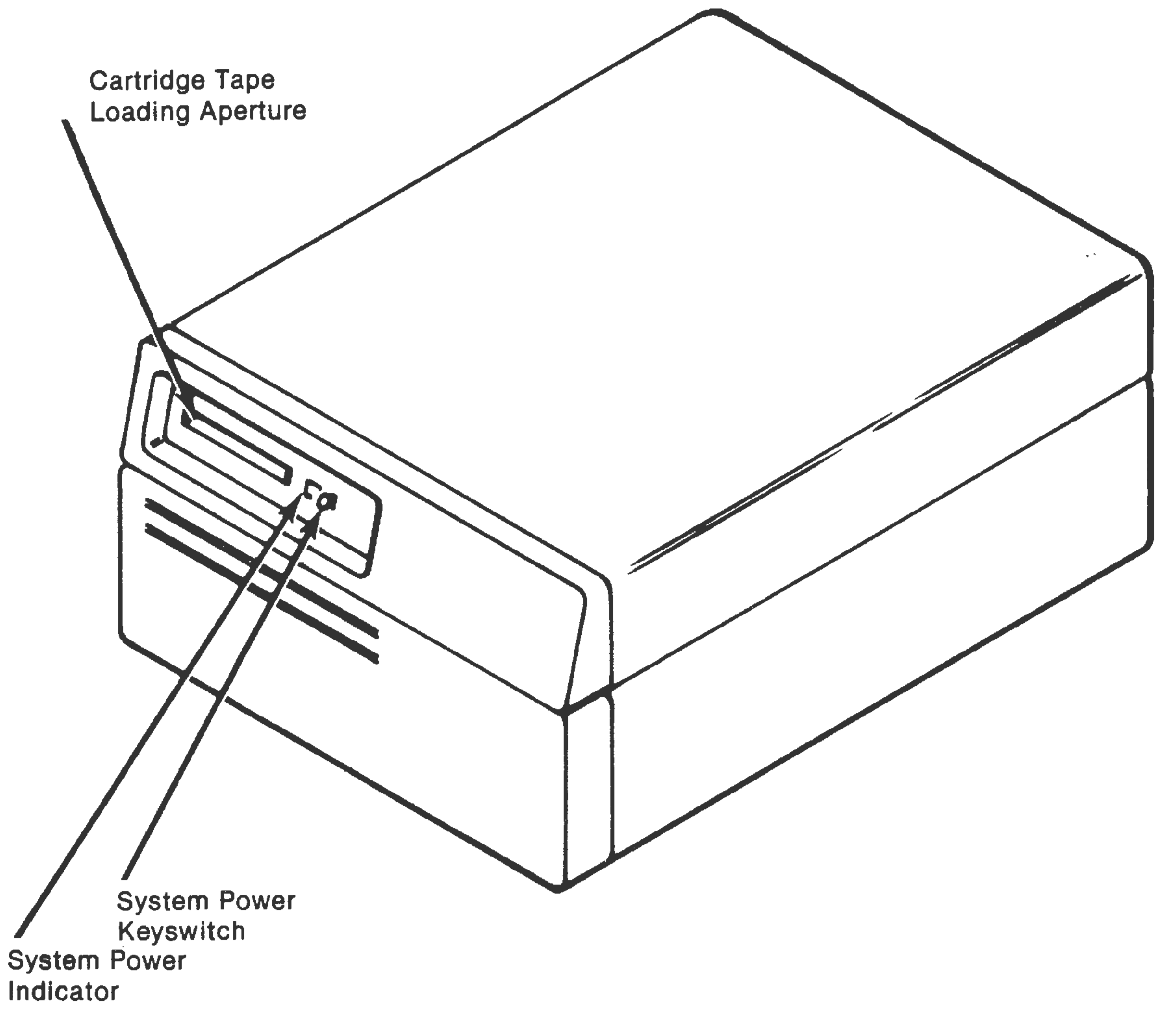

Figure 1-1. Standard P/35 System

The P/35 enclosure houses a cartridge-type tape drive, an 8-inch winchester-type disk drive, a power supply, and up to 6 processor, memory, and controller printed circuit cards (see figure 1-2). Access to this equipment is gained via removable top, and side covers. The front control panel can also be removed when necessary. All input/output connections (including power) are made at the rear panel.

The basic P/35 disk facility can be expanded from one unit up to four disk units by using either the Plexus Storage Expansion Module (SEM) unit or a rack.

The use of a SEM limits the expansion of the disk facility to three disks, one internal and two external (in SEM).

The use of a standard RETMA rack to mount the add-on disk units and, if desired, the P/35 unit, enables up to three 14-inch disk drives or three 8-inch disk drives to be added to the basic system.

A P/35 system can be connected to an Ethernet* Local Area Network (LAN). In such a network, the user can, transparently, access files and devices and can also execute jobs on any other Plexus computer in the network.

1.6.1.1 Controls and Indicators

The only system controls are:

1. A SYSTEM POWER on/off keyswitch and a power on indicator which are mounted on the front panel of the unit and
2. A RESET pushbutton which is mounted on the rear panel of the unit. The RESET pushbutton, when pushed, resets the processor and reboots the system software.

1.6.1.2 Dimensions and Weight

Dimension	*English*	*Metric*
Height	11 in.	(28 cm)
Width	19 in.	(48 cm)
Depth	26 in.	(66 cm)
Weight	85 lb	(39 kg)
Shipping weight	95 lb	(43 kg)

* Ethernet is a trademark of Xerox Inc.

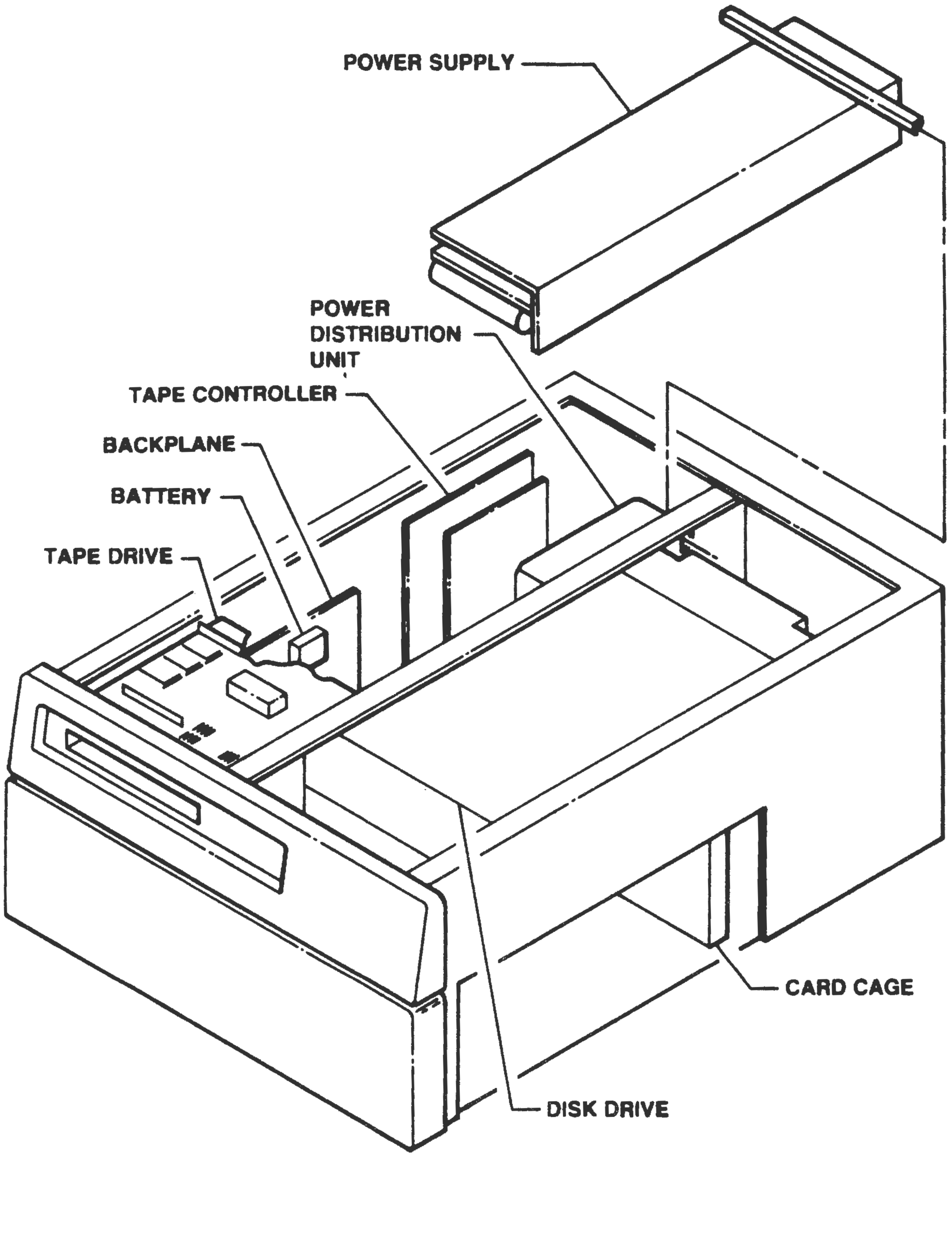

Figure 1-2. P/35 System, Primary Components

1.6.2 Power Requirements

The P/35 computer is available in either a 115vac or a 230vac version. Unless ordered otherwise, Plexus will ship the 115vac version. The power required is:

115vac (+ or - 10%)	49-61hz	(@ 6A)	(Plexus default)
230vac (+ or - 10%)	49-61hz	(@ 3A)	

The amperage indicated is only for a standard configuration, that is, a single disk drive and one memory board. Additional equipment will increase the amperage requirements.

1.6.3 POWER SUPPLY

The P/35 is equipped with a multiple-output, switching power supply which provides over-voltage and short-circuit protection. It supplies the following dc levels:

Volts		Amps(max.)
+5	at	45
+12	at	10
—12	at	10
+24	at	4
—5	at	4

1.6.3.1 Environmental Requirements

The system requires the following environment to operate properly:

Temperature	40-100F (5-38C)
Humidity	20-80% noncondensing

1.6.4 System Expansion Module

A custom designed System Expansion Module (SEM) is available for the P/35 system. This unit is a small cantilever desk designed to hold a single P/35 computer and to expand its disk capability to three disk units (one internal, two external). The SEM (see Figure 1-3) consists of a top, a column and a base. The column can house two 8-inch disk units and their associated cables, the base can house the disk drive power supply units. Instructions for mounting a P/35 system onto a SEM are given in the manual "*SEM Installation Guide*".

1.6.5 Rack Mounted Systems

Plexus does not supply, as standard, rack-mounted P/35 systems. Plexus, however, does supply kits which permit these systems to be mounted into a standard 19 inch rack. Contact Plexus for further information.

1.6.6 Ethernet

To connect a P/35 to an Ethernet network requires the installation of an Ethernet controller board and its applicable network cable into the system card cage, and a cable and a transceiver to connect the P/35 to the Ethernet coaxial network cable. The installation of the Ethernet equipment is described in the manual "*Ethernet Installation Guide*".

Chapter Two

STARTUP, SHUTDOWN, AND TAPE DRIVE OPERATION

This chapter describes how to start up and shut down your Plexus computer. It assumes that the computer is installed correctly, the system console is attached, and all parts are functioning correctly as described in the *Installation Guide*, which is included with this product.

This chapter also contains instructions for the use of the tape drive.

2.1 Normal Startup

When you turn the power on or press the RESET button, the system automatically begins its startup routine. It completes its selftest, then waits for the command to boot the operating system. When the operating system is booted (loaded from the disk), the system comes up to single-user mode (state 1). The system can be used in state 1 or brought to multi-user mode (state 2).

A step-by-step procedure is provided later in this section.

2.2 PROM Selftest and Diagnostics

The system selftest program activates immediately after the system is powered on or reset. As it completes its tests, the message `PLEXUS SELFTEST REVX.X COMPLETE` appears on the console. Partial completion of this message indicates selftest failure. Typically, the words `PLEXUS SELFTEST REVX.X` appear quickly and the word `COMPLETE` takes longer.

Most systems contain an improved bootstrap PROM that monitors system performance during the bootstrap procedure. If the system cannot complete the bootstrap procedure, it returns an error message either directly before or after the message `:/sys3`.

2.3 Startup Procedure

The visual display and necessary action to start the system are shown below. User responses are shown in bold:

```
PROMPT/ACTION                          COMMENTS

+--------------------------------+
|PLEXUS SELFTEST REVx.x COMPLETE|  Indicates system has
                                   passed selftest.


  PLEXUS PRIMARY BOOT REV X.Y      Screen displays boot
  : <return>                       prompt. Press <return> key to
                                   continue, or start standalone
                                   programs.


 :/sys3                            Screen displays program
                                   to be booted, followed
 SYS3/x.x: sys3.xx                 about 30 seconds later by
 real mem = xxxxxx bytes           a report as shown here.
 avail mem = xxxxxx bytes          When the superuser
 sys3                              prompt (#) is displayed,
 single user                       the system is in state 1.

 fsck*                             Plexus Computers recommends
                                   that the program fsck be
                                   run to ensure file system
                                   integrity before doing
                                   init 2.

 # init 2                          Enter this line after
 errdemon started                  the superuser prompt to
 cron started                      change to state 2 and
 initializing ICPs                 send logins to terminals.
 multi-user
 type ctrl-d <ctrl d>              Type <ctrl d> to send login
 login:                            prompt to console.
|                                |
+--------------------------------+
```

*The program **fsck** is described in the *Plexus Sys3 UNIX Programmer's Manual.*

For new installations where no logins have been implemented, login as root (type **root**).

As shipped from Plexus, the erase character is '#' and the kill line character is '@'. To erase a character, follow it with a '#' (EXAMPLE: px#s = ps). To erase a line, type a '@'. The line will be ignored and the cursor will move to the next line.

If the startup sequence fails, see the chapter, "Crisis Recovery" for corrective action.

Systems that shipped with earlier version software but have been upgraded to release 3.1, might retain "pd(0,0)sys3" as the boot pathname. When the <return> is entered, the system displays:

```
PLEXUS PRIMARY BOOT REV 1.0
:pd(0,0)sys3
```

This is an acceptable substitute which does not change the system startup procedure.

CAUTION

To avoid system failure, read the chapter, "The Standalone Environment," before booting any program other than the default ("pd(0,0)sys3" or "/sys3").

2.3.1 Logging In

Plexus systems are shipped with a few active accounts, one of which is **root**. You must create the other user accounts in due time. At this point, however, if you would like to check the login procedure on this system, enter:

root

in response to the `login` prompt. The system will respond by returning some basic messages regarding the revision level of the operating system software, followed by the root or superuser prompt, `#`.

At this point the root account has no password. You can add one now or wait until you encounter the procedure in the chapter, "Setting Up Your Installation," where it is described in detail.

2.3.2 Shutting Down the System

After you have logged in as root (presumably from the system console), you may want to shut the system down before proceeding further. To do so,

1. Be sure you are logged in as root on the system console.
2. Enter:

   /etc/shutdown X

   where X = seconds of grace period (default = 60 sec)
3. The program **/etc/shutdown** sends a series of messages to all active terminals, warning that the system will shut down in X seconds (the length of the grace period mentioned above).
4. The program takes several minutes to execute, during which time the user is prompted for yes/no responses when necessary. When it asks:

```
Busy out (push down) the appropriate phone lines for this system.

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

Hang up any modems and enter:

y

5. The system displays some messages, which may take a few minutes. It displays a list of processes (**ps -eaf**), then asks:

```
Will a file save be done at this time (enter y or n).
```

The appropriate line in the list of processes should read:

```
INIT 1
```

Respond by entering:

n

NOTE

If **y** is entered, **/etc/shutdown** displays the message:

```
fsck will now be executed on files
in checklist
```

The complete **fsck** functions are completely displayed as described in the *Plexus Sys3 UNIX Programmer's Manual.*

6. The system displays the message:

```
Halt the system when ready.
```

7. The system is now in state 1. It may be used in single-user mode, or shut down by pressing the RESET button on the system front panel or turning the SYSTEM POWER keyswitch to its off position.

CAUTION

If the system is used after the shutdown program is finished, enter:

sync ; sync

before pressing the <reset> button or removing power.

2.4 Using the Tape Drive

The standard tape drive included with the P/35 uses self-contained cartridges which greatly simplify operation as compared to standard 9-track open-reel tapes. There are no controls, switches, or indicators on the P/35 associated with the tape. All such functions are handled by software commands. In addition, the cartridge itself features a write-protect switch.

2.4.1 Basic Operation

The tape cartridge itself has a switch labeled `SAFE`. If you turn the tab so the arrow points to `SAFE`, the tape can be read but not written to. Plexus recommends you set all software release tapes to `SAFE` to prevent losing your ultimate backup copies of system and application software.

Other than the `SAFE` switch, basic operation of the cartridge drive consists of inserting the cartridge into the tape slot provided on the front panel of the P/35. Insert the cartridge such that the label is facing up and the `SAFE` switch and center pinch roller face away from you and toward the rear of the tape slot.

2.4.2 Features

P/35s shipped from May, 1984 onward have 45 Mb tape drives with 9 tracks. These units are streaming tape devices, but do not appear to be so during operation. For a given segment, the drive moves the tape back and forth 9 times to completely fill up the tape before moving on to the next tape segment. This feature makes the drive faster to use because it does not have to rewind the entire tape before writing (or reading) the next track.

These newer drives can also read tapes written by the pre-May 1984 20 Mb 4-track tape drives. Figure 2-1 illustrates how this is possible, and also shows the 9-track "sidewinder" pattern in which it streams data onto the tape. Tapes written by the 45 Mb 9-track cartridge drives cannot be read by the 20 Mb 4-track drives, however. Because the 9-track 45 Mb drives operate to such close tolerances, head alignment and tape tension are critical factors in successful transfer of tapes from one drive unit to another.

2.4.3 Warnings

This section describes special considerations for most efficient and successful use of P/35 cartridge tape drives.

2.4.3.1 Retensioning

The tape drive manufacturer recommends that you retension the cartridge tapes before first use, before any read or write operation involving two or more megabytes of tape, and after two weeks of inactive storage. To retension the tape, enter from any terminal:

```
/usr/plx/tape retension
```

This retensioning rule applies to all tape utilities.

2.4.3.2 Specifying Density and Length

When using the **dump** utility with the 45 Mb drive, you will obtain more consistent results by specifying the tape length and density according to the syntax of **dump** as described in the *Unix Programmer's Manual — Vol 1A*. Tapes are 450 feet in length; the density constant is 82. Thus, as an example, to perform a **dump** of **/dev/dk3**, you would enter:

```
dump fsd /dev/rpt0 450 82 /dev/rdk3
```

2.4.3.3 fbackup Capacity

For 20 Mb drives, the tape capacity when using **fbackup** is 36000 512-byte sectors.

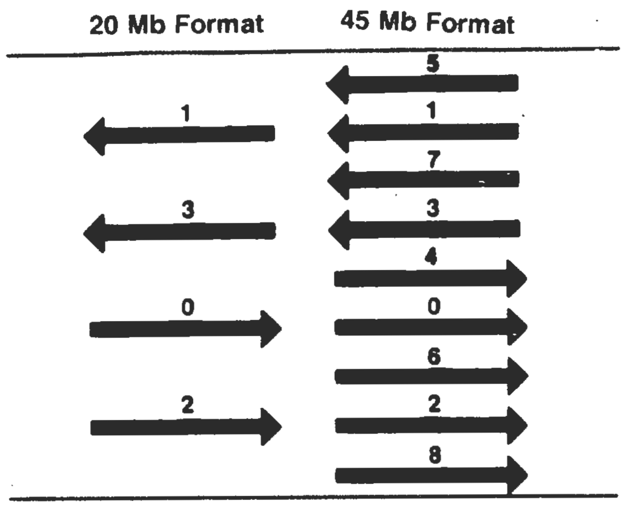

Figure 2-1. 4-Track and 9-track Cartridge Tape Formats

Chapter Three

SYSTEM ADMINISTRATION CONSIDERATIONS

This chapter provides an overview of the factors you must consider to be an effective UNIX system administrator.

The objective of system administration is to keep your system optimized to the purpose of your particular installation. The UNIX system is large and complex enough to allow a wide variety of performance and storage capabilities within a given computer model. It is the system administrator's job to configure and maintain these capabilities so the computer realizes its full potential for a particular purpose.

For example, engineering and accounting departments might both have 8-user, 1 MB RAM, 142MB disk P/35's. The accounting department might assign all 8 ports to secretaries and bookkeepers running word processing and spreadsheet applications on interactive terminals. The engineering department might use only four terminals, but have three different language compilers and a text formatter running at once. These two identical computers should be set up very differently within certain UNIX files so that each system realizes its performance potential for that particular installation. This configuring and system "tuning" is the domain of the system administrator.

3.1 Disk Mapping

When configuring a new system, the normal action is to get started as soon as possible: connect and turn on the system, boot up, log in as root, and modify and add the files and user directories. While it is a good idea to turn on the system, boot it, and log in to ensure that your system works properly, it is best not to proceed any further than that without some off-line planning. You can save yourself much repetitive work if you first determine how much disk space you have and how you want to sub-divide the disk into file systems before adding or modifying anything.

The three primary logical divisions of your disk are:

- The swap area
- The root file system

- The mounted file system(s).

You should allocate your disk space before configuring any files or loading any other software because if you want to increase the size of the root file system or greatly enlarge the swap area, you will have to configure that first with the standalone programs, **dconfig** and **mkfs**, and then reload the UNIX system from your software tape.

Disk mapping includes:

- The size and location of your swap area
- The size and boundary of the root file system if you want one larger than the default (18MB)
- The sizes and boundaries of the other (mounted) file systems
- The directories on which these other file systems will be mounted

3.1.1 Determine Swap Space

The P/35 is shipped with the software installed and the disk mapped. On systems shipped with Sys3 rev. 3.12 or earlier the default swap area is 2 MB. Systems shipped with Sys3 3.2 or later have a factory-set 4- or 6- Mb swap area. If you have an earlier model your installation probably needs a larger swap area. The system administrator must determine the swap area required and implement it. The following factors can help you determine how much swap space is needed:

Factors	Comments
Number of Users	As the number of users increases, so does the amount of processing and swapping required by the system. Swap area should be added in proportion to the number of users projected to use the system at any one time. Score one point for each user you project to be on your system at any one time. ____
Number of Processes	A system with a few users each running several processes at once is equivalent to a system with several more users. Swap space must be increased according to the projected number of simultaneous *processes* regardless of the number of people actually logged in. Score one more point for each *extra* process you anticipate your users will have running in background in addition to interactive tasks. ____

Types of Processing	Installations for software development, documentation, and scientific applications require more swap area than office environments running primarily accounting and inventory applications unless they are on a database management system. Score four points for each database management system, text formatter, and/or language compiler that will be in use on a typical day. ____
Background Tasks	Different systems (and different system administrators) implement varying amounts of automated background processing. This can take the form of cronfiles, printer schedulers, connecting your system to a telephone line network such as USENET, etc. Score four points for each such task your system will run on a fairly regular basis. ____

Now total up your points and divide them by four. Round any fractions up to the next whole number. The result is the number of megabytes of swap space you should implement on your system. This is an approximate answer. If your final figure is fairly high (8 Mb or above), you can probably adjust the swap area up or down by 2 megabytes to avoid complications when writing to **dconfig**. (See the chapters, "Configuring Disks" and "The Standalone Environment" for more information on running **dconfig**.)

3.1.2 Set Up the File Systems

The system administrator must also decide how to divide the rest of the disk space into file systems. File systems can be considered logical, rather than physical, disks. The system administrator assigns each file system a device number and a directory on which it is mounted.

There are two categories of file systems, the root file system, and the mounted file systems. The root file system is available whenever the system has been booted into any initialization state (see the chapter, "Changing System Initialization States"). It must exist for UNIX to run at all. The system administrator can also increase the size of the root file system. Regardless, the system administrator is responsible for setting up the number and size of the mounted file systems. These become active when the system is initialized into a multi-user state.

The following factors are involved in allocating disk space to set up the file systems.

Factors	**Comments**
Logical Functions	Disk space is usually divided up according to the functions important to your installation and the manner in which you want to organize your backup tapes. E.g., in addition to root, you might want separate file systems for user accounts, proprietary files, archives, general access files, the spooler, an on-line manual, etc.
Disk Capacity	You cannot set up more disk space than is physically available. If you have defined file systems totaling 80Mb and you have 72Mb disk space available, you need to either economize the allocations or add another disk. The number and sizes of these file systems are ultimately determined by your system's disk capacity.
User Space	If you elect to have one or more file systems for user accounts, the anticipated number of accounts and the space they occupy will be a major factor in disk allocation.
Enlarged Root File System	Certain installations have two or more kernels that can be booted alternately. Some installations acquire or create additional utilities to run in the root file system. Although the Plexus disk is set up to accommodate some growth in the root file system, enough extra kernels and/or utilities could require that it be enlarged.
Backup Capacity	Most UNIX systems are backed up onto tape using the **dump** utility. **dump** creates an image of the entire logical disk onto tape. Therefore, it is best not to create a file system larger than the capacity of your tape backup facility.
Security	When UNIX is booted into single-user mode, the person at the console often has password-free super-user access to any file on the root file system. This factor may influence the size and contents of the root file system as well.

3.2 Create Accounts

The system administrator ensures that all authorized users and groups have access to the system by creating "user accounts" on the system. This involves adding the person's or group's name to **/etc/group** and **/etc/passwd** files, creating directories in those names, and configuring a **.profile** or **.cshrc** file for that person's account to determine the pathnames and priorities for that person or group.

3.3 Establish Security

The system administrator establishes the levels of accessibility of all files in the system. The areas of control include the following:

Factors	**Comments**
Group Accounts	In addition to creating user accounts, the administrator can establish group level security as well by establishing group ids in **/etc/group**. This practice can separate functions and projects that share the same machine.
Permissions Levels	The system administrator can set and reset the permissions levels of any and all files and programs on the system. This determines what individuals and groups can read, write, and/or execute which files and programs. (See the manual pages on **/etc/passwd**(5), **/etc/group**(5), **chown**(1,), **chgrp**(1), and **chmod**(1,).)
Ownership	The system administrator can change the individual and group ownerships of any files on the system to meet specific accessibility paths. (See **chown**(1) and **chgrp**(1)).
Root Password	The system administrator sets the root password when logging into the system for the very first time after successfully booting it into multi-user mode. The root (or "superuser") password has read/write access to every file and program in the system. The system administrator decides who, if anybody else, also knows the root password.
SCCS	SCCS is UNIX's Source Code Control System, but it can be used to archive, control access, and document change to any files desired. The system administrator decides what files, if any, will be placed under SCCS, who will have read-only access, and who will have change (read/write) permission to the files.

3.4 Configure Device Software

As your office or lab (and consequently your system) grows, you will find a need to add or change peripheral devices attached to your system. When adding terminals, printers, tape and disk drives to Plexus hardware, you also have to alter certain UNIX files or run certain standalone programs to enable the software to "see" the new devices. The individual considerations are indicated below:

Factors	Comments
Interactive Terminals	When adding a video display terminal the system administrator must activate a "getty" for the terminal port in **/etc/inittab**, create the device in **/dev**, find or create a termcap entry in **/etc/termcap**, and enter the termcap entry in **/etc/ttytype** to ensure the terminal will demonstrate all compatible functions with UNIX and the Plexus hardware.
Serial Printers	Serial printer can be attached to the serial ports. The administrator must ensure that device **lp** exists in **/dev** to provide for printer control, find or create a **termcap** entry, reconfigure a line in **/etc/inittab**, and possible write a "filter" program to ensure a workable interface.
Parallel Printers	One parallel printer attaches to the parallel port of each ICP. The system administrator must ensure that device **pp** exists in **dev** and reconfigure a line in **/etc/inittab**. The spooler, **lp** provides for printer control.
Disk Drives	Besides ensuring that an auxiliary disk is installed correctly, the system administrator must run **dconfig** to alter the upper boundary of disk space available, use **mknod** to make nodes for block and character devices corresponding to the new logical disks (file systems) represented by the new physical disk, make new file systems via **mkfs**, **mount** the new file systems to existing or new directories, and add the devices to **/etc/rc** so they will automatically be mounted and unmounted when the system passes between single-user and multi-user states.
Modems	If your system is going to be linked with the outside world via telephone lines, the system administrator must assign a tty port for that line and ensure that the baud rate and other characteristics are established in **/etc/inittab** and possibly **/usr/lib/uucp/L-devices**.

3.5 Perform Backups

The system administrator decides how the contents of the system disk(s) will be backed up onto tape, and when. Below is a table of considerations:

Factors	**Comments**
Who	The system administrator decides whether (s)he will do it, or assign the duty to someone else (preferably a consistent, responsible person with a good attendance record). The system can be programmed to perform the backups automatically overnight, but someone has to take the old tape out each day and stick the new one in.
What	It is up to the administrator as to what parts of which file systems get backed up on particular days of the week or month. This manual suggests a schedule in the chapter, "Periodic System Routines," but it is up to the system administrator to decide what is best for the individual installation. This decision is more crucial if you use the standard P/35 20MB or 45MB cartridge backup, because attempting to back up too much at once could require changing tapes at a time when no one is there to do so. The optional 9-track open-reel backup device provides up to 80 Mb backup space on a 2400 foot reel if using **dump** or **fbackup**. This should obviate tape switching in most cases.
When	A full dump of file systems works best if the file system is first unmounted to ensure that no one changes it during the backup. Since most installations have rather large disk capacities, it is best to perform such backups overnight. Smaller systems, however, might have backups done at lunch time.
How	UNIX offers six different utilities that can copy disk contents to tape — **dump, cpio, tar, dd, fbackup,** and **volcopy**. It is the system administrator's responsibility to know the merits and drawbacks of each of these utilities, and prescribe the one(s) best suited to the installation. (Plexus recommends using a utility that backs up the entire file system.)

3.6 Crisis Intervention

The system administrator must respond whenever the system sends a crucial error message to the console, slows down to an intolerable level, or crashes. Duties within this category include:

Factors	**Comments**
Kill Processes	Runaway processes can quickly use up all available file and swap space. The system administrator may try to identify the delinquent process, invoke the superuser password, and kill the process.
Bring System Down	System "panics," damaged file systems, and other factors may require that the system be brought down for maintenance. The system administrator must see that all users are notified (this can be done automatically with **/etc/shutdown**) so they can quit their files and log out (if possible) to minimize or eliminate any lost files.
Bring System Up	Power failures, brownouts, hardware problems, and kernel problems can cause the system to "go down" automatically. Power fluctuations at night can result in a "dead" system in the morning. The system administrator must bring the system back up and see that an alternate person is also entrusted with that duty.
Repair File Systems	Some types of crashes can damage file systems. The system administrator must respond to any such occurrence at whatever level is necessary. This may involve running **fsck** in init 1 and respond to some prompts and/or **restor**ing the most recent full **dump**, or it may involve running **dformat**, **dconfig**, **mkfs**, and **restor**ing the entire system.
Report Bugs	The administrator should notify Plexus Software Support of any software problems encountered. Plexus may be able to offer a work-around solution over the telephone, mail out a fixed version of the software, or see that it is fixed for the next software release.
Get Help	As system administrator you have a lot of responsibilities, but no one expects you to know everything about the system, or be able to repair components. The system administrator must know when the system needs outside help, and procure it.

3.7 Implement Special Features

Probably your system runs some applications or enhancements such as optional language compilers, a data base management system, word processing, electronic spreadsheet, etc. The system administrator allocates the file system and directory for such software packages and installs it there. Some applications require several megabytes of disk space; the system administrator must locate such applications in sufficiently large file systems. The location also affects the pathnames each user must specify to use the application, and the amount of alteration to user **.profile** or **.cshrc** files.

In addition to installing application programs, the system administrator can implement certain utilities on a UNIX system as shipped from Plexus. These utilities include SCCS (a read/write/change control system for source code and critical documentation) and UUCP (UNIX to UNIX Communication Protocol which allows batch communication between UNIX machines via modem and telephone lines).

The system administrator can elect to automate any number of routine system commands by invoking or writing shellscripts and makefiles. Shellscripts and makefiles can automatically bring the system up multi-user; shut the system down gracefully; perform a schedule of backups; sort, collate, format, and print large documents; prevent runaway processes from exceeding system capacity; and perform any other duties which require a multiple number of tasks in either a predictable or conditional pattern.

3.8 Install New Releases

Throughout the year Plexus issues updated, improved, and further debugged versions of the UNIX operating system. Occasionally, Plexus also issues updates to a current release. When your company receives updates or new releases to the operating system, it is the system administrator's duty to install the new software promptly and correctly, making sure all current files are safe and can migrate to the new environment.

Chapter Four

SETTING UP YOUR INSTALLATION

This chapter contains a sequence of software installation and configuration procedures that minimize repetitive commands and superfluous system startups and shutdowns. Each section contains the procedure for a certain phase in setting up your Plexus computer and the sections themselves are in sequence as well.

The procedures in this chapter assume you have followed all hardware connection and power-on procedures in the *P/35 Installation Guide* before beginning here.

4.1 Run mkfs and/or dconfig (Maybe)

The Plexus utility, **dconfig**, can alter the fundamental organization of your disk(s) if run in the standalone (i.e., UNIX is not booted) mode. Therefore, if you need to change some disk parameters via **dconfig**, do so before booting UNIX. This ensures that the disk is properly set up for your needs before you configure any root files.

You should run standalone **dconfig** before booting UNIX under any of the following circumstances:

- You want your root file system larger than the default 18 Mb.

  You need to change the location of the swap space to map in the extra room for **/dev/dk1** on which the root file system resides. Then you must run standalone **mkfs** specifying the new size of the root file system, followed by re-**restor**ing your Sys3 software from the tape supplied with your system.

- Your system needs more than the default swap area.

  Default is 2 Mb on all systems shipped with Sys3 3.12 or earlier and 6 Mb on systems shipped with 3.2 or later except for 22 Mb versions which have 4 Mb swap.

- You want your swap area moved to a different logical disk than that of the root file system.

If the second or third of the above items apply, the procedures are contained under appropriate headings in the chapter titled "Configuring Disks." The procedure to increase the size of the root file system is contained in this chapter.

If none of the above items apply, proceed to the section, "Booting from Disk."

4.1.1 Invoking dconfig before Booting

The following procedure provides the correct sequences when invoking **dconfig** prior to booting your system, specifically:

a. To make your root file system larger than the default 18 Mb

b. To enlarge or move the swap area

The procedure is divided into main steps and subroutines. Main steps increment by arabic numerals (1, 2, 3, etc.) and represent steps to be followed by anyone invoking **dconfig** for either of the reasons stated above. Subroutines are identified by a condition set in italics (e.g., *If you are increasing the swap area*). Subroutines increment by lower case alphabetical characters (a, b, c, etc.). If the condition of use set in italics does not match your need, proceed to the next subroutine or to the next main step (marked by an arabic numeral).

1. Turn on the system according to the procedure in the *P/35 Installation Guide* but DO NOT BOOT THE SYSTEM.

2. Invoke standalone **dconfig** from tape by entering the command marked in bold immediately after the boot PROM prompt.

```
PLEXUS SELFTEST REV  X.X    COMPLETE

PLEXUS PRIMARY BOOT REV X.X
:dconfig <return>
```

2a. *If you are increasing the size of the root file system*, follow this sub-procedure:

a. Change the swplo sector address to mark the new upper boundary you want for the root file system, or move the swap device to a different **/dev/dk** number altogether. The details of these procedures are in the "Configuring Disks" chapter under the procedure, "Increasing Swap Space."

b. Make sure the new length of **/dev/dk1** does not run into the start sector address of the next logical disk you intend to mount and use on your system. For example, if the **/dev/dk2** start address default is offset 40000 and you increased the swap area to 12000 sectors starting at offset (swplo) 36000, you must change the start sector address of **dev/dk2** from 40000 to 48000 so the swap area does not run into the start of **/dev/dk2**.

c. After exiting **dconfig**, the system again returns the boot PROM prompt. Invoke the standalone **mkfs** command by entering the command marked in bold immediately after the boot PROM prompt.

```
        PLEXUS SELFTEST REV  X.X   COMPLETE

        PLEXUS PRIMARY BOOT REV X.X
        : mkfs <return>
```

d. Create a new larger file system for **/dev/dk1** by entering the following command:

    $$**mkfs /dev/dk1** xxxxx m n

   where xxxxx is the new number of 1K blocks you want for the root file system; m is the "m" or interlace factor, and n is 500 for all systems through Sys3 release 3.2 All systems with an IMSP disk controller (pd) have an interlace of 1. The system responds with a count of the isize and m/n factors, then exits to the boot PROM prompt again when finished.

e. Now reload the root file system from the software tape by invoking standalone **restor**.

```
PLEXUS SELFTEST REV  X.X   COMPLETE

PLEXUS PRIMARY BOOT REV X.X
: restor <return>
```

   When the system returns the standalone prompt, enter the following parts shown in bold:

    $$restor **r /dev/dk1 +nn**

   where nn is the number of dump files on your software tape. The system responds:

```
Spacing forward nn files on tape
Last chance before scribbling on /dev/dk1. <return>
End of tape
Exit 0
```

   Expect a long (~45 minutes) wait between pressing <**return**> and receiving the `End of tape` message.

WARNING

> Do NOT press <reset> before the system returns the `Exit 0` message.

2b. *If you are moving or enlarging the swap area only*, observe the following subprocedure:

a. To enlarge the swap area, specify a higher limit on the nswap prompt and then make sure the beginning sector address of the next highest **/dev/dk#** does not interfere.

b. To move the swap area, specify the **/dev/dk#** of your choice in response to the swapdev prompt, and adjust the swplo and nswap sector values accordingly.

NOTE:

The details of these procedures are contained in the chapter, "Configuring Disks" under the procedure, "Increasing Swap Space."

c. Exit **dconfig**.

3. You are now ready to proceed to the next section, "Booting from Disk."

4.2 Booting from disk

If you have satisfied your needs in the previous section or intend to use Plexus' settings for **/dev/dk1** and swap size and location, you are now ready for this section.

This section shows how to boot UNIX properly and safely from disk, bring it up as a multi-user system, and prepare it for first use as root.

Before you "Boot"

The expression "booting the system" refers to the process of loading the operating system from the disk into system memory. This takes place during the startup procedure when you first press <**return**> in response to the boot PROM prompt.

The boot name is the name of the kernel which is to be loaded into system memory. The system attempts to boot whatever you enter in response to the boot prompt. You may either type something or press <return>. Pressing <return> invokes the default which boots whatever file has been designated as the primary boot pathname. Plexus systems are shipped with the primary boot pathname, **/sys3**. Specifying a kernel by name allows you to keep and load different versions of the operating system. You can change the primary bootname with the program, **dconfig**.

Boot your system from disk using the following procedure.

1. Turn the system on, or if it is on, press <reset> on the computer itself to receive the boot PROM message:

```
PLEXUS SELFTEST REV  X.X   COMPLETE

PLEXUS PRIMARY BOOT REV X.X
:
```

2. To boot from disk, respond by pressing <**return**>. The console then displays the following messages:

```
:/sys3 [or pd(0,0)sys3]

SYS3/x.x: sys3.xx
real mem = xxxxxx bytes
avail mem = xxxxxx bytes
sys3
single user
#
```

The shell default characteristics are such that the erase character is # and the kill line character is @. To erase a character, follow it with a # (EXAMPLE: px#s = ps). To erase a line, type a @. The system will ignore the line and move the cursor to the next line.

3. Whenever bringing a system up, always perform a file system consistency check (**fsck**) to ensure that the disk files conform to the UNIX file system before initializing the multi-user mode. To do so, respond to the superuser prompt (#) by entering the following shown in bold:

   # **fsck**

   The system then incrementally displays a five-phase response over several minutes. The larger the file system, the slower the response. If something is not right, **fsck** prompts you to repair the file system that is damaged. Respond by answering **y** to the questions and then rebooting the system. (The program **fsck** is described in the *Plexus Sys3 UNIX Programmer's Manual — vol 1A*.) Then perform **fsck** again before proceeding to the next step.

4. When **fsck** has finished running satisfactorily, it returns the superuser prompt (#). Bring the system into multi-user mode by entering:

   # **init 2**

   The system responds with the following messages from the console. Respond as signified in bold to get the **login** prompt something like the one immediately below:

   ```
   errdemon started
   cron started
   initializing ICPs
   multi-user
   type ctrl-d <ctrl d>
   login:
   ```

5. Login as root.

   ```
       login: root <return>
   ```

   The system displays some login messages and then returns the superuser prompt (#).

6. Before doing anything else, add a password for root. The root password is also known as the superuser password. The superuser password gives read/write/execute permission on any file, directory, or program on your system. Obviously root should have a password, and it should not be divulged lightly. To set a password for root, type:

   #passwd

The **passwd** program then prompts you to enter and re-enter the password you select to ensure its correct entry.

4.3 Making Things Easier

At this point, you have a system up with one active account. The default shell for root, however, is relatively primitive compared to UNIX's capabilities. It has no pathnames specified, and the system does not know specifically what kind of terminal your console is. By defining a few parameters at this point you can avoid typing lengthy pathnames and enjoy the benefits of a full screen editor. This will make subsequent setup procedures much easier.

1. Specify your terminal to the system so you can use a full screen editor. You should find a mnemonic code in **/etc/termcap** to match your console terminal. To look at **/etc/termcap**, enter:

   ```
   /usr/plx/more /etc/termcap
   ```

   When you find the terminal code, specify it to the system. For example, if your terminal is a DEC VT100* or has the exact same **termcap**, enter:

   ```
   #TERM=vt100
   export TERM
   ```

2. Invoke **/usr/plx/vi** to examine **/etc/ttytype** to ensure that the console port ttytype matches the "TERM" you specified in the last step. If it does not, change it.

3. Create a **.profile** file for root to put in pathnames and terminal characteristics to ease later tasks. Type the following sequence:

   ```
   cd /
   /usr/plx/vi .profile
   ```

4. This creates a new file named **.profile** which is invoked whenever root subsequently logs in. The function of this file is to set up the login environment for root. The login environment includes default paths to be searched to find commands, and default terminal characteristics.

   Add the command paths and terminal characteristics to this **.profile** according to the syntax and example explained in the *UNIX Programmer's Manual — vol 1B* under **profile**(5). Below is another example of a basic **.profile** which sets terminal characteristics and specifies pathnames by priority. Enter everything exactly as shown including the single and double quotation marks.

* DEC and VT100 are trademarks of Digital Equipment Corporation.

```
stty ixon kill "^x" erase "^h" echoe
TERM=vt100
PATH=.:/etc:/usr/plx:/bin:/usr/bin
export TERM PATH
EXINIT='set aw number'
export  PATH TERM MAIL EXINIT
```

5. Write and quit this file.
6. Now log out.
7. Log back in as root. This causes the system to automatically read the **.profile** you created and set the parameters specified there.

4.4 Setting Up File Systems

The procedures in this section depend on your having followed the last one. For example, if you did not set the pathnames in the root **.profile** and you enter the command **mknod**, the system returns an error message because it does not know the pathname (which is **/etc/mknod**).

This procedure provides the sequence for activating the rest of your disk space for system use. It requires you to:

a. Create the logical disk devices in the device directory.
b. Create and size file systems on those logical devices.
c. Mount those file systems to directories.

To perform these procedures correctly, you must know how much disk space you have, how you want to allocate it, and which default start sector addresses correspond to which **/dev/dk#**s in **dconfig**. Reproduced for your convenience are two tables describing these parameters. The first table shows in 1K blocks how much space you have remaining on your disk *if none of the factory settings have been changed*. The second table shows the default start sector addresses for each logical disk. "Old" refers to systems shipped with Sys3 version 3.12 or earlier and "New" refers to those shipped with Sys3 version 3.2 or later when the swap area (and **/dev/dk1**) was increased by 4 Mb. In the second table, the ~ symbol means to the end of the disk. The first table shows what that end is.

TABLE 4-1. Default Sizes (1K blocks) for /dev/dk2

Disk Size	**Old** IMSP (pd)	**New** IMSP (pd)
22 Mb 8"	43	43
36 Mb 8"	13405	9405
72 Mb 8"	47473	43473
142.6 Mb 8"	116272	112272
72 Mb 14"	48085	44085
145 Mb 14"	116170	112170
289 Mb 14"	261120	257120

TABLE 4-2. Factory Settings for Start Sector (512-byte) Addresses

		Early & 22Mb P/35s	*New P/35s*
dk0	[0,0]:	**0,-**	**0,-**
dk1	[0,0]:	**0,40000**	**0,48000**
dk2	[0,0]:	**40000,-**	**48000,-**
dk3	[0,0]:	**60000,-**	**68000.-**
dk4	[0,0]:	**80000,-**	**88000,-**
dk5	[0,0]:	**100000,-**	**108000,-**
dk6	[0,0]:	**120000,-**	**128000,-**
dk7	[0,0]:	**140000,-**	**148000,-**
dk8	[0,0]:	**160000,-**	**168000,-**
dk9	[0,0]:	**180000,-**	**188000,-**
dk10	[0,0]:	**200000,-**	**208000,-**
dk11	[0,0]:	**220000,-**	**228000,-**
dk12	[0,0]:	**240000,-**	**248000,-**
dk13	[0,0]:	**260000,-**	**268000,-**
dk14	[0,0]:	**280000,-**	**288000,-**
dk15	[0,0]:	**300000,-**	**308000,-**

NOTE

Remember that the figures in the first chart are units of 1K blocks whereas the figures in the second chart are 512-byte sectors.

To set up file systems and allocate your disk space, perform the following procedures:

1. Be sure you are logged in as root.
2. Change your working directory to the device directory by typing:

   cd /dev <return>

3. Choose from the tables above which dk numbers correspond to the file system sizes you want to set up.

4. Create the logical disk devices as block devices for each of the dk numbers you chose from the previous step:

   mknod /dev/dkX **b 0** X

   where X is the dk number in both occurrences. The second occurrence establishes that number as the minor device number for system device tracking. Repeat this step for each logical disk you are creating for your system.

5. Create the logical disk devices as raw (character) devices for each of the dk numbers you chose.

   mknod /dev/rdkX **c 3** X

   where again X is the dk number in both cases. Repeat this step to match a raw device for each block device you created in the previous step.

6. Create the file systems to correspond to these logical devices and assign them sizes in 1K blocks:

   mkfs /dev/dkX xxxxx m n

   where again X is the dk number and xxxxx is the new number of 1K blocks you want for the root file system; m is the "m" or interlace factor, and n is 500 for all systems through Sys3 release 3.2 All systems with an IMSP disk controller (pd) have an interlace of 1. Repeat this step until you have created file systems to correspond to each logical *block* device.

7. *If you want special directories* to correspond to any of the file systems you created, create the directories now:

   mkdir /dirpath/dirname

   where dirpath is the path to the directory and dirname is the name of the directory you are creating. For example, if you were creating a file system named **user** under the **/usr** directory, you would enter **mkdir /usr/user**. Repeat this step until you have created all the new directories you want to represent file systems.

8. Mount the logical block devices to the directories you created or chose in the last step:

   mount /dev/dkX /dirpath/dirname

   Repeat this step until you have mounted all the directories you want to represent file systems.

9. Change your working directory to **/etc**:

   cd /etc

10. Invoke a text editor such as **vi** to open **/etc/rc**. Add **/etc/umount** commands and **/etc/mount** commands for the mounted **/dev/dk** numbers you established in the previous steps. Following is an illustration of the sections of **/etc/rc** to alter. Sample additions are shown in bold.

```
# /etc/rc curstate no.-times-prev-in-curstate prevstat
TZ=PST8PDT
export TZ
(
case ${1-2} in
1)
      uname
      echo single-user
      if [ "$2" != 0 ]
      then
            : put umounts here
      /etc/umount /dev/dk4
      /etc/umount /dev/dk10
      fi
      ;;
2|7)
      if [ "$2" = 0 ] ; then
            cp /tmp/Ex* /tmp/Rx* /usr/preserve 2>/dev/null
            rm -f /tmp/* /usr/tmp/* /etc/utmp
      fi

      if [ "$2" = 0 -o "$2" = "" -o "$3" = 1 ]
      then
      echo `initializing IMSP`
      sync
      sleep 3
      /etc/dnld -dd -a 5000000 -f /usr/lib/dnld/imsc -o /dev/im0
      /etc/devnm / | grep -v swap | grep -v root | /etc/setmnt
      : put mounts here
      /etc/mount /dev/dk4 /user
      /etc/mount /dev/dk10 /usr/spool
      /etc/update
      rm -f /usr/adm/acct/nite/lock*
      if [ "$1" = 2 ] ; then
#           /bin/su - adm -c /usr/lib/acct/startup

      . . . etc.
```

Write and quit this file.

11. It is more convenient and safer for the system to automatically perform a file system check on all active file systems when you invoke **fsck** in single-user mode. To do so, invoke a text editor such as **vi** to open the file, **/etc/checklist**. Insert the logical device names you want checked. For example, the **/etc/checklist** corresponding to the **/etc/rc** in the previous step would look like this:

```
/dev/dk1
/dev/dk4
/dev/dk10
```

12. Shut down the system gracefully (see the chapter, "Changing System Initialization States" for details).

13. Reboot the system and bring it up multi-user as described in the section, "Booting from Disk," in this chapter.

4.5 Setting Up /lost+found

When UNIX crashes, it attempts to preserve any open files by moving them to the directory, **/lost+found**. For this to work, however, there must be some space set up within the **/lost+found** directory. To set up space you must create some files and then erase them. This creates space but removes the pointers to the space; it cannot be transferred on the software distribution tape. The procedure for setting up **/lost+found** is given below:

1. Make sure you are *not* using the C-shell. If you are, enter:

   ```
   sh
   ```

2. Change your working directory to **lost+found**.

   ```
   cd lost+found
   ```

3. Create some null length files. The number of files you need to create is relative to the number of open files you anticipate your system supporting at any one time. Small systems may need about 10 files. Larger systems should have at least 20.

   ```
   >a
   >b
   >c
   >d
   :
   :
   >n
   ```

   where n represents the last of the files you created.

4. Now remove all the files you created:

   ```
   rm *
   ```

Your **lost+found** directory is now set up.

4.6 Setting Up User Accounts

Now that you have set up file systems to make use of all your disk space, and assuming you have chosen a directory for the system's users, your system is ready to be set up for user accounts.

Setting up user accounts involves the following elements:

- Create a login directory under each user's name.
- Insert this name and certain parameters into **/etc/passwd** and optionally **/etc/group**.
- Create a general **.profile** or **.cshrc** file and copy it into each user's directory.
- Optionally create a general **.login** file and copy it into each user's directory.
- Optimize the **.profile**, **.cshrc**, and **.login** files for each user's account.

4.6.1 Create login Directories

The following is the procedure for creating login directories:

1. Login as root.
2. Change your working directory to the one in which you have decided to locate the users' accounts:

   cd ./userdir

   where you specify the full pathname to the user directory you have created. For example, if your user directory is **/usr/user**, type **cd /usr/user**.
3. Create directories named after each person or account assigned to your system with the following command:

   mkdir username

   where username is the name of the person or account being created. For example, to create an account for someone named Fred in your user directory, you might enter **mkdir fred**.
4. Repeat the previous command until you have created directories for everyone who will have login privileges to the system on a regular basis.

4.6.2 Creating /etc/passwd and /etc/group Entries

The following procedure describes how to configure the /etc/**passwd** file as a part of creating user accounts.

1. Be sure you are logged in as root or become superuser.
2. Change your working directory to /**etc**:

   cd /etc
3. Make sure you are familiar with the /etc/**passwd** file and the information fields required for each entry. These are described in **passwd**(5) in the *UNIX User's Manual, Vol. 1B*.
4. Invoke a text editor such as **vi** to open the /etc/**passwd** file.
5. Create entries in /etc/**passwd** for each user name represented by a directory in step 3.
6. Create and fill in the appropriate fields (separated by colons) for each of these user entries. Below are two different examples:

```
john::103:1:John Niswonger:/user/john:/usr/plx/csh
tj::104:1:Theresa Jackson:/usr/user/therese:/bin/sh
```

   From these entries we see that john's numerical userid is 103 and tj's is 104; we know their real names to avoid confusion during turnover; John's home login directory is /**user** whereas Theresa's is /**usr/user**; and that John uses the "C" shell while Theresa uses the standard, or Bourne shell.

NOTE

> When assigning userid numbers, start at 102 and progress from there. Userid numbers 1 through 101 are reserved for system use.

Note that you do not enter anything in the password field itself (second from the left). That space gets filled with an encrypted password automatically when the individual user invokes the **passwd**(1) program as you did for root. If, for security reasons, you want passwords for all users before they even log in, then successively login under each user name, invoke **passwd**, and create one for that account. The new user can change the password when (s)he logs in later.

7. Write and quit (:wq!) the **/etc/passwd** file.
8. *If you plan to use group security as well*, use a text editor to open **/etc/group** and add the entries and fields appropriate to your system use. Make sure you are familiar with the **/etc/group** file and the information fields required for each entry. These are described in **group**(5) in the *UNIX User's Manual, Vol. 1B*. Write and quit this file. If you will not be using group-level security, ignore this step.

4.6.3 Create .profile and/or .cshrc

Creating and configuring .profile and/or .cshrc files for the system's user accounts saves much time and confusion when the users log onto the system and try to use it. Without one of these files, the users would have to specify the paths to each command they invoked, and some Berkeley-originated utilities would not work properly. Follow the procedure below to set up .profile and/or .cshrc files.

1. Create a model **.profile** for accounts logging into the **/bin/sh** (standard shell) environment and a model **.cshrc** for accounts logging into the **/usr/plx/csh** ("C" shell) environment. Below are a sample **.profile** and **.cshrc**:

   Sample .profile

```
stty erase "^h" kill "^x" echoe
TERM=vt100
PATH=:/usr/plx:/bin:/usr/bin:/etc
PS1="P35> "
export TERM PATH PS1
```

Sample .cshrc

```
setenv EXINIT `set aw wm=15 number redraw`
setenv TERM vt100
setenv PATH .:/usr/plx/:/bin:/usr/bin:/etc:
setenv MAIL /usr/mail/username
setenv TZ PST8PDT
setenv SHELL /bin/csh
stty erase `^h` kill `^x` ixon echoe
set history=20
set prompt=` % `
alias a alias
alias h history
alias lo logout
```

Notice that the **.cshrc** contains more parameters than the **.profile** because the Bourne shell does not support aliases or the history function whereas the "C" shell features both.

Also notice that the MAIL environment must specify the account's actual user name.

2. Copy your model **.profile** into each user directory that will use the Bourne shell as its environment:

    cp .profile userdir/.

    Repeat this command for each appropriate account.

3. Copy your model **.cshrc** into each user directory that will use the "C" shell as its environment:

    cp .cshrc userdir/.

    Repeat this command for each appropriate account.

4. Use a text editor to open each user's **.profile** or **.cshrc** and specify the individual's account name on the line specifying the MAIL environment.

5. Write and quit (:wq!) this file.

6. (Optional) **For accounts running "C" shell,** you may want to create **.login** files as well as **.cshrc** files. The difference is that the **.login** file sets parameters for the login shell only, whereas the **.cshrc** sets parameters for all forked shells as well. The **.login** file is not necessary to system or user operation, however. Refer to the chapter, "An Introduction to the C Shell" in the *Sys3 UNIX Programmer's Manual, vol 2C* for a more detailed description of the function of the **.login** file and suggestions for appropriate parameters.

4.6.4 Changing Individual and Group Ownerships

A basic UNIX principle is that anything you create is yours. If you log in as root and create directories and files, all those directories and files belong to root. Since root set up login directories for all the user accounts, root owns them. Now you (as root) must relinquish the ownership of the relevant directories and files to the accounts designed to use them.

Change ownerships of login directories and special files with the following procedures:

1. Make sure you are logged in as root.
2. Change your working directory to the user directory.
3. Change the ownership of each user's home directory to the owner's login name. Enter:

   chown username username

   For example, to give sam ownership of his own file, enter **chown sam sam**.
4. Change the group (**chgrp) of the user's home directory to the group number of his line in /etc/passwd.**:

   chgrp userno. username

   For example, if sam's userid number in **/etc/passwd** was 118, enter **chgrp 118 sam**.
5. If you want each account to own his or her **.profile, .cshrc**, and/or **.login** file(s), change ownership of each of these files with **chown** and the standard syntax:

   chown username **.profile**
   chown username **.login**
   chown username **.cshrc**

   assuming you have **cd**'d to the user's login directory.
6. If your terminals are properly connected and configured, your system is now ready to accept logins from everyone assigned to it.

4.7 Modifying Maximum File Size

To prevent large files from being created by mistake (e.g. runaway processes) Sys3 includes a mechanism to limit the maximum size of any file which a process can create or access. The system default is set at 2 Mb. The system administrator may allow individual users or all users to create and access larger files. The following procedure modifies the maximum file size limit for all processes on a system.

1. Log in as root and position yourself in **/etc**.
2. Enter the following C program and name it **unlimit.c**.

```
long ulimit();
main (argc,argv,envp)
int argc;
char **argv;
char **envp;
{  ulimit(2,10000000);
   execve("/etc/getty",argv,envp);
}
```

3. Compile this program using cc or ncc and place the executable file in **/etc**. The following is an appropriate command line:

```
cc unlimit.c -o /etc/unlimit
```

4. Modify **/etc/inittab**, changing all references to **getty** to references to **unlimit**. The following sub-procedure will accomplish this.

   a. Create a backup copy of **/etc/inittab**:

      ```
      cp /etc/inittab /etc/inittab.bu
      ```

   b. Invoke a text editor such as **ex** or **vi** to edit **/etc/inittab**.

   c. Make a global substitution, replacing **getty** with **unlimit** in all instances that it occurs in the file.

   d. Write and quit this file.

5. Ask all users to log off the system and bring it down with **/etc/shutdown** .

6. Bring the system back up to multi-user mode as documented in the chapter, "Changing System Initialization States".

When you execute the command **ps -ef** after performing this procedure, all login ports which are not active will have `/etc/unlimit` displayed in the right-hand column instead of `/etc/getty`. If this is not the case, or you note that the process identification numbers are increasing rapidly and users cannot login, the **unlimit** program has not been installed properly.

Check to see if **unlimit.c** compiled correctly and if the executable file is in **/etc** and has acceptable access permissions. Please note that you must repeat the last step of the above procedure to install any corrections.

If you are unable to change the system file size limit successfully, restore the original configuration and call the Plexus Software Support Center.

a. Run **/etc/shutdown** to bring the system down to single-user mode.

b. Save the new copy of **/etc/inittab**.

c. Move your backup copy of inittab (**/etc/inittab.bu**) to **/etc/inittab**.

d. Bring the system back up to multi-user mode.

Chapter Five

CHANGING SYSTEM INITIALIZATION STATES

As delivered, Plexus Sys3 UNIX is configured to run in any one of four system states. Each state is defined by what parts of the operating system are active at the time. Knowledge of what state your system is in is important because certain conditions require changing system states. The different states and the parts of the system that are active in each state are listed below:

State	*Functions*
1	System console only
2	System console, all terminals, **/etc/cron**, **/etc/openup**, **/etc/update** and **/etc/eccdaemon**
7	System console and **/etc/update**
8	Autoboot* mode

5.1 Finding the Current State

During normal, multi-user mode, the system runs in state 2.

If you are not in multi-user mode, you can find out what state you are in by entering the command:

```
ps -ef
```

The resulting display has one line listed in the "command" column as `INIT x` where x is the system state number.

* Autoboot mode also requires that switch 3 (of the 8-position DIP switch) be set to ON on the main processor board (CPU).

5.2 Changing States

The following procedures are described in this section:

- Going from state 2 to state 1
- Going from state 7 to state 1
- Going from state 1 to state 2
- Going from state 1 to state 7
- Using state 8

5.2.1 State 2 to State 1 via /etc/shutdown

Sys3 UNIX provides a program called **/etc/shutdown** for getting from multi-user mode (state 2) to single-user mode (state 1). It kills all the necessary processes in the right order and sends appropriate messages to active terminals. Perform the following steps to initiate a shutdown:

1. Login on the console as root. You must be on the system console, in the root directory.
2. Enter:

   /etc/shutdown X

   where X = seconds of grace period (default = 60 sec)
3. The program **/etc/shutdown** sends a series of messages to all active terminals, warning that the system will shut down in X seconds (the length of the grace period mentioned above).
4. The program takes several minutes to execute, during which time the user is prompted for yes/no responses when necessary. When it asks:

   ```
   Busy out (push down) the appropriate phone lines for this system.

   Do you want to continue ? (enter y or n)
   ```
5. Hang up any modems and enter:

   y
6. The system displays some messages, which may take a few minutes. It displays a list of processes (**ps -eaf**), then asks:

   ```
   Will a file save be done at this time (enter y or n).
   ```

The appropriate line in the list of processes should read:

```
INIT 1
```

Respond by entering:

n

NOTE

If y is entered, **/ect/shutdown** displays the message:

```
fsck will now be executed on files
in checklist
```

The complete **fsck** functions are completely displayed as described in the *Plexus Sys3 UNIX Programmer's Manual*.

7. The system displays the message:

   ```
   Halt the system when ready.
   ```

8. The system is now in state 1. It may be used in single-user mode, or shut down by pressing the RESET button on the system front panel or turning the SYSTEM POWER keyswitch to its off position.

CAUTION

If the system is used after the shutdown program is finished, enter:

sync ; sync

before pressing the <reset> button or removing power.

5.2.2 State 7 to State 1

To get from state 7 to state 1, perform the following steps:

1. Login on the console as root. You must be on the console, in the root directory.
2. Enter:

   /etc/init 1

3. Enter:

   ps -ef

4. Read the process number (PID) of **/etc/update** from the display.
5. Use the command **kill** to kill **/etc/update**. Enter:

   kill -9 XXXX

   where XXXX is the process number (PID) of **/etc/update**.
6. System is now in state 1. To verify, enter:

   ps -ef
7. The command line of the display should read:

   ```
   INIT 1
   ```

5.2.3 State 1 to State 2

To take the system from state 1 to state 2 involves two simple steps:

1. From the console, which is the only active terminal, enter:

   init 2

   The console displays:

   ```
   #cron started
   initializing ICPs
   Multi-user
   type ctrl-d
   ```

2. Type a character **d** while holding down the control (ctrl) key.

   The system should respond by sending login messages to all terminals, including the console.

WARNING

Plexus recommends that you always invoke **fsck** on all file systems while in init 1 before invoking init 2.

5.2.4 State 1 to State 7

Going from state 1 to state 7 is similar to the above procedure. On the console, enter:

init 7

The system starts **/etc/update** and sends a login message to the console.

5.2.5 Autoboot Mode (State 8)

Sys3 has an autoboot feature, which causes the system to boot itself automatically after a system reset. Implementing autoboot involves setting switch 3 ON on the 8-position DIP switch on the CPU board. Detailed instructions for implementing this feature are in the *Plexus P/35 Engineering Manual.*

When autoboot is running, the system runs in state 8 until the autoboot mode automatically implements the multi-user mode (state 2).

5.3 Shutting Down UNIX Gracefully

If it becomes necessary to shut down your computer for any reason (repairs, known power outage, etc.), certain steps can be taken to protect it. This section describes the steps and provides instructions for their implementation.

As shipped from Plexus, Sys3 UNIX runs in one of four system states. Each state is defined by what parts of the system are operational at the time. The system can be shut down gracefully only from state 1.

To shut down the system gracefully, use the following procedure:

1. Using the procedures that appear earlier in this chapter take the system to state 1.
2. To update the super-blocks and flush the system buffers, enter the command:

   ```
   sync ; sync
   ```

   CAUTION

   The two **sync** commands must be the last items entered before the system is powered off.

3. Set the keyswitch on the system front panel to its OFF position.

Chapter Six

CREATING USER ACCOUNTS

This chapter provides instructions for implementing the login ID and home directory required for each new user.

It also provides two optional items: instructions for creating a directory where all users' home directories will reside (necessary only on new installations where one does not already exist), and instructions for setting the users' environments in their home directories.

This section provides instructions which accomplish the stated tasks in one of many possible ways. For additional information, refer to the following items in the *Plexus Sys3 UNIX Programmer's Manual.*

Subject	**Command/Title**
password	passwd(1)
group	group(5) chgrp

profile	.profile(5)
	.cshrc(5)
	.login(5)

terminal type	termcap(5)
	ttytype(5)

shell	sh(1)
	csh(1)

6.1 Procedure to Add a User

A user must be equipped with a login ID and a home directory. To provide this, you must:

a. Add a descriptive line in the file **/etc/passwd**

b. Create a home directory.

c. Assign that directory to the new user.

The following procedure provides instructions for adding a new user. It also provides steps for creating a directory to contain the user's home directories.

1. Login as root or or become superuser.

2. If there is no directory where users' home directories reside, create one using the following steps:

   a. Access the directory root (/) by entering:

      cd /

   b. Enter the command:

      mkdir userdir

      NOTE:

      The name of this directory is a customer option. Throughout this section, the name **user** is used to represent the directory in which users' home directories reside. If these are in a directory with a different name, use that name in place of **user**.

3. Using one of the editors, add a line like the following at the end of **/etc/passwd:**

   loginname::n1:n2:optional:/user/homedir:

   where:

loginname	is name by which the user will login
::	is the field where an encrypted version of the user's password will appear after the user selects a password.
n1	is the user number (use a unique number, 102 or greater for each user)
n2	is the group number. Dividing users into groups requires collaborating changes to the file /etc/group. In most cases the group number will be 10 or greater.
optional	This space was created for use by a system that is no longer supported. It often contains the user's full name for information purposes. If not used, create an empty field by entering the two colons next to each other (::).
/user/homedir	is the pathname to the directory where the user starts after logging in. The home directory usually (but not always) has the same

name as the loginname.

shell	is the field after the last colon (:). It defines the shell that the user starts under after logging in. If blank (like the example), it defaults to the standard (Bourne) shell (/bin/sh).
EXAMPLE:	The following line describes a user "sam". He is user 120 in group 11, his full name is "Sam Jones", his home directory is /user/sam and he uses the standard (Bourne) shell:

```
sam::20:11:Sam Jones:/user/sam:
```

After he selects his password, the line may look like this:

```
sam:2/xyzsP:20:11:Sam Jones:/user/sam:
```

4. Move to the directory /user and create a home_directory (this directory usually has the same name as the user's loginname).

   EXAMPLE: **mkdir sam**

5. Change the owner of the directory "home_directory" to "loginname".

   EXAMPLE: **chown sam sam**

6. Change the group (chgrp) of the user's home directory to the group number of his line in **/etc/passwd.**

   EXAMPLE: **chgrp 11 sam**

6.2 Setting the User's Profile

The user's environment can be controlled by creating a file called .profile in the user's home directory. UNIX automatically scans for .profile in the user's home directory and implements the commands there every time the user logs in. While a number of things can be done with .profile, only these three basics are covered here:

1. Set the command path -- When a user enters a command, UNIX looks for the command in the places specified in the user's command path.
2. Set the terminal type -- UNIX needs to look up information about the user's terminal type for certain UC Berkeley-originated utilities.
3. Export the command path and terminal type (PATH and TERM) -- This makes the information available to other shells.

The following is an example of the contents of a .profile file:

```
PATH=:/usr/plx:/etc:/user/sam/bin:/bin:/usr/bin
TERM=vt100
export PATH TERM
```

6.2.1 Command Path

The command path is the first line in the .profile example. When a user enters a command that is not part of the standard shell, UNIX searches in all the directories in the command path until it finds the command.

The format of the command path line is:

```
    PATH=:/xxxx/xxxx:/yyyy/yyyy:/zzzz/zzzz
```

where the colon (:) is a field separator that separates the different path names.

NOTE

> To avoid confusion, remember that in this section, it is assumed that all users' home directories reside in the directory /**user**. This is not true of all installations; substitute the proper name for the word **user** if another directory name is used.

A typical system has commands in **/usr/plx, /etc, /bin** and **/usr/bin.** Commands and shell procedures created especially for /user/sam usually go in **/user/sam/bin.** Before /user/sam can execute the commands in these locations, the pathnames must be specified in his .profile file.

Other directories can be added to the path list. For example, if Sam wanted to use the commands in Sara's bin, he could add **/user/sara/bin** to his path list. However, there is a cost in system resources; UNIX scans every location in the path list whenever a command is entered. To promote efficiency, keep the path list as short as possible and order the directories so that the most often used appear earlier in the list.

To initialize the changes to .profile, enter the command:

sh /user/home_directory/**.profile**

EXAMPLE: **sh /user/sam/.profile**

6.2.2 Terminal Type

Because of the differences between terminals, some of the UC Berkeley programs need to know what type of terminal they will interface. Two files contain this information: **/etc/termcap** and **/etc/ttytype.**

The file **/etc/termcap** lists all the common terminal types and provides some code to deal with their various differences. The file **/etc/ttytype** is a list of system ports, which details the terminal type attached to each port. It is keyed to **/etc/termcap**; the terminal names in **/etc/ttytype** are used to access the data in **/etc/termcap**.

There are two ways to make this information available to .profile. If the user will be working from only one terminal type, add the following line to the .profile file:

TERM = termtypeXXXX

where termtypeXXXX is one of the terminal names provided in **/etc/termcap** (some terminal types have many names in **/etc/termcap**).

For users who regularly switch terminals, add the line:

```
TERM = `tset`
```

The tset command looks up the terminal type in **/etc/ttytype** and enters that name in the space between the accents. Use the grave accent (`), not the single quote (').

6.2.3 Exporting PATH and TERM

To make the command path (PATH) and terminal type (TERM) available to other shells, add the line:

```
export PATH TERM
```

to the .profile file after the lines for command path and terminal type. This appears in the example of a .profile file earlier in this section.

Chapter Seven

ADDING PRINTERS

This chapter provides information required to configure the appropriate software files when connecting printers to a Plexus computer. The procedures include how to connect a printer to the spooler, how to reassign the spooler to another port, and how to connect an additional printer to a port without a spooler.

All printers attach to the system I/O panel, which is described and illustrated in the *P/35 Installation Guide*.

Pinouts of the RS232C serial connectors and the Centronics-type parallel connector are listed in Tables 7-1 and 7-2.

7.1 Using the Spooler

Sys3 UNIX provides a spooler to queue print requests to a single parallel port (the default is parallel port 0). Other ports can drive printers, but there is no queueing capability unless the spooler is reassigned.

In the default spooler configuration, files piped to **lpr** are queued and printed in the order they are received on the special file **/dev/lp,** which is assigned to parallel port 0 in ICP 1. To use this spooler at its default location, attach the printer cable to PAR 0 on the system I/O panel.

TABLE 7-1. Serial Printer Pinout Connections

Serial Connector Signals (RS232 Standard)		
Pin #	**Description**	**Direction**
1	(not used)	
2	Transmitted data	TO dce*
3	Received data	FROM dce
4	Request to send	TO dce
5	Clear to send	FROM dce
6	Data set ready	FROM dce
7	Signal ground (common return)	(not applicable)
8	Carrier detect	FROM dce
9	(not used)	
10	(not used)	
11	Speed select	TO dce
12	(not used)	
13	(not used)	
14	(not used)	
15	Transmit clock	FROM dce
16	(not used)	
17	Receive clock	FROM dce
18	(not used)	
19	(not used)	
20	Data terminal ready	TO dce
21	(not used)	
22	(not used)	
23	(not used)	
24	(not used)	
25	(not used)	
*dce = data communication equipment.		

TABLE 7-2. Parallel Printer Pinout Connections

Parallel Connector Signals (Centronics Standard)			
Signal Name	**Interface Pin#**	**Source**	**Description**
DATA STROBE/	1,19	Processor	Pulse clocks data to printer
DATA 1	2,20	Processor	LSB)
DATA 2	3,21	Processor	)
DATA 3	4,22	Processor	)
DATA 4	5,23	Processor	) LOW = 0
DATA 5	6,24	Processor	) HIGH = 1
DATA 6	7,25	Processor	)
DATA 7	8,26	Processor	)
DATA 8	9,27	Processor	MSB)
ACKNLG/	10,28	Printer	Acknowledges receipt of character or end of function
BUSY	11,29	Printer	[NOT USED]
PE	12	Printer	Out of paper
SLCT	13	Printer	Printer selected
Gnd	14	Printer	Signal ground
Gnd	16		Signal ground
INPUT PRIME/	31,30	Processor	Initializes printer
FAULT/	32	Printer	Printer fault
NOTE 1. Second pin number (number after comma) indicates the ground wire of a twisted pair return. NOTE 2. The symbol / indicates "negative true" (i.e. logical NOT).			

7.2 Reassigning the Spooler

To assign the spooler to another port, you must remove the file **/dev/lp** and link the name **/dev/lp** to the device file of the new port. The following procedure shows how:

1. Login as root or become superuser.
2. Remove the file **/dev/lp**. by entering:

   rm -f /dev/lp

3. Link the device file of the new port to **/dev/lp.** Enter either:

   ln /dev/ttyxx /dev/lp — for serial port
   or
   ln /dev/ppx **/dev/lp** — for parallel port

   where x and xx stand for the device numbers.

4. If you are adding a serial printer, complete the procedure "Adding a Serial Printer" in this section. A parallel printer need only be connected to a parallel port on the system I/O panel.

7.3 Assigning a Printer with No Spooler

To connect a printer to a port with no spooler, link the port's device file to another device file (/**dev**/name). Requests to print to the new device file go to the port having a device file linked to it.

Use the following procedure:

1. Login as root or become superuser.
2. Change your working directory to the device directory:

   cd /dev

3. Link the device number to the device name by entering:

   ln /dev/ttyxx /**dev**/name -- for serial ports
   or
   ln /dev/ppx /**dev**/name -- for parallel ports

   where x and xx are the device numbers, and name is the name used to identify the port for print requests.

   For example, connecting an NEC Spinwriter* to tty12 could translate in to the command, **ln** /**dev/tty12** /**dev/nec**, which links serial port 12 to /**dev/nec**. To print a file on this port, direct output to /**dev/nec**.

4. If a serial printer is being added, complete the procedure in "Adding a Serial Printer" in this section. A parallel printer need only be connected to a parallel port on the system I/O panel.

7.4 Adding a Serial Printer

Any serial I/O port can drive a serial printer. Because only one spooler is implemented, additional printers must be used carefully; overlapping requests to print on a port without a spooler port will cause unacceptable results.

Because of the flexibility of UNIX serial ports, certain parameters must be set before a serial printer can be attached to a serial port. The following procedure shows how:

1. Modify /**etc/inittab.**

   Serial printers must be connected to non-login (i.e. output) ports. The /**etc/inittab** flag for a non-login port is an "o". Refer to the *Sys3 Programmer's Manual*, Volume 2B, Section 5, for a description of /**etc/inittab**.

   Edit file /**etc/inittab** to set the flag for the printer port correctly. The third information field should be an "o" instead of a "c". For example:

   2:03:o:/etc/getty tty3 b

* NEC and Spinwriter are trademarks of Nippon Electric Corporation.

2. Modify **/etc/rc.**

   The characteristics of the serial port must be set to support the speed and capabilities of the serial printer.

   a. Open the port.

      The port must be opened to set the line characteristics appropriately and must remain open for the characteristics to be maintained. Port characteristics are reset each time the line is closed.

      The port may be opened and kept open by executing the "openup" program. The port is identified as an output port (write only) with the "-w" flag followed by the path name to the desired port.

      Parameters may be added to the openup command included in /etc/rc or another openup command may be added to the file. Note that each openup command can accept only 10 parameters (devices, directories, or files).

      If you relink the port so that it can be referenced by another name, such as "lp", you may refer to the port by that name in the openup command. Below are examples:

      /etc/openup -w /dev/tty3
      /etc/openup -w /dev/lp

   b. Set port characteristics.

      Port characteristics are set with the **stty** command and/or a program.

      [1] One or more **stty** commands should be placed after the openup command in the **/etc/rc** file to set the baud rate (unless the default speed of 9600 baud is acceptable) and other port characteristics. Refer to the *Sys3 Programmer's Manual*, Volume 1A, **stty**(1) to select the characteristics you need to support your serial printer.

      The following commands support the majority of serial printers in use:

      stty sane </dev/ttyxx
      stty 1200 -echo -tabs onlcr </dev/ttyxx

      To enable flow control based on a CTS (clear to send) signal, set the icts flag instead of the ixon flag. Note that you may have to disable ixon if it has been set by a stty sane command.

      In Sys3 revisions 3.0 and later icts may be specified as a parameter to the **stty** command. Previous revisions require setting the ICTS flag in a program which must be executed after the **openup** command. A sample program is included at the end of this chapter which includes the sequence required to set ICTS.

      If you are going to use flow control based on a CTS signal, rejumper the ICP to recognize the signal on either pin 4, 11, or 20.

      (a) Shut down the system and remove the ICP board.

      (b) Locate the port's 10-pin jumper network. Refer to the figure titled, "ICP Board Option Selectors", in the *P/35 Engineering Manual.*

(c) Using the jumper placed between pin pair 3 and pin pair 4 by default, jumper pin pair 1, 2, or 3 depending on whether the CTS signal is provided to the 25 pin connector on pin 20, 11, or 4 respectively.

Additional documentation on the purpose and jumpering of each pin pair is available in the *P/35 Engineering Manual* in the section, "Configuring an ICP"

[2] A program is used to set port characteristics whenever **stty** does not provide the desired capabilities. The program may be invoked by the **/etc/rc** script immediately after the **openup** on the port or after the last **stty** referencing the port. The following example sets the ICTS flag and may be used as a guide to setting other tty flags:

```
/*  This program utilizes the "flow-control" features    */
/*  of the Plexus ICP and its' software.  It should be   */
/*  invoked by `/etc/rc` after the `openup` statement.    */

#include <errno.h>
#include <sys/param.h>
#include <sys/tty.h>
#include <sys/ioctl.h>
#include <stdio.h>
int errno,
main()
{
struct termio buf;
int fd;
char c;
fd=open("/dev/tty??",2); /* replace ?? with the port number */
if (fd<0) {
        perror("error\n");
        exit(1);
}
ioctl(fd,TCGETA,&buf);
buf.c_iflag |= ICTS;
ioctl(fd,TCSETA,&buf);
ioctl(fd,TCGETA,&buf);
printf("%o\n%o\n%o\n%o\n",buf.c_iflag,buf.c_oflag,
                buf.c_cflag,buf.c_lflag);
}
```

Chapter Eight

CONFIGURING DISKS

Configuring disks for a Plexus system involves two separate software utilities, **dconfig** and **mkfs**. **dconfig** is a Plexus utility that defines the disk space by logical disks allocated in 512-byte sectors. **mkfs** is a UNIX utility that sets up the file systems in 1K blocks which are integral to the operation of UNIX. File systems are composed of the file space itself plus pointers and file definition information which enables UNIX to find and use the files.

The two levels of software are joined by the UNIX utility, **mknod**, which specifies which logical disks and their start addresses listed in **dconfig** will be created and utilized in UNIX's device directory, /**dev**. Another UNIX utility, **mount**, attaches a specified directory hierarchy to a specified file system.

The following diagram illustrates the roles of **dconfig** and **mkfs** in organizing and utilizing disk space. Boundaries marked by double lines (=) are established by **dconfig**; single-lined boundaries illustrate the organization of file systems as established by **mkfs**.

As the diagram shows, **dconfig** establishes the *start* addresses of each logical disk, the swap address and length, and the logical device upon which the swap space resides (which in this case, is **/dev/dk1**). (**dconfig** establishes several other parameters as well which are documented in the chapter, "The Standalone Environment.")

As the diagram also shows, **mkfs** sets up the boundaries illustrated by the single lines — the internal organization of each file system — plus the length of each file system. Ordinarily, **dconfig** does not set up discrete lengths for the logical disks. It is instead handled by **mkfs** at the file system level.

Optionally, however, **dconfig** can also establish discrete upper boundaries for each logical disk, but does not do so by default except for **/dev/dk1**. Rather, all other logical disks have the same upper boundary, the end of the physical disk itself.

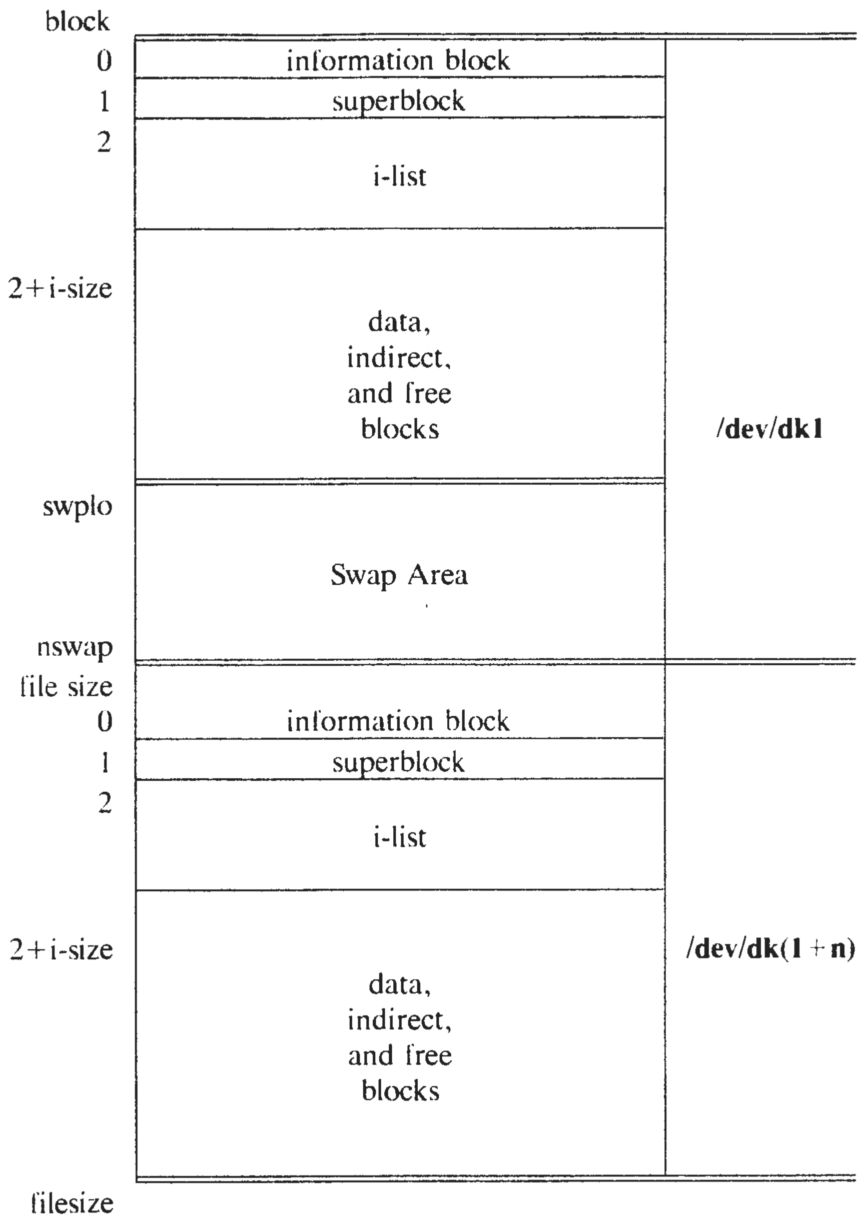

Figure 8-1. dconfig and mkfs Control Parameters

The two levels are joined by **mknod** as indicated by the syntax of this command. Below is an example:

```
mknod /dev/dk3 b 0 3
```

Use the **mknod** command to create and define a logical device in the UNIX device directory, /**dev**. In this example, the device is logical disk dk3. It is a block device ("b"), major device number 0, and has been assigned minor device number 3. You must **mknod** the logical device before you proceed to attach a file system to it via **mkfs**.

Then use the **mkfs** command to create a file system affixed to the same start address as the specified logical disk (in this case /**dev/dk3**) and specify the file system length in 1K blocks (in this example, 20000):

```
mkfs /dev/dk3 20000 m n
```

where m is the interlace factor (all P/35 disks have an interlace of 1 as of this printing) and n is a blocking factor constant which is 500.

For example, if a disk had an **m** of 7 and an **n** of 500, the disk would write to every seventh sector for 500 writes, and then return to the next to the first sector and repeat the pattern six more times until the first 3500 blocks have been used up that way. The logically contiguous sectors would occupy physical sectors 1, 8, 15, 22, . . . 3495, return to write to blocks 2, 9, 16, 23, . . . 3496, etc. until the last pass in that section would occupy sectors 7, 14, 21, 28, . . . 3500. Then the next group of logical sectors would occupy 3501, 3508, 3515, etc. and repeat the pattern until sectors 3501 through 7000 were occupied as well.

Finally, use the **mount** command to attach the file system/logical device to the directory hierarchy of your choice. For example, if you wanted **/dev/dk3** to hold your printer spooler you would enter:

```
mount /dev/dk3 /usr/spool
```

Thus, the basic sequence for configuring a disk requires you to:

a. Determine *how* you want to split up your disk(s). Each logical disk must be contained within a given physical disk; i.e., logical disks and file systems cannot span the boundaries of the actual physical disks upon which they reside.

b. Run **dconfig** to ensure that the default values are sufficient to your goals, and if not, change them, specifying the addresses and lengths in 512-byte sectors.

c. Run **mknod** to create logical disks in **/dev**. These logical devices must correspond to the start addresses you picked or established when running **dconfig**.

d. Make the file systems via **mkfs** which assigns the file systems to the logical disks you created via **mknod** and specifies the file system length in 1K blocks.

e. Create or assign UNIX directories for the logical disks/file systems you just created via **mkdir**.

f. Mount the file systems to the directories via **mount**.

This chapter provides the following procedures involved in configuring disks:

- Creating multiple or mounted file systems
- Changing the size and/or location of the swap area
- Configuring the software when installing an additional disk drive

8.1 Creating Mounted File Systems

Plexus systems are shipped with **/dev/dk1** activated, created in the device directory, established via **mkfs**, and occupied by the Sys3 root file system. That still leaves the rest of the disk(s) to be configured for your installation. Plexus has also created logical disk **/dev/dk2** in the device directory; default values in **dconfig** dictate that it extend over the rest of the disk. **/dev/dk2** does not have a usable file system until you make one with **mkfs** and **mount** it to a directory. The size of the **/dev/dk2** file system — or any file system that extends to "end of disk" — depends on the size of your disk. The following table shows the size of the default **/dev/dk2** for typical disks:

TABLE 8-1. Default Sizes (in blocks) for /dev/dk2

Disk Size	Old IMSP (pd)	New IMSP (pd)
22 Mb 8"	43	43
36 Mb 8"	13405	9405
72 Mb 8"	47473	43473
142.6 Mb 8"	116272	112272
72 Mb 14"	48085	44085
145 Mb 14"	116170	112170
289 Mb 14"	261120	257120

You may leave **/dev/dk2** as is and create a file system for it and mount it. Assuming you are already satisfied with **/dev/dk1** and the size and location of the swap area, your disk configuration task would then be finished. With smaller disks, this is usually satisfactory.

Certain circumstances, however, dictate that you further delineate the disk space between the end of dk1 and the end of the disk. Particularly, you should keep each file system at least slightly smaller than your tape backup capacity. For open-reel 9-track tape, you can archive approximately 1 Mb for every 30 feet of tape if you are using a low-overhead utility such as **dump** or **fbackup**. At this rate, you could get up to 80 Mb on a 2400-foot reel of tape. Cartridge tape drives have smaller tape capacities as follows:

System	*Capacity*
Cartridge drives shipped prior to June 1984	20MB
Cartridge drives shipped after June 1984	45MB

An advantage of smaller file systems is faster disk I/O because the operating system does not have to search through an extremely large disk area. The disadvantage is that it requires more **dump** or **fbackup** operations to backup the entire disk.

To figure the remainder of your disk you must determine exactly how long each file system can be and which dk numbers in **dconfig** provide the correct sector start addresses. This second factor is required so you know which logical disk numbers you should create in **/dev** upon which to mount file systems.

The general procedure for configuring the remainder of the disk is contained in the chapter, "Setting up Your Installation" and is not reproduced here. This section, however, takes a specific example and shows exactly how the general procedures in the other section apply to a specific case.

In our example, let us say you have a "new" (i.e. 6 MB swap) P/35 with an 8" 72 Mb disk drive and you want to create two file systems beyond **/dev/dk1** of approximately equal size. According to Table 8-1, the rest of your disk beyond **/dev/dk1** is 43473 blocks, or 86946 sectors. To subdivide **/dev/dk2** into two parts of approximately the same size and yet make use of the factory set sector start addresses, you could make **/dev/dk2** 20000 blocks (40000 sectors). That leaves a remaining space of 23473 blocks or 46946 sectors. Reproduced below are the default start sector addresses for **dconfig**'s dk numbers:

TABLE 8-2. Default Start Sector (512 byte) Addresses

		Pre-Sys3 3.2 and 22Mb P/35s	*New P/35s*
dk0	[0,0]:	**0,ˉ**	**0,ˉ**
dk1	[0,0]:	**0,40000**	**0,48000**
dk2	[0,0]:	**40000,ˉ**	**48000,ˉ**
dk3	[0,0]:	**60000,ˉ**	**68000.ˉ**
dk4	[0,0]:	**80000,ˉ**	**88000,ˉ**
dk5	[0,0]:	**100000,ˉ**	**108000,ˉ**
dk6	[0,0]:	**120000,ˉ**	**128000,ˉ**
dk7	[0,0]:	**140000,ˉ**	**148000,ˉ**
dk8	[0,0]:	**160000,ˉ**	**168000,ˉ**
dk9	[0,0]:	**180000,ˉ**	**188000,ˉ**
dk10	[0,0]:	**200000,ˉ**	**208000,ˉ**
dk11	[0,0]:	**220000,ˉ**	**228000,ˉ**
dk12	[0,0]:	**240000,ˉ**	**248000,ˉ**
dk13	[0,0]:	**260000,ˉ**	**268000,ˉ**
dk14	[0,0]:	**280000,ˉ**	**288000,ˉ**
dk15	[0,0]:	**300000,ˉ**	**308000,ˉ**

Default Start Sector (512 byte) Addresses

As you can see from this second chart, dk2's starting sector address for this system is 48000, and its default length is 43473 blocks which translates into 86946 sectors. In such case, you could assign 20 Mb (20000 blocks or 40000 sectors) to **/dev/dk2**. That means the starting sector address of the last file system would begin at sector address 88000, which is dk4. This would then mean that the third logical disk you added to **/dev** would be **dk4**, and its length would be 23473 blocks (43743 blocks minus 20000 blocks assigned to dk2) specified by running **mkfs**. The entire sequence to set up a disk this way is shown below. Further, let us say that the two logical devices created will be the user directory and the spooler.

1. Be sure you are logged in as root.

2. If you are merely altering the sizes of these files, make sure you can backup the data that is currently on these file systems first, and that they will fit into the new file systems you will be creating.

3. Change your working directory to the device directory by entering:

   cd /dev <return>

4. As shipped from Plexus, your device directory already contains dk0, dk1, and dk2. You will need to add dk4.

5. Create a logical block device for **/dev/dk4** by specifying the following:

   ```
   /etc/mknod /dev/dk4 b 0 4
   ```

   where b is a block device, 0 is the major device number, and 4 is the minor device number.

6. Create raw devices for each of the logical disk devices you created in the previous step:

   ```
   /etc/mknod /dev/rdk4 c 3 4
   ```

   where c is a "character" (raw) device, 3 is the major device number, and the last number is the minor device number (it corresponds to the rdk number).

7. Now create the file systems to correspond to these logical devices, assigning the sizes established in the introduction to this procedure.

   ```
   /etc/mkfs /dev/dk2 20000 m n
   /etc/mkfs /dev/dk4 23473 m n
   ```

   where m is the software freeblock interleave and n is 500, the blocking constant as explained in the beginning section of this chapter. You need not specify the m and n parameters with any standard P/35 disk configurations through Sys3 version 3.3, but the information is provided so you know what is actually happening when you run **mkfs**. Also, other Plexus computers require specifying the m and n factors when running **mkfs**.

8. The spooler already exists as **/usr/spool**, but you must still create the user directory:

   ```
   /etc/mkdir /user
   ```

9. Mount the logical block devices to the directories you created or chose in the last step:

   ```
   /etc/mount /dev/dk2 /usr/spool
   /etc/mount /dev/dk4 /user
   ```

10. Change your working directory to **/etc**:

    ```
    cd /etc
    ```

11. Invoke a text editor such as **vi** to open the file, **/etc/rc**. Add **/etc/umount** commands and **/etc/mount** commands for the **/dev/dk2** and **/dev/dk4** you established in the previous steps. Figure 8-2 is an illustration of the sections of **/etc/rc** you will alter. Sample additions are shown in bold.

```
# /etc/rc curstate no.-times-prev-in-curstate prevstat
TZ=PST8PDT
export TZ
(
case ${1-2} in
1)
       uname
       echo single-user
       if [ "$2" != 0 ]
       then
              : put umounts here
       /etc/umount /dev/dk2
       /etc/umount /dev/dk14
       fi
       ;;
2|7)
       if [ "$2" = 0 ] ; then
              cp /tmp/Ex* /tmp/Rx* /usr/preserve 2>/dev/null
              rm -f /tmp/* /usr/tmp/* /etc/utmp
       fi
       if [ "$2" = 0 -o "$2" = "" -o "$3" = 1 ]
       then
       echo 'initializing IMSP'
       sync
       sleep 3
       /etc/dnld -dd -a 5000000 -f /usr/lib/dnld/imsc -o /dev/im0
       /etc/devnm / | grep -v swap | grep -v root | /etc/setmnt
       : put mounts here
       /etc/umount /dev/dk2
       /etc/umount /dev/dk14
       /etc/update
       rm -f /usr/adm/acct/nite/lock*
       if [ "$1" = 2 ] ; then
#             /bin/su - adm -c /usr/lib/acct/startup
```

. . . etc.

Figure 8-2. Inserting mounts and umounts in /etc/rc

Write and quit this file.

12. It is more convenient and safer for the system to automatically perform a file system check on all active file systems when you invoke **fsck** in single-user mode. To do so, invoke a text editor such as **vi** to open the file, **/etc/checklist**. Insert the logical device names you want checked. For example, the **/etc/checklist** corresponding to the **/etc/rc** in the previous step would look like this:

```
/dev/dk1
/dev/dk2
/dev/dk4
```

13. Perform **fsck** on each of the new file systems to ensure their integrity.
14. Shut down the system gracefully (see the chapter, "Changing System Initialization States").
15. Reboot the system and bring it up multi-user as described in the section, "Booting from Disk."

8.2 Increasing the Swap Area

If you experience problems with swap area size, you should increase the size of the swap area. The following system messages are symptoms of swap area problems:

- `No more processes`
- `Sys3: panic: out of swap space`
- `Cannot fork`

To increase the swap space, follow the general procedure below. Examples of individual methods follow the general procedure.

1. Determine *how large* you want the swap area. As a general rule, the bigger the better; but you must also consider how much free space is available, and how much is required for other purposes.

   The factory setting size, beginning with Sys3 3.2, is 6 Mbytes; this is adequate for most sites. 2 Mbytes, the former default, is usually inadequate; 10 Mbytes is generous in most cases. For a more accurate accounting of your swap needs, refer to the section, "Determine Swap Space" in the chapter, "System Administration Considerations."

2. Determine *where* you want the swap space. Usually you want it on the root file system, which is **/dev/dk1** by default. But even a moderately large swap area at the end of **/dev/dk1** overlaps the starting address of the space assigned for **/dev/dk2** (see Table 5-1). Therefore, some users choose a 10 Mbyte **/dev/dk2** for the swap area.

   Note that in the course of changing the size of the swap area, you might change the logical disk boundaries. If this happens, you must back up (using **dump**) and remake (using **mkfs**) any other file systems whose boundaries have changed.

   If you will change file system boundaries, make a chart of the new boundaries, including starting and ending addresses in sectors (512 bytes), to use as an aid in the steps that follow.

3. Back up any file systems whose boundaries will change as a result of changing the swap area. Use **dump**(1M).

4. Run **dconfig** to change some or all of the following parameters:

Swapdev	The file system containing the swap area. If you want the swap area on **/dev/dk1**, and **/dev/dk1** is your root file system, you don't need to change this.
Dumpdev	Should be the same as *swapdev*.
Swplo	The starting address of the swap area. This must be in 512-byte sectors (not 1024-byte blocks).
Nswap	The number of sectors to be used for the swap area.
dk	The new logical disk boundaries.

5. Type **sync** if in single user mode.

6. Reboot the system.

7. Using **mkfs**, re-create the file systems you previously dumped to the new sizes determined by reallocating the swap area.

8. Restore those file systems from the dump tape and mount them.

8.2.1 Examples

Three approaches are typically used to increase the size of the swap area. The first extends dk1; the second uses dk2 as the swap area, and the third creates one logical disk only and places the swap area at the end of the disk. (This last approach is most useful for 36 Mbyte disks.)

Plexus recommends that you use one of the first two methods so that the logical file system configuration is as close as possible to the default file system configuration.

8.2.1.1 Extend dk1

1. Back up any file systems whose boundaries will change.
2. Enlarge the swap area to 6 Mbytes and place it at the end of dk1 by running **dconfig** with these parameters:

Rootdev	1
Pipedev	1
Dumpdev	1
Swapdev	1
Swplo	36000
Nswap	12000
dk0	0,~
dk1	0,48000
dk2	48000,~

NOTE

When you first view **dconfig**, the values in the default logical device configuration table (the table where the sizes of the dk devices are defined), are shown as zeroes. In an unaltered **dconfig** file, these zeroes signify that the default values have not been changed. If, however, you change *one* sector start address or logical disk length, **dconfig** will then assume that *all* displayed values are literal, i.e., that all other logical disks are located at offset zero and are zero length.

Therefore, if you change any start sector addresses in the default logical device configuration table you must adjust any other dk offsets and file system lengths that you intend to **mknod**, and **mount**. Otherwise these logical disks would have zero lengths.

You must also adjust other *dk* offsets and lengths to avoid having one file system overlap into the start address of another. For example, if your dk2 is presently 20 Mb and you want to keep it that size, you also plan to use dk4, and you change dk2's start address from sector offset 40000 to 48000, you would then need to also change the sector offset of dk4 from 80000 to 88000 so the 20 Mb length of dk2 did not overlap dk4.

3. Exit **dconfig**
4. Update the superblock by entering

   sync ; sync

5. Reboot the system, bringing it up multi-user.
6. Remake (**mkfs**) and restore (from a backup tape) any file systems that you altered while in **dconfig**

8.2.1.2 Use dk2 as Swap Area

1. Back up any file systems whose boundaries will change.
2. Make a 10 Mbyte swap area using dk2 by running **dconfig** with these parameters. This example works for 72 Mbyte disks or larger.

   To make a 10 Mb swap area out of dk2, key in the following values when running **dconfig**.

Rootdev	1
Pipedev	1
Dumpdev	2
Swapdev	2
Swplo	0
Nswap	20000
dk0	0,~
dk1	0,40000
dk2	40000,20000
dk3	60000,~

NOTE

BE SURE you have backed up anything that was previously in dk2. You will move this data to another available file system later.

Note that if you change any entries in the default logical device configuration table in **dconfig** (the table where the sizes of the dk devices are defined), you must change any other dk device sector offsets that are affected by the changes.

3. Exit **dconfig**.
4. Delete the device file, **/dev/swap,** and remake it (using **mknod**). Note that this step is necessary ONLY if you are placing the new swap area on a file system other than dk1. Follow these steps:

   mknod /dev/swap b 0 2

   Make sure to do the **mknod** before you shut down the system before rebooting to use the new swap area.
5. Update the superblock by entering:

   sync ; sync
6. Reboot the system.
7. Remake (via **mkfs**) **and restore (from your backup tape) any file systems affected by the device and boundary changes.**

8.2.1.3 Create a Single Logical Disk (36 Mbyte Systems)

To make a 36 Mbyte disk one file system with the swap area at the end of the disk, follow these instructions.

First do a level 0 dump to back up the entire dk1. Back up dk2, if it exists, with **tar** or **cpio**. Check the **tar** or **cpio** tape by reading it back with the "t" option.

Then run **dconfig**. Change the following parameters:

Rootdev	1
Pipedev	1
Dumpdev	1
Swapdev	1
Swplo	54800
Nswap	12000
dk0	0,-
dk1	0,66800
	BE SURE you have backed up anything that was previously in dk2. You will move this data into dk1 later.

Type "sync", reboot the system, and make a new file system at **/dev/dk1** by typing

mkfs /dev/dk1 27400

Finally, restore the dump of dk1 and the **tar** or **cpio** backup of dk2.

8.3 Configuring a Second Disk Drive

Configuring a second disk drive involves the same general steps and file modification procedures as those performed for the original system disk. In both cases, you:

1. Decide how you want to divide the physical disk into logical disks and file systems.
2. Choose the logical disks in **dconfig** that contain the sector offsets you need to divide up the disk according to your plan.
3. **mknod** the logical devices in **dev** to correspond to the sector start addresses you chose in **dconfig**.
4. **mkfs** the file systems and specify their sizes.
5. **mkdir** or choose the directories that will become file systems.
6. **mount** the chosen directories to the logical devices you created in **dev** (via **mknod**).

The main difference is that running **dconfig** is much simpler for add-on disks. Whereas setting up the original system disk required you to run **dconfig** to allocate swap space and designate the device number for the root file system and its system-intensive parameters, no such configurations are necessary for add-on disks because they add storage space and nothing else. Parameters such as *Nswap*, *Swplo*, *Swapdev*, *Dumpdev*, *Pipedev* and *Rootdev* should all be set to zero because the Sys3 does not use add-on disks for these functions.

You may not even need to run **dconfig** at all for add-on disks, because the default values will probably be sufficient to your purpose. You will, however, need to know the sector start addresses for each logical disk (dk number) so you can choose the appropriate ones to **mknod** and **mount**.

Plexus' **dconfig** is set up such that each physical disk is divided into 16 logical disks. The first one starts at sector offset zero (0) and extends to the end of the disk. The second logical disk also starts at offset zero and is 20 Mb long (except for systems shipped with Sys3 3.2 or later, in which case the factory has reset dk1 to 24 Mb on P/35s with 36 Mb-and-larger disks). Starting with the third logical disk, each of the rest of the logical disks starts at a sector offset 10 Mb (20000 sectors) higher than the previous one.

When you are in **dconfig**, regardless of the physical disk, **dconfig** always prompts for logical disks 0 through 15. As far as Sys3 is concerned, however, dk0 through dk15 reside on the first disk, dk16 through dk31 reside on the second disk, dk32 through dk47 reside on the third disk, and dk48 through dk63 reside on the fourth disk. The four disks are numbered 0 through 3. You must know the ranges of dk numbers for each of the physical disks so you can create the correct logical devices in the **dev** directory.

Standalone **dconfig** knows the four disks as pd(0,0); pd(1,0); pd(2,0); and pd(3,0). The Sys3 version, **etc/dconfig**, knows the four physical disks as **/dev/dk0**, **/dev/dk16**, **/dev/dk32**, and **/dev/dk48**.

The sector start addresses for the logical disks within each of the four possible physical disks are displayed in Table 8-3.

TABLE 8-3. Add-on Disk Start Sector Addresses

disk0	*disk1*	*disk2*	*disk3*		*Sector Offset,Length*
dk0	dk16	dk32	dk48	[0,0]:	**0,~**
dk1	dk17	dk33	dk49	[0,0]:	**0,40000**
dk2	dk18	dk34	dk50	[0,0]:	**40000,~**
dk3	dk19	dk35	dk51	[0,0]:	**60000,~**
dk4	dk20	dk36	dk52	[0,0]:	**80000,~**
dk5	dk21	dk37	dk53	[0,0]:	**100000,~**
dk6	dk22	dk38	dk54	[0,0]:	**120000,~**
dk7	dk23	dk39	dk55	[0,0]:	**140000,~**
dk8	dk24	dk40	dk56	[0,0]:	**160000,~**
dk9	dk25	dk41	dk57	[0,0]:	**180000,~**
dk10	dk26	dk42	dk58	[0,0]:	**200000,~**
dk11	dk27	dk43	dk59	[0,0]:	**220000,~**
dk12	dk28	dk44	dk60	[0,0]:	**240000,~**
dk13	dk29	dk45	dk61	[0,0]:	**260000,~**
dk14	dk30	dk46	dk62	[0,0]:	**280000,~**
dk15	dk31	dk47	dk63	[0,0]:	**300000,~**

where ~ represents the length in sectors from that offset to the end of the disk. If you alter any of these entries, you must specify the actual file length or distance to the end of the disk in lieu of the ~ symbol.

The procedure for adding a second-through-fourth disk drive is described below:

1. Determine the number of sectors you have available on the new disk. You can obtain this figure from Table 11-1 in Chapter 11, "The Standalone Environment," or from the **dformat** print-out supplied with the disk itself if you ordered it from Plexus. If neither of these means is reliable for your installation for some reason, determine the number of sectors with the following equation:

   total sectors = (((totcyl-altcyl)-2)*no.heads)*sectors per track

2. Determine how you want to allocate the space for file system(s) on the add-on disk. Two factors involved are:

   — The size of each file system you will put on the new disk

   — The specific file systems and directories you want to put on this disk

3. Examine the sector offsets in Table 8-3 and decide which dk numbers correspond to the number and size file system you wish to create or move to this disk.

   For example, if you wanted to create a series of 20 Mb (20000 1K blocks) file systems, you would choose sector offsets 40000 sectors apart (512-byte sectors).

4. Create the block and raw devices in /**dev** for each dk offset you chose in the previous step:

   mknod /dev/dkX **b 0** X
   mknod /dev/rdkX **c 3** X

   where X is the dk number you chose. Repeat this step for every dk number you are using on the new disk.

5. Make file systems for each of the logical block devices you created in the previous step. Determine the number of blocks per file system by taking the sector lengths determined in step 3 and dividing by 2.

   mkfs /dev/dkX nnnnn m **500**

   where X is the device number, nnnnn is the file size in 1K blocks, and m is the freeblock interleave factor of 1 for all P/35s through Sys3 version 3.3.

6. Ensure that the superblock is updated before performing any more steps by entering:

   sync ; sync

7. Create or choose the directories that will be associated with these file systems:

   mkdir ./dirname

   where ./ is the path to the directory and dirname is the name of the directory you are creating. For example, to create a second user directory called **/usr/user2**, enter:

   mkdir /usr/user2

8. Mount the logical disks you created to the directory names you just created or chose:

   mount /dev/dkX ./dirname

   where X is the dk number you have allocated from the add-on disk and ./dirname is a variable as explained in the previous step. For example, to mount **/dev/dk22** to **/usr/user**, enter:

   mount /dev/dk22 /usr/user2

9. Check your success by running

   df -t

   on the new file system(s) to ensure that the operating system sees the files as you created and sized them. If the system response is unsatisfactory, check every step you followed for accuracy and consistency. Especially check that you calculated the disk size properly and did not confuse sector counts with block counts.

10. If you are satisfied with the results of the previous step, add the new logical devices to the **mount** and **umount** sections of the **/etc/rc** file so they too will automatically mount and unmount when you move the system between single- and multi-user initialization states.

Chapter Nine

IMPLEMENTING SCCS

This chapter is intended to provide a short introduction to SCCS. SCCS has many more features, commands, and command options than are described here. For more information, see the Volume II document on SCCS, and the appropriate manual pages.

9.1 What SCCS Does

SCCS was originally developed as a way for individual programmers to track changes to code. It saves previous versions and generates bookkeeping information such as number of lines added, deleted, and unchanged between versions, when each revision was created, and provides the means for the user to add summary comments associated with each version. It can be used in conjunction with **make** or shell procedures to combine modules.

SCCS can also track code and documentation changes for large programming and technical writing projects, where many people may make changes to the same files. In this environment, SCCS ensures that

— Only authorized persons may write to the files in the SCCS database. When the SCCS administrator puts a file into SCCS, s/he includes a list of people authorized to write to this file. As part of its bookkeeping SCCS records who created each revision.

— Only authorized persons may add or delete files in the SCCS database. SCCS directories are owned by the SCCS administrator, so it is difficult for anyone but the SCCS administrator to make any really destructive changes to the SCCS database.

9.2 How to Use SCCS

You can use SCCS as an individual or as a member of a team. How you use SCCS commands, and how much access to them you have, depends on whether or not you are the only user.

This is a list of the most common SCCS commands and what they do.

admin	Puts files into SCCS.
get	Gets you a copy of the SCCS file. With the -e option, not available to all users, gets you the file with delta-permission, i.e., you may make official changes (deltas) to it.
delta	Puts a file back into SCCS and creates an officially changed version. Not all users have access to this command for all files.
rmdel	Undo the last delta and leave the file status exactly as it was at the most recent revision.
prs	Gives history of an SCCS file. Using **prs**, you can find out practically anything about any rev level of an SCCS file.

These commands are discussed one by one in the following sections with the versions of the commands noted that have demonstrated practicality.

9.2.1 admin

To put a file into SCCS, you can use the command

```
admin -i<filename> -a<userid> ... -y<comment> s.<filename>
```

The -i means this is a new SCCS file. Note that you put no spaces between the -i and the filename.

The '-a' means that the users whose user ids follow are permitted to make changes to this file. More specifically, these are the people allowed to use the commands **get -e** and **delta** on this file. Note no spaces between the '-a' and the user ids. You must include this field if you want to be the only one who can write to the file. If you leave it out altogether, SCCS lets anyone write to the file.

You can use the '-y' field to insert special comments. If you leave it out, SCCS supplies a stock first comment. Example A at the end of this chapter demonstrates using the '-y' field to distinguish customized from source versions of UNIX manual pages. (The ones without a comment in the '-y' field are exactly the same as the SYSTEM III source versions.)

The -y option serves the same function for **admin** that delta comments serve for later revisions, since in SCCS terms, **admin** creates the first delta.

The s. in the last field is especially important. All SCCS filenames begin with s..

Admin doesn't give any response if all goes well; it just returns your shell prompt. It will complain if your file does not contain SCCS id keyword, however. This is not fatal — your file gets **admin**'d anyway — but SCCS will continue to complain about the absence of id keywords every time you **delta** the file, so you might as well put them in the file. SCCS will not complain as long as you have added at least one line id keywords. The following three lines provide the information essential to most applications.

```
'\" Revision Level %I%
'\" Last Delta     %G% %U%
'\" File Name       %M%
```

If you put these at the beginning of each file, when the file is gotten (with **get**) read-only, the id keywords are interpolated with the appropriate information. This way users can tell the revision level of software or documentation produced on your computer. The manual page on **get** (not **admin**) gives the complete list of id keywords.

When you **admin** a file, you create revision 1.1 by default. Subsequent revisions are 1.2, 1.3, 1.4, etc., and you can also bump the major revision level number with an option to **get**.

The format of the 's.' file created by **admin** is described in *sccsfile*(5). As an SCCS user, you may never need to know much about the internal structure of the 's.' file. You should realize, however, that the 's.' file is where all the history is kept--all the bookkeeping information as well as all the previous versions of the file. If it gets destroyed, you lose all the SCCS information. This is why 's.' files are so well protected: they are automatically made mode 444, and in large projects, the directories in which they live are owned by the SCCS administrator, not by users.

Admin can also be used to give a new user permission to make deltas, or remove or renumber previous deltas, and many more abilities beyond the scope of this chapter. As you might expect, **admin** is used mostly by the SCCS administrator in large projects.

Like all SCCS commands, **admin** can be used in shell procedures, like the one in Example A. That one **admins** all the files in one section of the Volume I *UNIX Programmer's Manual*.

9.2.2 get

To read an SCCS file, use the command

<path>/s.<filename>

This gets you the most current version of the file read only (mode 444). Sometimes you just want to know what's there; no need to get it with more powerful permissions.

To **get** a specific version read only, use

get -r<rev level> <path>/s.<filename>

You can use this for verification of changes and when executing makefiles to put manuals together. (The makefile in Example D executes a series of **get -r** commands. The variables R_<FILENAME> stand for the revision levels defined in the revs file. This is discussed more fully in the "Setting Up for Many Users" section of this chapter.)

To read and write an SCCS file, but NOT with the intention of creating an official delta, use

get -k <path>/s.<filename>

This is much more useful than it may seem. For example, if you want to create a new file that is almost identical to the SCCS file, **get -k** gives you the file with permission mode 644, but does not create a lock file (see below).

To read and write with the intention of creating an official revision, use

get -e <path>/s.<filename>

Not all users are allowed to use **get -e** on every file: only the ones named in the original **admin** (or subsequently added).

The file you receive with any **get** has the 's.' prefix removed; it is the result of SCCS processing of the 's.' file. All **gets** leave the file 's.<filename>' in the SCCS directory, but **get** with -e has another effect: it creates an additional file called 'p.<filename>'. This file is a lock file; once you get the file with '-e', 'p.filename' is created and no one else can **get** the file with '-e'. (They can, however, **get** the file read only, or with '-k'.) The existence of 'p.filename' is useful to remind you which files you, or other users, still have out under **get** with '-e' when it comes time to put all the documents together. It is also the best way to find out, if many people can **get** the file with '-e', who actually has gotten it. To find out, just read it.

The 'p.file' goes away when you **delta** back the file you obtained with **get -e**.

In team projects, users cannot execute **gets** from within the SCCS directory, since this directory is owned by the SCCS administrator. If you create SCCS files for your own personal use, your working directory makes no difference.

Get has other flavors, but these are the ones used most of the time in most applications.

9.2.3 delta

If you use SCCS for your files alone, you automatically have permission to make deltas to them. For large projects, not everyone can **delta** — you must have delta-permission, which is bestowed by the SCCS administrator. The standard **delta** command is

delta <path>/s.<filename>

SCCS responds

```
Comments?
```

and you type whatever you like. On large projects there is usually a standard format for delta comments, which includes programmer's name, the revision level of the software for which the delta was made, and what was changed in the file. These comments are saved along with the file, and can be displayed with **prs** or **prt**. SCCS then reports the number of lines added, deleted, and unchanged between this revision and the last and gives the new revision number. The file you **delta** is also removed automatically from your directory.

Other options to **delta** allow you to keep a copy of the file you **delta**, suppress **delta**'s responses, give the Modification Request (such as an engineering change order) number for which this **delta** was made, give your comment on the command line, and automatically **diff**, on standard output, the file you **delta** with the previous version.

9.2.4 rmdel

This command removes revisions of SCCS files. It has important restrictions on use, however. First, you can only remove a delta that you created; and second, you can only remove the last (i.e., most recent) one. Standard syntax is

> **rmdel -r<rev level> <path>/s.<filename>**

When lots of delta-makers are involved, the use of **rmdel** is sometimes restricted to the SCCS administrator.

9.2.5 prs

Read the Volume I account of **prs**; it contains examples of how to construct **prs** requests. The sample shell **prt** in Example B does a **prs**; sample output is shown in Example C.

Prs has options to list information by time parameters, e.g., before or after a specified revision. You can use **prs** with these options to verify files like Example E, which illustrates keeping the current revision levels of all the files for a specific manual. This way, you can double-check that no deltas were made that were not recorded in the revs file. You can also use it as a source for writing summaries of the changes to the files since the last revision of a manual. **Prs** takes directory names as arguments, giving you history information for all files within a directory, so you can get much information with one command.

9.3 Updating an SCCS File

This is a typical command sequence for updating an SCCS file. It is not very complicated; most of the time, this is what using SCCS amounts to.

1. Be sure you are in a directory you can write to. If the file you want to change is in a directory owned by the SCCS administrator, you cannot write to it while you are in that directory.

2. Get the file with **get -e** by entering:

   > **get -e s.**filename

   Remember that the name of the copy that goes to your working directory does not have the 's.' prefix.

3. Edit the file.

4. When you are finished, put the file back into SCCS using **delta**.

   > **delta <path>/s.<filename>**

   SCCS responds

   ```
   Comments?
   ```

   and you type whatever you like.

5. If you are using makefiles and revs files as discussed in the next section, update the revs file.

9.4 SCCS and Makefiles

SCCS can be used in conjunction with **make** to create large multi-file manuals. This same procedure can also be used, after each module has had its changes made, for combining program modules.

It's handy to have one file containing the makefile, and a separate one containing the list of revision levels you want for a particular **make**. You may want to keep these files under SCCS control, too, because they are tedious to recreate. If you have used **make**, the procedure is pretty self-explanatory. You can substitute the tools and build procedures appropriate for your task (e.g., compiles). Samples of the makefiles and revs files are included in Example D and Example E.

To put together a manual composed of many separate files, all under SCCS control, and assuming you want the latest revision of each file, do the following:

1. Make sure no relevant files are still being delta'd. Perform **ls** on the appropriate directories for 'p.' files. If you find any still being changed, read the 'p.' file to see who is changing it; find out whether the change is important enough to delay the build.
2. Be sure the revs file reflects the current highest rev level of all relevant files. Verify this with **prs** or **prt** if you're not sure.
3. It's handy to do the make in a directory reserved just for doing makes. Create one if you don't have one.
4. Move or copy the makefile and revs file to the special directory. The directory should now contain these two files only.
5. From within the make directory, issue the make command

   make -f makefile_name

6. The makefiles for the manual sections first do a series of **get -r** commands. They get the value of 'r' from the revs file that is included at the beginning of the makefile. This procedure assumes the revs file is current; i.e., all the levels listed are the most recent ones. (To create a manual that mixes current and down-level versions of specific pages, just make sure the revs file you use lists the levels you want, rather than all the current ones.) Then you see the typical **get** responses as each file is gotten: the revision level and the number of lines in the file are listed. The files are then concatenated and printed.

   This makefile also allows you to put its result, the manual section, into a directory ($T) other than the one in which you did the **make**. If you want to do this, define $T as something other than '.'.

7. Clean up the directory when you're done. Use **rm -f** because the files are all mode 444.

9.5 Preparing to Make a New Online Manual

Because all the manual files are in SCCS format, you can't just copy the 's.' files into **/usr/man**; they contain many lines that are unrecognizable to the program that formats the man pages for the screen. Therefore, you must **get** (no options) the SCCS files before putting them into **/usr/man**. This does what you want:

1. It fills in the SCCS id keywords, so anyone reading the raw **/usr/man** file can see the rev level and date of last delta.
2. It makes the file mode 444, so the master customer copy can't be written to.
3. It strips the 's.' prefix, so the filenames look exactly like the old ones.

9.6 Setting Up for Many Users

The SCCS document, "Function and Use of an SCCS Interface Program" in the *Unix Programmer's Manual, vol2B*, describes one way to set up SCCS systems for many users.

This document points out that for a given project whose documents will be controlled and held in an SCCS directory you should assign an SCCS administrator who will own all the files and the SCCS interface program. It is best that this administrator have a login name unique to the project; that is, the ownership of the files and interface program should not be root or the login name of any of the project participants. The reason for this is to keep SCCS administrative and update work separate, and to keep a project member's usual login (or root) from inadvertently removing SCCS files. This can be accomplished in one of at least two ways:

1. Assign SCCS administration to a person not involved in the project. This person's only involvement in the project would be to maintain the files and alter project members' privileges such as **get** and **delta** according to the project's needs.
2. Assign SCCS administration to a member of the project, but create a fictitious alternate login name for that person. Give ownership of the SCCS files to the fictitious name. This keeps the project member's role as SCCS administrator (maintenance of the SCCS'd files) separate from the role as an active project member (updating of the files).

Then, following the guidelines in the "Function and Use of an SCCS Interface Program" in the *Unix Programmer's Manual, vol2B*, the SCCS administrator could write or copy a simple C program such as **inter.c** shown in Example F, which executes the command on the command line.

NOTE

> Be sure to follow either the syntax in Example F at the end of this chapter or the *description* of how to write an interface program in the "Function and Use of an SCCS Interface Program" in the *Unix Programmer's Manual, vol2B*, rather than the sample interface program listed there. That sample interface program contains errors.

In Example F, the set-userid bit has been set so the object file, **inter**, looks like it's always being executed by the SCCS administrator, who in this case is sandy1. The sample program then links **inter** with the sensitive commands, **get**, **delta**, and **rmdel**. Users must have **/user/sandy1** in their command paths before **/usr/bin**, where **get**, **delta**, and **rmdel** usually reside. SCCS takes care of the rest.

With the program illustrated in Example F, any user can do **get** with no options, or with '-r', or '-k'; SCCS makes sure only the users designated by the SCCS administrator have access to 'get -e', **delta**, and **rmdel**. When a user tries to execute a **get -e** or **delta** or **rmdel**, SCCS reads the permission list associated with the file. Then if the user is authorized, SCCS executes the command in **/user/sandy1**, (in the user's path ahead of the one in **/usr/bin**), linked with **inter**, which makes the command look like it came from sandy1.

If you were to link inter with **admin** as well, all project members would have **admin** privileges. Whenever someone **admin**s a new file, however, it involves putting new entries into the appropriate makefile and revsfile. This extended privilege could defeat the purpose of having an SCCS administrator.

You may elect to achieve the same effects in other, perhaps simpler, ways such as:

a. Copying and renaming **get** and the other sensitive SCCS commands,

b. Fixing the set-userid bit,

c. Putting the renamed commands in a special directory,

d. Instructing users to put this directory in their path.

EXAMPLE A

This is the first part of a shell procedure that **admin**s an entire directory at once.

```
admin -ia.out.5 -asandy -agreg -amike -acraig -y"Changed pdp-11 and vax
refs to z8000 and 68000; changed base for calcs from octal to hex;
68000 stuff all commented out" s.a.out.5; rm a.out.5
admin -iacct.5 -asandy -agreg -amike -acraig -y s.acct.5; rm acct.5
admin -iar.5 -asandy -agreg -amike -acraig -y"Changed pdp-11 and vax
to z8000 and 68000"  s.ar.5; rm ar.5
admin -ichecklist.5 -asandy -agreg -amike -acraig -y s.checklist.5;
rm checklist.5
admin -icore.5 -asandy -agreg -amike -acraig -y"Removed ref to
sdb" s.core.5; rm core.5
admin -icpio.5 -asandy -agreg -amike -acraig -y s.cpio.5; rm cpio.5
admin -idir.5 -asandy -agreg -amike -acraig -y s.dir.5; rm dir.5
admin -idump.5 -asandy -agreg -amike -acraig -y"Changed block
size from 512 to 1024 bytes" s.dump.5; rm dump.5
admin -ierrfile.5 -asandy -agreg -amike -acraig -y"Put
in correct #defines" s.errfile.5; rm errfile.5
admin -ifilesystem.5 -asandy -agreg -amike -acraig -y s.filesystem.5;
rm filesystem.5
admin -ifs.5 -asandy -agreg -amike -acraig -y"Changed block
size from 512 to 1024 bytes; changed how to calculate
where inodes are" s.fs.5; rm fs.5
admin -ifspec.5 -asandy -agreg -amike -acraig -y s.fspec.5; rm fspec.5
admin -igps.5 -asandy -agreg -amike -acraig -y s.gps.5; rm gps.5
admin -igroup.5 -asandy -agreg -amike -acraig -y s.group.5; rm group.5
admin -iholidays.5 -asandy -agreg -amike -acraig -y"Plexus
addition" s.holidays.5; rm holidays.5
admin -iinittab.5 -asandy -agreg -amike -acraig -y"Usage
notes for init" s.inittab.5; rm inittab.5
admin -iinode.5 -asandy -agreg -amike -acraig -y s.inode.5; rm inode.5
admin -iintro.5 -asandy -agreg -amike -acraig -y"Mentioned Plexus
additions and deletions" s.intro.5; rm intro.5
```

EXAMPLE B

```
# This is the shell procedure that corresponds to the
# command prt.  Prt prints history information about SCCS
# files in the format specified here.

prs -d"OF:Rev :I:  Delta made on :Dm:/:Dd:/:Dy: :T: by :P:
:DL:OC:" -e -l $1
```

EXAMPLE C

This is output of **prt**.

```
s.fsck.1m                                         Rev 1.6  Delta made on
11/10/82  15:27:02  by  sandy  00016/00005/00266  added  note  that
Plexus provides stand version, and that this version supports at
most five args.

s.fsck.1m                                         Rev 1.5  Delta made on
10/27/82 13:12:08 by sandy 00001/00001/00270 Changed id keywords

s.fsck.1m                                         Rev 1.4  Delta made on
10/25/82  12:11:12  by  sandy  00009/00004/00262  sandy--delete 1st 2
options  for  -sX;  add  bug  that  too  quick  rebooting  after  mesg
invalidates corrections.

s.fsck.1m                                         Rev 1.3  Delta made on
10/19/82  16:49:32  by  sandy  00003/00000/00263  sandy  for  sys3  1.1:
added bug notice about fifos.

s.fsck.1m                                         Rev 1.2  Delta made on
10/18/82 17:11:10 by sandy 00003/00000/00260 ID keywords

s.fsck.1m                                         Rev 1.1  Delta made on
10/18/82   14:21:16   by   sandy   00260/00000/00000   Added   ref   to
appropriate doc
```

EXAMPLE D

This is a sample makefile for Volume 1, Section 5.

```
# Revision Level 1.1
# Last Delta     16:51:13 11/3/82
# File Name      makeman5

#include revsman5

# S = SCCS directory
# C = Source directory
# T = Target directory

S = /user/sandy/man5
C = .
T = .

# Tools required
CAT    = cat
NROFF  = nroff -man
TROFF  = troff -man

# The source file names. Order is the order of the manual.
MAN_PAGES        = \
$C/intro.5         \
$C/a.out.5         \
$C/acct.5          \
$C/ar.5            \
$C/checklist.5     \
$C/core.5          \
$C/cpio.5          \
$C/dir.5           \
$C/dump.5          \
$C/errfile.5       \
$C/fs.5            \
$C/fspec.5         \
$C/gps.5           \
$C/group.5         \
$C/holidays.5      \
$C/inittab.5       \
$C/inode.5         \
$C/mnttab.5        \
$C/passwd.5        \
$C/plot.5          \
$C/pnch.5          \
$C/profile.5       \
$C/sccsfile.5      \
$C/termcap.5       \
$C/tp.5            \
$C/utmp.5
```

```
# Build all
all : $T/man5
$T/man5  :  $(MAN_PAGES); \
$(CAT)  $(MAN_PAGES) | $(NROFF) > man5

# How to make each page.
$C/intro.5     :$S/s.intro.5     ;cd $C; get $(R_INTRO_5)    $S/s.intro.5
$C/a.out.5     :$S/s.a.out.5     ;cd $C; get $(R_A_OUT_5)    $S/s.a.out.5
$C/acct.5      :$S/s.acct.5      ;cd $C; get $(R_ACCT_5)     $S/s.acct.5
$C/ar.5        :$S/s.ar.5        ;cd $C; get $(R_AR_5)       $S/s.ar.5
$C/checklist.5 :$S/s.checklist.5 ;cd $C; get $(R_CHECKLIST_5) $S/s.checklist.5
$C/core.5      :$S/s.core.5      ;cd $C; get $(R_CORE_5)     $S/s.core.5
$C/cpio.5      :$S/s.cpio.5      ;cd $C; get $(R_CPIO_5)     $S/s.cpio.5
$C/dir.5       :$S/s.dir.5       ;cd $C; get $(R_DIR_5)      $S/s.dir.5
$C/dump.5      :$S/s.dump.5      ;cd $C; get $(R_DUMP_5)     $S/s.dump.5
$C/errfile.5   :$S/s.errfile.5   ;cd $C; get $(R_ERRFILE_5)  $S/s.errfile.5
$C/fs.5        :$S/s.fs.5        ;cd $C; get $(R_FS_5)       $S/s.fs.5
$C/fspec.5     :$S/s.fspec.5     ;cd $C; get $(R_FSPEC_5)    $S/s.fspec.5
$C/gps.5       :$S/s.gps.5       ;cd $C; get $(R_GPS_5)      $S/s.gps.5
$C/group.5     :$S/s.group.5     ;cd $C; get $(R_GROUP_5)    $S/s.group.5
$C/holidays.5  :$S/s.holidays.5  ;cd $C; get $(R_HOLIDAYS_5) $S/s.holidays.5
$C/inittab.5   :$S/s.inittab.5   ;cd $C; get $(R_INITTAB_5)  $S/s.inittab.5
$C/inode.5     :$S/s.inode.5     ;cd $C; get $(R_INODE_5)    $S/s.inode.5
$C/mnttab.5    :$S/s.mnttab.5    ;cd $C; get $(R_MNTTAB_5)   $S/s.mnttab.5
$C/passwd.5    :$S/s.passwd.5    ;cd $C; get $(R_PASSWD_5)   $S/s.passwd.5
$C/plot.5      :$S/s.plot.5      ;cd $C; get $(R_PLOT_5)     $S/s.plot.5
$C/pnch.5      :$S/s.pnch.5      ;cd $C; get $(R_PNCH_5)     $S/s.pnch.5
$C/profile.5   :$S/s.profile.5   ;cd $C; get $(R_PROFILE_5)  $S/s.profile.5
$C/sccsfile.5  :$S/s.sccsfile.5  ;cd $C; get $(R_SCCSFILE_5) $S/s.sccsfile.5
$C/termcap.5   :$S/s.termcap.5   ;cd $C; get $(R_TERMCAP_5)  $S/s.termcap.5
$C/tp.5        :$S/s.tp.5        ;cd $C; get $(R_TP_5)       $S/s.tp.5
$C/utmp.5      :$S/s.utmp.5      ;cd $C; get $(R_UTMP_5)     $S/s.utmp.5
```

EXAMPLE E

This is a sample revs file.

```
# Revision Level 1.1
# Last Delta     16:52:01 11/3/82
# File Name      revsman5
# Rev levels of man5 source.

R_INTRO_5        = -r1.2 # s.intro.5
R_A_OUT_5        = -r1.2 # s.a.out.5
R_ACCT_5         = -r1.2 # s.acct.5
R_AR_5           = -r1.2 # s.ar.5
R_CHECKLIST_5    = -r1.2 # s.checklist.5
R_CORE_5         = -r1.2 # s.core.5
R_CPIO_5         = -r1.2 # s.cpio.5
R_DIR_5          = -r1.2 # s.dir.5
R_DUMP_5         = -r1.2 # s.dump.5
R_ERRFILE_5      = -r1.2 # s.errfile.5
R_FS_5           = -r1.2 # s.fs.5
R_FSPEC_5        = -r1.2 # s.fspec.5
R_GPS_5          = -r1.2 # s.gps.5
R_GROUP_5        = -r1.2 # s.group.5
R_HOLIDAYS_5     = -r1.2 # s.holidays.5
R_INITTAB_5      = -r1.2 # s.inittab.5
R_INODE_5        = -r1.2 # s.inode.5
R_MNTTAB_5       = -r1.2 # s.mnttab.5
R_PASSWD_5       = -r1.2 # s.passwd.5
R_PLOT_5         = -r1.2 # s.plot.5
R_PNCH_5         = -r1.2 # s.pnch.5
R_PROFILE_5      = -r1.2 # s.profile.5
R_SCCSFILE_5     = -r1.2 # s.sccsfile.5
R_TERMCAP_5      = -r1.3 # s.termcap.5
R_TP_5           = -r1.2 # s.tp.5
R_UTMP_5         = -r1.2 # s.utmp.5
```

EXAMPLE F

```
/*
        This is the SCCS interface program, which makes possible
        the use of SCCS commands on files in the SCCS administrator's
        directories by other users.
*/

#include <stdio.h>

#define LENGTH 100

main(argc, argv)
int argc;
char *argv[];
{
     register int i;
     char cmdstr[LENGTH];

/*
     Invoke SCCS command, passing args.
*/
     sprintf(cmdstr, "/usr/bin/%s", argv[0]);
     execv(cmdstr, argv);
}
```

Chapter Ten

IMPLEMENTING UUCP

Chances are if you can phone someone at another UNIX site, your computers can also talk to each other using the UNIX utility **uucp**. **uucp** allows UNIX systems to communicate with each other over phone lines or hardwired communication lines. Using **uucp**, you can transfer files from one UNIX machine to another, send electronic mail, issue UNIX commands from your terminal to be executed on another UNIX machine, send or receive software or patches, and perform many other general communications tasks. There is even an informal UUCP Network that links several thousand UNIX sites.

Note that **uucp** works as a *batch* operation. This means that jobs can accumulate and then all get done at one time. **Uucp** is primarily for queueing and executing remote jobs, i.e., jobs on UNIX systems different from your home system. A *job* might be a file transfer, copy, mail, list, etc.; most UNIX commands are capable of being executed remotely, though in practice, sites seldom allow remote users to do much. The jobs awaiting execution are kept in a *spool* directory; on Plexus systems, this is **/usr/spool/uucp**. Then, when your computers call each other, they search the spool directory to see if there is any work to do. If there is work, they begin to perform the jobs.

The setup effort involved for **uucp** is minimal: all that's required is a bit of customization of a couple of files, a modem or hardwire connection, and someone to talk to. This chapter provides the general procedure to get **uucp** working for you. For more information, read the standard UNIX documentation on **uucp**. Your **uucp** neighbors may also be helpful.

Note that the procedures listed here are a subset of those in the document UUCP Implementation in the *Plexus Sys3 UNIX Programmer's Manual -- vol 2B*. Some of the setup procedures described in that document involve modifying UNIX source files such as **uucp.h** and **makefile**, and recompiling source. Plexus' license with Bell Laboratories does not permit us to distribute UNIX sources to customers, so Plexus has done this portion of the **uucp** setup for you in advance. In particular, you can assume that all the **uucp** programs are in the right directories with the right permissions attached to them, have reasonable defaults, and are ready to run. You just have to supply information particular to your site, e.g., who you want to talk to, and what port(s) you will use.

You must decide a few issues and obtain various pieces of information before you can begin the setup.

1. Will your link be over phone lines or direct?
2. Do you want to call only, be called only, or both?
3. Who will you contact?
4. What sorts of tasks do you want to be able to do on the remote system?
5. What sorts of tasks do you want remote users to be able to do on your system?

The next few introductory sections deal with these questions.

10.1 To Call or Not to Call?

With **uucp**, as with most computer communications protocols, a distinction is made between the caller and the callee. If you call the other site, your site is "master"; if they call you, you are "slave". This distinction is important because it determines the type and, to a lesser extent, the amount of setup work you need to do. So you should first determine whether you can connect with another site on an "on-demand" basis (i.e, both call *and* be called), or whether you will always call *or* be called. How do you decide?

This sort of decision is often made on the basis of what hardware is available to you. The general rules are: if you can be called, you must make a login port and modem available to **uucp**; if you can call, a non-login port and modem (preferably autodial) must be available. If you can both call and be called, you need two ports and two modems. The same rules apply to any sites you might want to link to.

So if you don't have an autodial modem but want to establish contact regularly and automatically (e.g., to transmit electronic mail), you'd probably want always to be called by your **uucp** neighbors. You'd take the slave role. If you have an autodial modem but only one available port, you might elect to be master.

Many users find the most convenient way to set up **uucp** connections is on an "on-demand" basis; that is, each partner is capable of both calling and being called, and calls are made only when one or the other site has work. In this case, both sites must equip themselves to play either role.

The minimum hardware requirements are just a plain modem and dialup port on both sides. Note that even if you and your potential neighbor both have only this minimum, you can still converse over **uucp**. You just have to dial each other manually. In this case, your connection would be "direct", as far as **uucp** is concerned, since you do not use an Automatic Call Unit (ACU).

Many installations with several CPUs on site link them directly *via* cables rather than over phone lines. If you are going to set up direct links, you still need a login port for each remote machine, if they call you; and a dialout non-login port, if you call them; and two ports per remote machine if you both call and are called.

If you run the Plexus Network Operating System (NOS) to link several local CPUs, you may link all the local nodes with virtual ports. Connecting to remote sites -- those not on the NOS net -- involves the same hardware requirements listed above.

So first figure out what kind of hardware you have available for **uucp**, whether you want calls to be made automatically, and how many free ports you have. Decide what your options are; your potential neighbor's requirements may also influence this decision, since one or the other of you may have more flexibility.

10.2 Who Will Be Your Neighbor?

Who will you contact? You can in theory contact any UNIX SYSTEM III or Version 7 site that also has the **uucp** package. But some sites are going to be especially important to you; these are your **uucp** "neighbors". These are sites you are called by or call. They in turn have other neighbors, and through the network of neighbors, you can actually communicate with many more systems than you call, because messages are piggy-backed as they are carried from one site to the next.

Once you've got a site in mind, get in touch with them to inquire if they would like to be your **uucp** "neighbor". People usually agree readily if you just want to exchange mail and an occasional file; it can be more difficult to find a neighbor if you also wish to exchange Usenet news.* If they agree, then exchange system names, phone numbers, login ids, and passwords as necessary (see below). The second half of this document (section 2.0) discusses in detail how to make the necessary modifications to system files, once you have this information in hand.

You should still be able to communicate via **uucp** with sites that are unwilling to be your immediate neighbors; you can probably reach them through a shared neighbor, for example.

Your system name (the "nodename") is how your system will be known to the **uucp** world. It can be found by issuing the **uname** command with the "-n" option. (**Uname** alone -- without "-n" -- may not work, if your system name and nodename differ.) If you don't have a nodename, you can assign one by running the program **dconfig**. See the appropriate Plexus *User's Manual* for your system for complete information on running **dconfig**.

The phone number you give the other site is the phone number of your modem; the login id and password are what *they* will use on *your* system. They in turn may give you their system nodename, phone number, and a login id and password for you to use on their system.

Less exchange of information is necessary if one or the other party is always going to be master or slave. If you are always going to be master, you need all four pieces -- system name, phone number, login id and password for your neighbor's system -- for every site you call; but your neighbors need only your system nodename. They already know their own phone number, and they know your login id and password because they gave them to you. They don't need your phone number or a login id or password for your system since they never login to your system. So masters need the following:

* Usenet **news** is not a Plexus product, and Plexus does not support it.

Master Needs	**Slave Needs**
Slave system nodename	Master system nodename
Slave phone number	
A login id on slave system	
A password on slave system	

Conversely, if you are always going to be slave, your neighbor sites each need your nodename, phone number, a login id and password on your system, but you need only the nodenames of your neighbor systems. You don't need login ids or passwords for their systems because you never login to other systems.

If both sites are going to call and be called ("on-demand"), then you both need all the information about each other. The following table illustrates the information that needs to be exchanged for two-way "on-demand" calling.

You Provide	**They Provide**
Your system nodename	Their system nodename
Your phone number	Their phone number
A login id on your system	A login id on their system
A password on your system	A password on their system

Before contacting potential neighbors, make a list of the information you think you will need and the information you will have to provide.

10.3 What Can Be Done?

During the **uucp** transaction, the master site first does everything it needs to do on the remote system, and then the asks the slave site if it has work. If the slave site has work, then it begins to execute its jobs on the master's system. Both master and slave may set limits on what can be done by the other; few **uucp** users allow their neighbors to have unlimited access to their machines. In fact, the set of commands allowed to neighbors is usually quite restricted; furthermore, all the file protection mechanisms of your system apply within **uucp** transactions. You can specify what directories will be accessible to remote and also local users of your system; this is what the file **/usr/lib/uucp/USERFILE** is for. Changing this file is discussed at length in section 2.2.6 of this document. Often systems make the directory **/usr/spool/uucppublic** the only login directory for remote users.

So now let's suppose you know what role you will assume in the exchange (master or slave or both), and you have found a neighbor, and have the necessary calling and login information in hand. You also have an idea how much you want your neighbor to be able to do on your system. What do you do now?

10.4 Procedures

This section tells what you need to do to your system to set up a **uucp** link with a neighbor site. All these steps must be done by root.

10.4.1 Setup for Autodialing

You must do three things to set up for autodialing:

1. Verify that you can use the ACU **/usr/plx/dial** (Racal Vadic 3451 modems only). Otherwise, prepare a different ACU.
2. Disable logins on the logical **uucp** port.
3. If you are autodialing over phone lines, make sure the logical device file **/dev/ttyX** is owned by **uucp**.

If you are not going to use autodialing, skip this section and proceed directly to 2.2.

10.4.1.1 Prepare the ACU

Plexus provides the autodialing program **/usr/plx/dial**, which works for Racal Vadic 3451 modems. This program is an Automatic Call Unit (ACU).

If you have a different kind of modem, you must create a file that will serve as ACU in place of **/usr/plx/dial**. This file should be named instead of **/usr/plx/dial** in all places that specify the ACU, e.g. the file **/usr/lib/uucp/L-devices** (see below). The **uucp** system sends the following four arguments to the dialing program.

Argument 0	Program in the L-devices file (your program name).
Argument 1	Modem TTY line.
Argument 2	Baud rate of line.
Argument 3	Phone number.

Your program should do the right things with these arguments, and return 0 if successful, non-zero if unsuccessful. It should have permissions of 775.

We'll assume for the sake of simplicity in the rest of this explanation that your ACU is **/usr/plx/dial**; but remember that if yours is different, you must use yours instead.

10.4.1.2 Disable Logins on the Port

To use autodial, the port to which the autodial modem is connected must not be a login port. The file **/etc/inittab** controls whether or not ports are login ports, so you must change this file. Find the line in **/etc/inittab** that refers to the port you have decided to use for **uucp**; suppose it is port 6. It should look like this:

```
2:06:c:/etc/getty tty6 b
```

Change it so it looks like this:

```
2:06:o:/etc/getty tty6 3
```

The "o" disables logins on this port.

10.4.1.3 Change Owner of Device

Finally, if you will be dialing out over phone lines (as opposed to direct lines), verify that **uucp** owns the logical device file **/dev/ttyX** you use for dialing. **Uucp** needs to own the file so it can restrict access to it when **uucp** is in use. Suppose the device is tty6; if it is not owned by **uucp**, issue the command

```
chown uucp /dev/tty6
```

10.4.2 UUCP Implementation

To implement **uucp**, you must do the following steps:

1. Verify that your ICP is jumpered correctly for use with a modem. Prepare cable if necessary.
2. Verify the system nodename.
3. Verify that the **/etc/passwd** entry for **uucp** has **/usr/lib/uucp/uucico** as its shell.
4. If your system will make calls, modify **/usr/lib/uucp/L.sys**.
5. Modify **/usr/lib/uucp/L-devices**.
6. Modify **/usr/lib/uucp/USERFILE**.
7. If your system will call other systems automatically, write the shell procedure to do the calling, and modify **/usr/lib/crontab** to execute the shell procedure automatically.

As we shall see, ordinary users can modify some **uucp** files, e.g., they can remove the **/usr/spool/uucp/STST*** files. But all the setup of **uucp** files described here must be done by root.

10.4.2.1 Hardware Preparation

The Plexus ICP pinouts are a "null modem". That is, like most systems, the ICP sends data out as if it were a modem already; that is, it does what modems do. (For example, it reverses transmit and receive.) Since UUCP non-direct connections require a modem, and since the modem will continue to perform the same functions already performed in the ICP, certain signals from the ICP must be reversed. The cable from the ICP (TTY port) to the modem must perform this reversal. The following cabling instructions make the modem work properly for modems that will autodial:

Plexus Pin Number	to	Modem Pin Number
2		3
3		2
4		5
5		4
6		20
7		7
8		8
20		6

Do not connect clock pins; they are for synchronous use only.

If you wish your modem to be autoanswer only, you may connect only pins 2, 3, and 7 as shown above. This way, the port may be a regular login port when it is not in use by the remote site.

Modems that will autodial also require the ICP to be specially jumpered. See the ICP Board Option Selectors figure in the of your *Plexus Engineering Manual* for the locations of the relevant jumper blocks for the various ports on the ICP. Note that ports 5 through 7 come to the left of port 4, which is to the left of ports 0 through 3. Normally you will jumper pin pair 3 (for Clear To Send) and pin pair 4 (for Carrier Detect). See the *Plexus Engineering Manual* chapter, "Configuring the ICP" for alternate configurations. Do not jumper the ICP for an external clock.

Usually you will not have to worry about the hardware or cabling on the system being talked to by your Plexus computer; that is your neighbor's problem. Your neighbor may or may not have a Plexus system. But sometimes a site will have several computers of different types running UNIX. If you happen to be responsible for another system in addition to the Plexus, and you must connect the two machines by modems (not direct, wire-to-wire connections), then you must first determine whether the other computer also behaves like a null modem, and make cable accordingly. If it does behave like a null modem, you must swap the signals not only when they come out of the Plexus, but again when they enter the other computer.

Direct, wire-to-wire connections require cable just like terminals, i.e., pins 2, 3, and 7 must be connected.

10.4.2.2 Verify Nodename

Issue the command

```
uname -n
```

to get the nodename of your system. Your neighbors, as well as other users of **uucp**, know your system by this name, and include it in the **uucp** pathnames of any commands to your system. For example, if your network nodename were plx, a **uucp** neighbor sending mail to userid chris on this node would use the command

```
mail plx!chris
```

A more remote **uucp** user (non-neighbor) could reach chris on plx through a more complicated expression, possibly including several nodenames separated by exclamation points (!).

```
mail node3!node2!node1!plx!chris
```

Csh users must remember to escape the !, preceding it with a backslash (\) since it has special meaning in the C-shell.

If you don't have a nodename, run the program **dconfig** to give yourself one. See the chapter, "The Standalone Environment", on how to run **dconfig**.

10.4.2.3 Modify /etc/passwd

The file **/etc/passwd** should include a login for **uucp**. If the login shell (last field) for **uucp** is not **/usr/lib/uucp/uucico**, change it so it is.

10.4.2.4 Modify L.sys

If you will call other sites, you must modify the file **/usr/lib/uucp/L.sys**. If you do not plan to call other sites, you can still modify this file if you have the relevant information; but it won't have any effect either, since **L.sys** is activated only when you call out.

The purpose of **L.sys** is to list the most secret information about sites that you call, such as your userid and password on each remote system. Thus **L.sys** should have access mode 400 (readable only by root).

The format of lines in **L.sys** is as follows:

remote_nodename time device class phone_number login_info

where

remote_nodename	the nodename of the system you are calling.
time	times acceptable for calling. "Any" here means any time is all right; this is often used. Or you can specify days and times. Days are abbreviated Su Mo Tu We Th Fr Sa or "Wk" for any weekday. Time should be given as a range of times (e.g., 0800-1230). If no time is specified, any time is assumed to be acceptable.
device	either "ACU" if you are using one; or a hardwired device name (e.g., tty5), if the connection is direct. If your system is running Plexus Network Operating System (NOS) and this connection is on the NOS network, this can be a virtual device.
class	the line speed, usually 300 or 1200 baud for phone connections. Higher for direct connections.
phone_number	the phone number of the modem at the site you are calling. This can consist of just digits (e.g., 415-555-1212) or an optional alphabetic part derived from the file **/usr/lib/uucp/L-dialcodes**. For hardwired

	devices, this field contains the same string as used in the "device" field.
login_info	a series of expect-send fields, where "expect" is what you expect to read when you connect with the remote site (e.g., login); and send is what you will send when the expect string is received (e.g, your userid). Passwords are also normally expected.

A sample **L.sys** line for a site using an ACU might be the following

pip Any ACU 1200 415-555-1212 login:-EOT-login: Ppip ssword: giants82

This says that you contact the site whose nodename is pip. Their phone number is 415-555-1212 and they have given you the userid Ppip and the password giants82.

A sample **L.sys** line for a direct connection might look like this

pip Any tty5 9600 tty5 login:-EOT-login: Ppip ssword:giants82

This says that you have a direct link with the site pip over tty5, at a line speed of 9600 baud.

The **L.sys** line for a NOS connection using a virtual terminal looks like the line for a direct connection.

pip Any vtty.pip 9600 vtty.pip login:-EOT-login: Ppip ssword:giants82

This says you have a link with the system pip over the virtual terminal vtty.pip at a line speed of 9600 baud.

10.4.2.5 Modify L-devices

The file **/usr/lib/uucp/L-devices** must be modified to contain information about your ACU or direct line. Add a line whose format is

type line call_unit speed

where

type	ACU or DIR (direct).
line	the logical device you are using, e.g. tty6.
call_unit	the ACU. Direct lines have "0" in this field.
speed	the line speed.

For example, if you use **/usr/plx/dial** and tty6, add the following line to **L-devices**:

ACU tty6 /usr/plx/dial 1200

If you use a direct connection over tty6, the line would look like this:

DIR tty6 0 9600

If you use a virtual terminal connection with the virtual terminal vtty.pip, the line would look like this:

DIR vtty.pip 0 9600

10.4.2.6 Modify USERFILE

All sites connected by **uucp** must modify the file **/usr/lib/uucp/USERFILE**. This is where **uucp** looks when another system announces it has work to do on your system. **Uucp** uses USERFILE to double-check that the system is allowed on your machine and to find out what that system is allowed to do. Even if you never receive calls from other systems, you still have to change the USERFILE, so that you can control what the slave sites can do during that portion of the conversation when they execute their jobs on your system. Like **L.sys**, the USERFILE is confidential, and should have access mode 400.

Lines in USERFILE have the form

login,remote_nodename [c] pathname(s)

where

login	the login name you assigned to the caller. This must be followed by a comma.
remote_nodename	the nodename of the caller. If this field is blank, then the line applies to anyone with the same login name.
c	optional field, which designates that callers should be called back to confirm their identity. If you are not set up to call out, you shouldn't specify this.
pathname(s)	the directories to which the remote user has access.

If the system "pip" calls you, and you have given them the login name "SysPip" and you don't want to call them back, and you want them to have minimal access to your system, your USERFILE line for them should read

SysPip,pip /usr/spool/uucppublic

USERFILE must contain a line for the user uucp in order to perform remote functions. That line might be

uucp, /usr/spool/uucppublic

USERFILE must contain directory access permissions for local users in addition to remote ones. You need at least one line that applies to your own site. If you don't put this line in the USERFILE, then local users may perform no **uucp** work at all; you receive the message `ACCESS (DENIED)` in the LOGFILE for all your local users' jobs.

Remember that all the ordinary access permissions continue to apply in **uucp** transactions. Therefore, a USERFILE line such as

, /

would preserve exactly the current access permissions for all your current users. Although this line does not specify a login, it does NOT permit anybody to do anything on your system. Remote callers must have a line that matches their login name and nodename exactly; the line " , /" does not match any remote caller.

10.4.2.7 Implementing Automatic Calling

If you want to call another site regularly and automatically, you need an ACU and some way of regularly executing a shell procedure to start up **uucp**. The latter is usually accomplished by putting an entry in **/usr/lib/crontab**.

For example, suppose you want to link every hour. You first need a shell procedure to establish the link. Then you name that shell procedure in **crontab** with instructions to execute it every hour.

For the shell procedure, two simple lines like the following will do:

```
/usr/bin/uulog
/usr/lib/uucp/uucico -r1 -s<system_nodename>
```

The angle brackets around the system_nodename are **not** part of the syntax; they are just a signal to you to substitute here the name of the system you want to call. So if you want to call nodename plx, this line would read

```
/usr/bin/uulog
/usr/lib/uucp/uucico -r1 -splx
```

/usr/bin/uulog wakes up the **uucp** log, so these transactions will be properly recorded. It puts its record in the file **/usr/spool/uucp/LOGFILE**. **Uucico** is the name of the program that performs all the transfers.

Suppose you name this shell procedure "hour" and put it in the directory **/usr/lib/uucp**. Don't forget to **chmod** it to make it executable. Then, to get it to execute every hour (e.g., at 20 past the hour), the line in **crontab** would read

```
20 * * * * /usr/lib/uucp/hour
```

For more information, see **cron**(1).

10.4.2.8 Optional Procedures

The following two procedures are optional but recommended if you will be automatically calling other sites. These are **uucp** system administrator tasks; ordinary **uucp** users do not have to know about them.

10.4.2.8.1 Purge the Logs

If you communicate via **uucp** often, you may want to set up automatic purging of the file **/usr/spool/uucp/LOGFILE**. This file contains records of transactions over **uucp**; every execution of **uucico** is recorded, along with information about whether or not the call succeeded, what problems were encountered, and what jobs were done. If you call just one other site every hour with minimal activity (mail only), the LOGFILE can grow as large as 40K bytes per week. If you run **news**, the log is routinely over 500K bytes per week. So you can see the need for keeping its size under control. To purge the LOGFILE, execute

```
cp /usr/spool/uucp/LOGFILE /usr/spool/uucp/o.LOGFILE
rm /usr/spool/uucp/LOGFILE
```

You'll probably want to make this a shell procedure. Note that this procedure always saves the last LOGFILE if you need to refer to it. Suppose you want to do this weekly, so you call this file "week". You put it in **/usr/lib/uucp** and make it executable. Then, this shell procedure can be automatically invoked weekly by a **crontab** entry such as the following:

30 5 * * 1 /usr/lib/uucp/week

This executes the file **/usr/lib/uucp/week** at 5:30 AM on Monday mornings.

The same considerations apply to the file **/usr/spool/uucp/SYSLOG**, but with less urgency. If you run **news**, the file **/usr/lib/news/log** should also be cleaned up every few weeks.

10.4.2.8.2 Remove Unused Job and Lock Files

Finally, you may want to remove on a regular basis job files that are queued but for some reason (e.g., line problems, login problems) were never sent or never sent completely. These include temporary data files (prefix TMP), copy files (prefix C) and data files (prefix D). You may not have any of these in your **/usr/spool/uucp** directory; they come into existence when **uucp** starts operating. Read more about these in the regular UUCP documentation if you need to know about them.

You may also want to remove those files that prevent **uucp** transmissions from taking place: system status files (prefix ST), and lock files (prefix LCK). The status files have the form **STST**.sys", where sys is the nodename of the system you are trying to call. They are created when ordinary failures take place, such as encountering a busy line. They prevent repeated tries for about 55 minutes. They are important because if you want to start **uucp** manually before the 55 minutes are up, you must first remove the ST file for the system you are calling. Since any user can start up **uucp**, any user can delete ST files.

Lock files prevent the device you are using from being accessed by another user while **uucp** transmissions are taking place. If the system crashes during transmissions, these files may be left in the **/usr/spool/uucp** directory and inhibit any further activity on the device until they are removed. Therefore, check for these and remove them if you find you are unable to make a connection.

The program **uuclean** removes unwanted files and sends mail to root and to the user whose job was lost. To have the **uuclean** performed regularly, a shell procedure like the following can be used:

/usr/lib/uucp/uuclean -pTM -pC. -pD.
/usr/lib/uucp/uuclean -pST -pLCK

The -p option means remove all files with this prefix. The **uuclean** program automatically removes files older than 72 hours. You can change the number of hours a file must age with a -n option to the **uuclean** command: for example, -n12 for 12 hours instead of 72. If you want this performed daily, you can name the file "day", make it executable, and put it in **/usr/lib/uucp**. Then you need a line in **crontab** like the following:

0 20 * * * /usr/lib/uucp/day

This executes the "day" program every day at 8 PM.

Chapter Eleven

THE STANDALONE ENVIRONMENT

Standalone programs are programs that run independently of UNIX. Such an environment is sometimes required for Sys3 UNIX to create, alter, or repair the environment in which UNIX Sys3 runs. Under certain circumstances you could not change the Sys3 environment without disabling the operating system (and your accumulated files and programs) at the same time. Programs of this category alter or maintain the organization and/or format of the disk drive(s).

Some Sys3 UNIX standalone utilities are provided, not to alter the environment, but to check system status of various parameters while the system is "down." Others provide device-to-device copying, backup, and/or restore capabilities when it is inconvenient or impossible to bring the system up to single-user or multi-user mode first.

Although you will frequently run the Sys3 versions of several standalone programs described in this chapter such as **cat** and **ls** you will rarely find it necessary to invoke the standalone environment.

You will definitely need to invoke standalone programs, however, if you:

- Install a new disk
- Experience a catastrophic disk failure
- Increase the swap area or change its location
- Spare some bad disk tracks
- Implement more than two file systems on your disk
- Change the default logical disk sector addresses on the disk

Notice that most of these examples are once-per-system occasions.

This chapter explains what these standalone programs do and gives general instructions for their use.

Plexus release tapes contain some or all of the following files. The number in parentheses before the name indicates the position of the file on the release tape.

(0) **help**	Gives information about the release tape and use of standalone programs.
(1) **boot**	Secondary boot. Loads requested program into memory and executes it.
(2) **sys3**	A binary copy of the Sys3 UNIX operating system kernel.
(3) **dformat**	Disk format. Formats a disk drive, prompts for format information, enables sparing of bad disk sectors.
(4) **mkfs**	Make file system. Creates and organizes a bare file system. See MKFS(1).
(5) **restor**	Restore file system from dump tape. See RESTOR(1).
(6) **fsck**	Interactive file system consistency check. See FSCK(1).
(7) **dd**	Device-to-device copy program. Can read out of file systems. See DD(1).
(8) **fbackup**	Fast backup. Does a quick image copy from disk to tape or from tape to disk.
(9) **od**	Octal dump. Dumps a file to the screen in octal or hex format. See OD(1).
(10) **dconfig**	Configures a disk with initialization information and the logical file system layout for Sys3 UNIX.
(11) RESERVED	
(12) **fsdb** *	File system debugger. See FSDB(1).
(13) **du** *	Reports disk usage. See DU(1).
(14) **ls** *	Lists contents of directories. See LS(1).
(15) **cat** *	Concatenates and prints files. See CAT(1).

A few general remarks about standalone programs:

1. NEVER run the standalone version of a program while UNIX is booted.
2. Standalone programs cannot create or write to regular disk files, although some can write special files.
3. Standalone programs do not allow redirection, pipes, or pattern matching on the command line.
4. If you make a mistake when you type in the program name, # or <backspace> erases one character; "@" or <control-x> erases one line.
5. The delete key or rubout key aborts a running standalone program.

* These standalone programs are in the /**stand** directory on disk, but may not be included in your tape copy of standalone programs.

11.1 How to Run a Standalone Program

Most standalone programs can be run either from tape or from disk. This means you can use either the copy of the program that resides on tape or the one on disk. (Naturally, if your disk is bad, you will use the tape version of the program; this is the point of having two copies.) Disk standalone programs reside in the directory **/stand**.

To run any standalone program (or any program), you must name it unambiguously. Therefore, some conventions are in force to specify program names. For historical reasons, two formats work — one longer, one shorter. The longer command format is required only in very unusual circumstances — e.g., if you have two programs with the same name on a tape, and you need to distinguish between them. The longer method is never necessary for disk resident standalone programs.

The Longer Method

Programs may be specified by a pathname of the form:

device (number, offset) program_name

where

device	is a device specifier — pt is tape, pd is disk.
number	is a device number (from 0 to 3).
offset	is the starting point (in files for tape, sectors for disk). For disk files, offset is the offset of the *file system* that contains the program, not the offset of the program itself.

To call disk-resident programs, you use the "device (number, offset)" prefix, specifying both the starting address of the file system that contains the program, and its complete pathname. This prefix is not always necessary to call tape-resident programs. For example, no ambiguity in naming can result if you run the standalone programs from the Plexus release tape, since Plexus release tapes do not duplicate program names. In this case you can omit the pathname: if you use the Plexus tape version, you can just mount the tape, obtain the primary boot prompt, and type the program name.

Thus, using this method, to execute the standalone program **mkfs** from the release tape, only the program name must be typed in response to the boot prompt. (The boot prompt is `PLEXUS PRIMARY BOOT REV` X.X, where X.X is the revision number.) So to execute the tape resident program, the sequence is

```
PLEXUS PRIMARY BOOT REV X.X
: mkfs
```

However, if you had two programs named **mkfs** on the tape (something that would never happen on a Plexus release tape), you would have to specify which one you wanted. To specify one at offset 10 (the tenth file on the tape) on tape unit 0, you would type

```
PLEXUS PRIMARY BOOT REV X.X
: pt(0,10)mkfs
```

To execute this same program from disk, the long method requires

```
PLEXUS PRIMARY BOOT REV X.X
: pd(0,0)stand/mkfs
```

In this example, pd indicates a disk device (pd is driven by the IMSP disk controller, (0,0) indicates disk unit 0 at offset 0, and **stand/mkfs** indicates the pathname of a file within a file system starting at this location.

The Short Method

All P/35s permit truncating both the tape and the disk-program names; the slash (/) as the first character in the pathname is a sort of shorthand the system understands to mean **/dev/dk0**, which means the program is disk resident. (You may, of course, give the whole longer-style pathname, too.)

Thus, to execute the standalone program **mkfs** from the release tape, only the program name must be typed in response to the boot prompt. So to execute the tape resident program, the sequence is

```
PLEXUS PRIMARY BOOT REV X.X
:  mkfs
```

To execute the same program from disk, the short method allows just

```
PLEXUS PRIMARY BOOT REV X.X
:  /stand/mkfs
```

The slash (/) indicates that *stand* is on the disk.

Most of the standalone programs have counterparts described in the *Plexus Sys3 UNIX Programmer's Manual — vol 1*. When called, the standalone programs all return a double dollar sign ($$), instead of the usual single dollar sign, followed by the program name. The cursor is positioned one space to the right of the program name. You type arguments to the program there. For example, if you type

```
pd(0,0)stand/dd
```

or

```
/stand/dd
```

the system responds

```
$$ dd □
```

where □ represents the cursor.

Thereafter, the arguments to the standalone programs are usually the same as described in the *Plexus Sys3 UNIX Programmer's Manual — vol 1*.

The standalone programs that do not follow the Volume 1 formats are generally interactive; that is, they prompt for any parameters needed.

Upon completion, standalone programs print "Exit" followed by a value. The value "0" indicates that the program completed normally.

11.1.1 Standalone Programs and Special Files

The following special files can be used as arguments to the standalone commands:

/dev/dkXX	Logical disks on default physical disk(s).
/dev/pdXX	Logical disks on IMSP physical disk(s). *XX* ranges in value from 0 to 15 for disk unit 0 (pd(0,0)), 16 to 31 for disk unit 1 (pd(1,0)), 32 to 47 for disk unit 2 (pd(2,0)), and 48 to 63 for disk unit 3 (pd(3,0)). The disk addresses of the logical disks are listed in Section 11.3, "dconfig." For more on logical disks, see the account of **mkfs** below and in the chapter, "Configuring Disks." (The logical disk addresses may be changed by the standalone program **dconfig**.)
/dev/mt0	Default tape.
/dev/nmt0	Default tape with no rewind.
/dev/pt0	IMSP cartridge tape.
/dev/npt0	IMSP cartridge tape with no rewind.
/dev/null	The bit bucket.

The following restrictions apply to the use of these special files in standalone programs:

1. The initial access to an IMSP tape (device 'pt') rewinds it before reading or writing it.
2. Writing to an IMSP tape (device 'pt') must be in multiples of 512 bytes.
3. Standalone programs automatically buffer read requests from special files; the buffer size is 1024 bytes. Thus read requests for less than 1024 bytes result in 1024 bytes being allocated per read anyway.
4. Writing to any of the special disk files (**/dev/dk**XX or **/dev/pd**XX) must be in blocks of 1024 bytes per write. Thus the command

   ```
   $$ dd if=/etc/rc  of=/dev/dk2 obs=1024
   ```

   must include the 'obs=1024' argument.

5. Once a cartridge has been rewound, writing to it can overwrite data on it even if you try to skip to the last file on the cartridge. See **pt**(4) in the *Plexus Sys3 UNIX Programmer's Manual — vol 1*.
6. The standalone commands normally use **/dev/dk1** as the root file system. This can be changed by changing the *rootdev*, *pipedev*, *dumpdev*, and *nswap* values in block 0 of **/dev/dk1**. See the description of the **dconfig** standalone command in this manual.

11.1.2 The Mount and Skip Commands

The standalone commands recognize two special built-in commands that set up their environment. These are **mount** and **skip**. **Mount** allows access to disk files on file systems other than **/dev/dk1**. To use it, type an exclamation point (!) followed by a **mount** command in response to the standalone prompt $$. For example, to concatenate files on a file system on **/dev/dk2**, (user response in **bold**):

```
        $$ cat ! /dev/dk2 /fs2
        <ctl-d>
        $$ cat /fs2/user/foobar
```

Note that typing <ctl-d> causes a reprompt of $$ followed by the command name.

The **skip** command allows direct access to particular files on tape; it positions the tape at the beginning of the file number you specify. To use **skip**, type an exclamation point (!) followed by the **skip** command in response to the standalone prompt $$. For example, to concatenate tape file 3 (user response in **bold**):

```
        $$ cat ! skip /dev/nmt0 3
        <ctl-d>
        $$ cat /dev/mt0
```

Note that entering <ctl-d> causes the reprompt $$ followed by the name of the command. Note also that the **skip** command requires **/dev/nmt0** rather than **/dev/mt0** — using the latter would rewind the tape before the second prompt.

11.2 What Standalone Programs Do

This section describes in detail the functions of all the standalone programs, with special emphasis on the non-trivial standalones that create or alter the disk environment for Sys3 — **dformat**, **dconfig** and **mkfs**.

11.2.1 dformat

dformat is the controlling program for formatting hard disks. Formatting involves encoding the disks' magnetic surface with boundaries so the operating system can find byte patterns to read or blank spaces to write to. Each disk is divided (encoded) into tracks and sectors. Tracks are magnetic patterns of concentric circles on the disk surfaces. Tracks are divided into sectors.

To the user, sectors have 512 bytes each, but actually, **dformat** "knows" that each sector is larger than 512 bytes by about 15%. This larger figure is known as the "actual bytes per sector," and it varies according to the disk controller and disk drive installed in your system. For example, a disk might actually have 586 bytes per sector. The extra 72 bytes (beyond 512) provide the disk controller hardware and firmware the pointers and addresses it needs to mark the beginning sector data.

A microscopic flaw in the magnetic coating on a disk can prevent the operating system from reading or writing to the sector(s) or track(s) that share disk space with that flaw. When this happens, that track or sector must be "spared." That is, to keep the disk usable, the operating system must be informed that that track or sector is bad. Then it can set up a new area of the disk to substitute for the bad part, remap the disk to skip the bad and seek the substitute sector or track.

The standalone program, **dformat**, performs both types of disk preparation and maintenance routines: initial setup and formatting, and sparing — either automatic or manual.

The main purpose of **dformat** is to set sector size, number of sectors per track, number of alternate cylinders or tracks, etc. Like any operating system, Sys3 can only access its disk by sectors. It must know exactly how large these sectors are in order to transform block requests into sector requests. Once the sector size and other issues have been decided, **dformat** goes out and defines sectors on the disk, writing sector headers for each sector. These headers make the sector able to identify itself to the disk controller — and ultimately to Sys3 — when the sector is accessed, so the system can be sure it's getting the right one.

dformat can also perform disk tests and repairs: it finds out if the disk can write data, and whether the disk can read what it writes. If **dformat** finds a sector it can't read or write, it automatically spares it. It sets up a bad-sector table such that any requests to access a bad sector are automatically redirected to a good sector. You can also use **dformat** to manually spare any bad tracks or sectors reported by a Sys3 error message.

Dformat has several options. Not every option is available on every Plexus machine.

f Format the disk. The formatting operation has two phases: format and test. The format part consists of writing sector headers, and writing the user pattern on the disk. The test portion consists of reading the user pattern just written, looking for unreadable sectors, and sparing any unreadable sectors it finds. Error messages of the form

    track <cylinder>/<head> spared

  are printed for all bad sectors found and spared.

s Spare tracks. Enables manual (i.e., operator intervened) sparing of tracks.

r Read the disk, performing only a surface analysis. Does not re-format or spare bad tracks. This option generates and prints a report of the bad tracks on the disk, in the form

    bad trk <cylinder>/<head>

  This option is available only on disks with IMSP controllers.

l List bad tracks. This option prints a table of all the bad tracks (pd) and their corresponding spares, in decimal and hex.

Only one of these options can be requested per invocation of **dformat**.

NOTE

> **dformat** spares disks controlled by the IMSP by whole tracks. If there is a bad area on the disk, it remaps to one of twenty alternate tracks.

Dformat with the f option rewrites the disk's sector headers. These are written by Plexus in the factory and should never have to be rewritten. The f option should be used only after consultation with Plexus field service (hardware maintenance) personnel. In most cases, no matter how badly the file system has been scrambled, the disk can be completely restored by the use of **dconfig**, **mkfs**, and **restor** in that order.

Dformat with the f option writes a pattern of bits all over the disk, so any previous contents of the disk are lost. If the disk still has good data on it, back it up before running **dformat** with the f option.

The default sector size is 512 bytes; the default block size is 1024 bytes. The following table gives the normal total number of 512-byte sectors created by **dformat** on Plexus disks.

TABLE 11-1. Total Sectors on Plexus Disks

Total Available Sectors on Plexus Disks	
Disk Size	**IMSP (pd)**
22 Mb 8"	40290
36 Mb 8"	67150
72 Mb 8"	135422
142.6 Mb 8"	272544
72 Mb 14"	136510
145 Mb 14"	273020
289 Mb 14"	546176

The total number of 1K-byte file system blocks can be calculated by dividing each sector count above by 2.

11.2.2 mkfs

The standalone program, **mkfs**, like its **/etc/mkfs** counterpart, creates file systems. It declares that a certain range of disk addresses is to be considered a file system, and sets up special disk blocks — the superblock, the free list, and the i-list — within that file system for UNIX to maintain housekeeping information about the files within the file system.

The special blocks contain information such as how many files are in the file system, how many blocks are free, where files live on the disk, and how big they are. The file system created by **mkfs** is empty and is organized as illustrated in Figure 11-1.

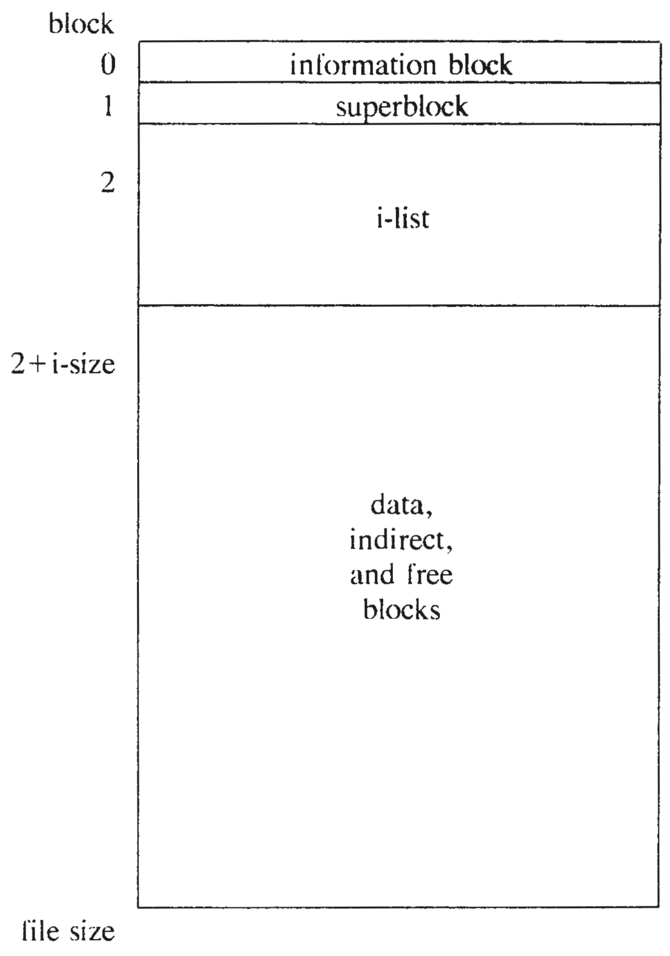

Figure 11-1. mkfs File Organization

In Figure 11-1, "isize" is the length of the i-list, and "file size" is the length of the file system. For more detailed information, see the article "FSCK" in the *Plexus Sys3 UNIX Programmer's Manual — vol 2B*.

When creating mounted file systems (i.e., file systems other than the root file system), you can run **/etc/mkfs** under UNIX or **mkfs** of the standalone environment. When creating the root file system, however, standalone **mkfs** *must* be used because it prepares a file system so Sys3 and its utilities (the root file system) can be **restore**d from tape onto the disk. Plexus has already run standalone **mkfs** on the root file system, so you will not need to run this version unless you alter the default size or location of the root file system or the swap areaz

Because **mkfs** re-initializes the superblock and i-lists, any data that is already on the logical disk, though not overwritten, becomes inaccessible, since UNIX loses all pointers to it. Therefore **mkfs** must be used with caution.

After you run **mkfs** and before you **mount**(1) the file system you made, it's a good idea to run **fsck** on the new file system, to insure the integrity of the file system.

11.2.3 dconfig

dconfig is a rather comprehensive program that determines several disk and system factors. You can run **dconfig** to obtain or change the status of any of the following parameters:

- The disk identification (id)
- The kernel name (primary bootname)
- An alternate kernel name (secondary bootname)
- Disk information specific to the actual disk unit installed, including number of cylinders, sector size, sectors per track, etc.
- Your system's node name (if you want to give your system a name to put it on a **uucp** network)
- Which logical disk will contain the root file system and perform pipes and dumps
- Which logical disk will provide the swap area
- The address and size of the swap area
- The boundaries and default start addresses of each potential logical disk

These parameters are vital to the system. Some parameters (such as disk cylinders, sector size, etc.) have no alternate values for your system. Changing them simply disables your system.

The other parameters can be changed if done carefully and consistently, such as the device numbers for root, swap, etc., the swap size and location, and the start addresses of the logical disks.

Changing your nodename will not affect internal system performance, but other nodes on your **uucp** net must be notified if you still want their messages to "find" your system.

You can run **etc/dconfig** or standalone **dconfig** to *look at* the status of any of the parameters previously mentioned. You should run standalone **dconfig**, however, to *change* any of the parameters. You will most likely invoke standalone **dconfig** to perform the following functions:

1. Create a Sys3 node name for **uucp**
2. Enlarge or relocate the swap area
3. Divide your disk into 3 or more file systems

The procedure for the first function is contained in the chapter, "Implementing UUCP." Procedures for performing numbers 2 and 3 are contained in the chapter, "Configuring Disks."

Because **dconfig** is so comprehensive, the following page displays the prompts you will see when running it. The responses in bold show the default values or factory settings according to the type of system you have.

"Early P/35s" are P/35s shipped with Sys3 version 3.13 or earlier. They have a default swap area of only 2 Mb, and should probably be enlarged according to the swap increase procedure in the chapter, "Configuring Disks." "New P/35s" are systems shipped with Sys3 version 3.2 or later. New P/35s have a 6 Mb swap space and a 24 Mb sized **/dev/dk1**. P/35's shipped with the smallest disk (22 Mb) have a 4 Mb swap area and retain the 20 Mb size of **/dev/dk1**.

The responses marked by an asterisk (*) vary widely; the range of responses are indicated in Tables 11-2 and 11-3 that follow the **dconfig** display, below.

```
: dconfig
$$ dconfig
Disk? :                                        /dev/dk0
Disk id? [pd]:                                 pd
Primary bootname? [/sys3]                      /sys3
Secondary bootname? []:                        <return>
Number of cylinders? [0]:                      *
Number of heads on removable? [0]:             0
Number of heads on fixed? [0]:                 *
Data bytes per sector? [0]:                    512
Sectors per track? [0]:                        *
Number of alternate cylinders? [0]:            20
File system blocksize? [0]:                    1024
Sys3 nodename? []:                             <return>
Change the default unix device mapping? [y/n]: y
```

[The following six items appear only if you answer "y". You may confirm with carriage return. Change these only if you really know what you're doing.]

```
                                                  Early P/35s   New 22Mb P/35s   New P/35s

Rootdev? [0x0]:                                   0x1           same             same
Pipedev? [0x0]:                                   0x1           same             same
Dumpdev? [0x0]:                                   0x1           same             same
Swapdev? [0x0]:                                   0x1           same             same
Swplo? [0]:                                       36000         32000            36000
Nswap? [0]:                                       4000          8000             12000
Change the file system disk configuration? [y/n]: y
```

[The following sixteen lines appear only if you answer "y". You may confirm the default values by carriage returns.]

```
File system logical configuration? [sector start,sector count]

                                              Early and
                                              22 Mb P/35s   New P/35s
dk0   [0,0]:                                  0,-           0,-
dk1   [0,0]:                                  0,40000       0,48000
dk2   [0,0]:                                  40000,-       48000,-
dk3   [0,0]:                                  60000,-       68000,-
dk4   [0,0]:                                  80000,-       88000,-
dk5   [0,0]:                                  100000,-      108000,-
dk6   [0,0]:                                  120000,-      128000,-
dk7   [0,0]:                                  140000,-      148000,-
dk8   [0,0]:                                  160000,-      168000,-
dk9   [0,0]:                                  180000,-      188000,-
dk10  [0,0]:                                  200000,-      208000,-
dk11  [0,0]:                                  220000,-      228000,-
dk12  [0,0]:                                  240000,-      248000,-
dk13  [0,0]:                                  260000,-      268000,-
dk14  [0,0]:                                  280000,-      288000,-
dk15  [0,0]:                                  300000,-      308000,-
Is the above information correct? [y/n]:      y             y
Are you sure you want to rewrite block 0?:    y             y
Block 0 of [pd](0,0) initialized successfully!
Exit 0

PLEXUS PRIMARY BOOT REV X.X
:
```

The **dconfig** prompts for number of cylinders and for number of fixed heads varies according to which disk is installed in your P/35. Use the following tables as guides for inserting the proper values for the asterisk prompt in the **dconfig** program.

TABLE 11-2. IMSP Fujitsu Responses

Controller	**IMSP**	**IMSP**	**IMSP**	**IMSP**
Disk Model	14" 72Mb Fujitsu	14" 145Mb Fujitsu	14" 289Mb Fujitsu	8" 72Mb Fujitsu
Unit	pd(0,0)	pd(0,0)	pd(0,0)	pd(0,0)
# of cylinders	823	823	1024	589
# removable heads	0	0	0	0
# fixed heads	5	10	16	7
Data bytes per sector	512	512	512	512
Sectors per track	34	34	34	34
# alternate cylinders	22	22	22	22
Interleave factor	1	1	1	1
User pattern	a5a5a5a5	a5a5a5a5	a5a5a5a5	a5a5a5a5
File system block size	1024	1024	1024	1024
Default boot name	/sys3	/sys3	/sys3	/sys3

TABLE 11-3. IMSP NEC Responses

Controller	**IMSP**	**IMSP**	**IMSP**
Disk Model	NEC 8" 22Mb	NEC 8" 36Mb	NEC 8" 142.6
Unit	pd(0,0)	pd(0,0)	pd(0,0)
# of cylinders	415	415	1024
# removable heads	0	0	0
# fixed heads	3	5	8
Data bytes per sector	512	512	512
Sectors per track	34	34	34
# alternate cylinders	22	22	22
Interleave factor	1	1	1
User pattern	a5a5a5a5	a5a5a5a5	a5a5a5a5
File system block size	1024	1024	1024
Default boot name	/sys3	/sys3	/sys3

11.3 fsck

Fsck is a file system consistency checker like the V7 utility **icheck**, but **fsck** is more powerful and easier to use. The document "FSCK" in the *Plexus Sys3 UNIX Programmer's Manual — vol 2B* describes in great detail what **fsck** does and how it works. Standalone **fsck** can check a maximum of five file systems per invocation.

11.4 restor

The standalone program **restor** is the partner of **dump**. It puts the contents of dump tapes back onto the disk. Dumps are designated as full or incremental. A full **dump** either puts the entire file system on tape. An incremental **dump** puts on tape all the components of the file system that have been altered since the last **dump**. See the relevant pages of the *Plexus Sys3 UNIX Programmer's Manual — vol 1* for further information.

Standalone **restor** works as described in the *Plexus Sys3 UNIX Programmer's Manual — vol 1* with the following differences.

1. The x key is not supported.
2. An argument of the form "+<file no>," where <file no> is a positive integer, may follow the last argument. This directs standalone **restor** to skip over <file no> number of files on tape before doing the restore from tape. For example, the sequence below (user response in **bold**) spaces forward 20 files on tape before starting the restor.

```
$$ restor r /dev/dk1 +20
   Spacing forward 20 files on tape
   Last chance before scribbling on /dev/dk1 <return>
   End of tape
```

A few things to remember about using **restor**:

- The file system being restored must exist on the disk. **Restor** does not create a file system; if the file system does not exist, you must use **mkfs** to create the file system before running **restor**.
- Like **dump**, **restor** works in terms of file systems. Therefore, it's hard to retrieve single files from a dump tape. If you're interested in archiving single files, you should use either **tar** or **cpio**.
- **Restor** replaces the previous contents of the file system. Be sure you don't care!

11.5 dd

This program works as described in the *Plexus Sys3 UNIX Programmer's Manual — vol 1*. See the section on backups in the chapter, "Periodic System Routines," for an explanation of how **dd** fits in with the other Sys3 programs that do tape I/O.

11.6 fbackup

Fbackup is a fast, standalone Plexus program that copies from disk to tape, or tape to disk, in streaming mode, blocking files at 32 disk sectors per tape record.

See the section on performing backups in the chapter, "Periodic System Routines," for an explanation of how **fbackup** fits in with the other Sys3 programs that do tape I/O.

Fbackup is faster than **dump** and writes in a format that is understood by **dd**, so you should use **fbackup** rather than **dump** if you need the speed.

To use **fbackup**, you need to know the starting disk address of the file system, and its length in 512-byte disk sectors. If you have created your file systems (with **mkfs**) so they correspond to **dconfig**'s predefined logical disk sizes, you can determine the starting disk address and length of each file system by running **/stand/dconfig**. If you have used **mkfs** to change the file system sizes so they no longer correspond to the defaults listed by **dconfig**, then you may still use the numbers reported by **dconfig**. **Fbackup** will do its job anyway: your file systems cannot in any case be larger than the logical disk sizes reported by

dconfig, and **fbackup** does not mind if the file system size figure is larger than the size of the actual file system. (You will have trouble only if you have changed the file system sizes to larger than standard values without having increased the default logical disk sizes.)

11.7 Other Standalone Programs

Plexus provides standalone versions of the following commonly used UNIX commands. These all work as described in the *Plexus Sys3 UNIX Programmer's Manual — vol 1*.

fsdb	a file system debugger.
od	octal dump. It allows dumping of a file in various octal, decimal, or hex formats. It is mainly a debugging tool.
ls	list files and directories.
cat	concatenate and print.
du	reports disk usage.

11.8 help

Information about the release tape is displayed on the system console by the **help** program. It is self-explanatory and makes use of menus to allow access to more detailed information. To use the help program, load the release tape into tape drive 0 and, in response to the boot prompt, type either "**?**" or **help**.

11.9 boot

This standalone program is not normally used, since the primary boot in firmware performs the same function. No special operator instructions are required.

11.10 sys3

Under normal circumstances, this copy of the Sys3 UNIX operating system is never accessed. It is included only as a last-resort backup in case all other copies have been lost.

Chapter Twelve

PERIODIC SYSTEM ROUTINES

Most system administration duties are performed once-only (e.g., setting up the disk, installing **uucp**, etc.) or as-needed (e.g., adding new accounts, recovering from a crash, etc.). There are some duties, however, that must be performed on a regular basis ranging from daily to biannually. These duties are of two main types: tape backups, and maintenance of the physical condition of the computer itself.

Tape Backups

You must make some sort of tape backup daily, even if it records only the changes made in the file system(s) from the previous day. This could take the form of an incremental **dump** or selective archiving of active user files. You should also ensure that the system receives a complete backup at least once per week.

In addition, some installations **dump** all the file systems onto tape and then **restor** them on a regular basis (monthly, quarterly, or annually, depending on how much frequent disk I/O degrades overall performance). As a disk drive is accessed, the system does not necessarily put files back in exactly the same physical location as where it got them. Over a period of time, the file system loses its physical contiguity. While the system can still find the files, they become more spread out, slowing disk access time. A full **dump** and **restor** reorders all the files contiguously, which restores performance to the original level when shipped from Plexus.

Physical Maintenance

Plexus computers are built to work reliably in a wide variety of environments, but they still require daily, monthly, and biannual routines to prevent dirt, friction, or wear from impairing system performance. These routines require cleaning or replacing components or filters on the periodic bases described in the subsection of this chapter, "Preventive Maintenance."

12.1 Backups

Several Sys3 programs can put Sys3 files on tape: **tar**, **cpio**, **dd**, **fbackup**, **volcopy**, and **dump**. These programs have different purposes. **tar**, **dd**, and **cpio** allow selective archiving of files. **tar** is dedicated to tape archiving, while **cpio** is more general; it writes to standard output that can be redirected to a tape device. Both **tar** and **cpio** permit limited manipulation of tape record size. **dd** can perform EBCDIC-ASCII conversion, and allows very flexible specification of tape record size. **dd**, **tar**, and **cpio** allow you to specify exactly which files you wish to move. With different options, they also allow you to retrieve files selectively; e.g., you can mount a tape and tell these programs what files you want to retrieve.

dump, **volcopy**, and **fbackup**, on the other hand, are for backing up whole file systems. **dump** either puts the entire file system on tape, or puts on tape all the components of the file system that have been altered since the last **dump**. The first type of **dump**s called a full or level 0 **dump**. The second **dump** (archives only components that have changed) is called an incremental or level 9 **dump**.

fbackup is a fast, standalone Plexus program that copies from disk to tape, or tape to disk, in streaming mode, blocking files at 32 disk sectors per tape record.

volcopy makes a literal copy of the file system using a blocksize matched to the device. The **volcopy** files may include labels.

The following table outlines the differences among these programs:

TABLE 12-1. Sys3 Programs that Write to Tape

Program	**Usage Notes**	**Restore Using**	**Remarks**
dump	For use on file systems. Allows both complete & incremental dumps.	Standalone **restor**	
cpio	For use on files. Writes to standard output.	**cpio**	
tar	For use on files.	**tar**	Does not handle multiple tape volumes.
dd	For use on files and file systems. Does data conversion. Good for I/O on raw devices. Same format as **fbackup** if bs=32b.	**dd**	Standalone. Does not handle multiple tape volumes.
fbackup	For use on file systems. Fast. Same format as **dd**.	**fbackup**	Standalone only
volcopy	For use on file systems. Uses labels.	**volcopy**	

See the relevant pages of the *Plexus Sys3 UNIX Programmer's Manual — vol 1* for further information.

The standalone program **restor** is the partner of **dump**. It puts the contents of dump tapes back onto the disk.

12.1.1 Performing Backups with fbackup

This program copies from disk to tape or from tape to disk while using the tape in streaming mode. It blocks files at 32 1024-byte blocks per record.

For example, to copy a file system that is 20000 sectors long (512-byte sectors), starting at offset 40000 on disk drive 0, onto the tape in tape unit 0, respond to the boot prompt as indicated in **bold**:

```
PLEXUS PRIMARY BOOT REV X.X
: fbackup
```

This program will then prompt for the location of the file system (in this example, the first 20000 sectors of dk2).

```
$$ fbackup
Backup to tape or Restore from tape? [br]: b
Disk unit? [is(0-3,0)] or [pd(0-3,0)]: pd(0,0)
Tape unit? [rm(0-3,0)] or [pt(0,0)]: 0
Starting disk block number for backup/restore? [0,111719]: 40000
Number of blocks in backup/restore? [1,67720]: 20000
Disk reads complete!
Backup to tape completed successfully
Exit 0

PLEXUS PRIMARY BOOT REV X.X
:
```

The numbers in brackets in the `Starting disk block number` and `Number of blocks` lines depend on the controller type and disk type. See the tables under "Disk-Dependent Responses" in the chapter, "The Standalone Environment" for the correct values. This tape can be restored using the **fbackup** program or by using the Sys3 UNIX command dd(1) with **bs=32b**.

12.1.2 Using dump for Backups

This section describes how to create backup tapes using the command **dump**.

The **dump** command provide a system for making routine archive tapes which can then be used to recover entire file systems if data on the disk is damaged or lost.

12.1.2.1 Scheduling Dumps

A rigidly enforced schedule of dumps (i.e. making backup tapes) provides the best insurance against lost data.

There are two levels of dump: full and incremental. A full dump copies everything; an incremental dump copies only those files changed since the last dump was taken.

A typical schedule requires:

a. taking an incremental dump after every working day,

b. taking a full dump after work on Fridays.

Quarterly, you might also want to take a full **dump** and then **restor** the file systems to disk to maintain system disk I/O performance.

The tapes made for archival purposes are stored and used in rotation so that the oldest tape is always the next used. Incremental dump tapes are saved at least a week and full dumps are saved for at least a month.

The commands **dump and dumpdir** are described in the *Plexus Sys3 UNIX Programmer's Manual.* An alternate dump schedule, which works equally well, is suggested in the description of the command **dump** in these manual pages.

Installations with more than one logical file system must take a separate dump for each logical file system.

Use the following procedure to take dumps:

1. Obtain a dump tape.
2. Ensure that all users are off the system. If necessary, take the system to single-user mode (state 1) as described in the chapter, *Changing Initialization States.*
3. Insert the dump tape in the tape drive.
4. Enter the following:

   sync ; sync

   for full dumps, enter **dump 0uf /dev/rmt0 /dev/dk**X
   OR
   for incremental dumps, enter **dump 9uf /dev/rmt0 /dev/dk**X

   where dkX is the file system being dumped.
5. Wait for the dump to finish. This can take up to 30 minutes.
6. When the dump is finished, the screen displays information about the dump. Copy this information, preferably onto the tape label.

   If there is too much data for one tape, the system prompts the user to load a second tape.
7. Remove the tape(s) and store it.

12.2 Physical Preventive Maintenance

This section describes the preventive maintenance that must be done to your system to keep it functioning correctly. It includes a list of the necessary tasks, a schedule for performing them, and detailed instructions for each task. These procedures are designed to be completed without the help of service personnel.

Perform preventive maintenance tasks at the following recommended intervals:

Daily — Clean tape drive components.

Monthly — Clean or replace processor fan filters and tape drive fan filter.

Every two years — Change clock battery.

As required — Dust or wash rack exterior and all front panels.

The following sections describe each of these procedures in detail.

12.2.1 Tape Drive Maintenance

Perform this procedure after the first two hours of movement of a new cartridge and every eight hours of tape movement thereafter. During normal operation, the tape moves eight hours in a month.

Using only a lint-free cotton swab moistened with isopropyl alcohol or IBM Tape Cleaner, clean the recording head and integral tape cleaner.

12.2.2 Cleaning or Replacing the Fan Filter

Once every month, clean the system filter(s). Use the applicable removal procedure as described in the following paragraph.

Once the filter(s) have been removed either dry vacuum or rinse (in warm soapy water) and dry the filter.

Inspect the filter after cleaning. Depending upon its condition, either reinstall the filter or replace it.

12.2.3 Removal Of Fan Filters

Filters are supplied for the two fans mounted on front of the P/35 system. Two types of filters and two types of mountings are used for different systems. Older units use a single rectangular filter mounted directly behind the front panel of the chassis. See Figure 12-1 for access and removal.

Newer units use two separate filters which are mounted on the front of the front chassis panel immediately before each fan. To determine which type of filter system your unit has, examine the front bottom edge of the chassis immediately below the front cover. If you can see two retaining mounts below the two fan positions, you have the newer filter system. See Figure 12-2 for fan removal on newer models.

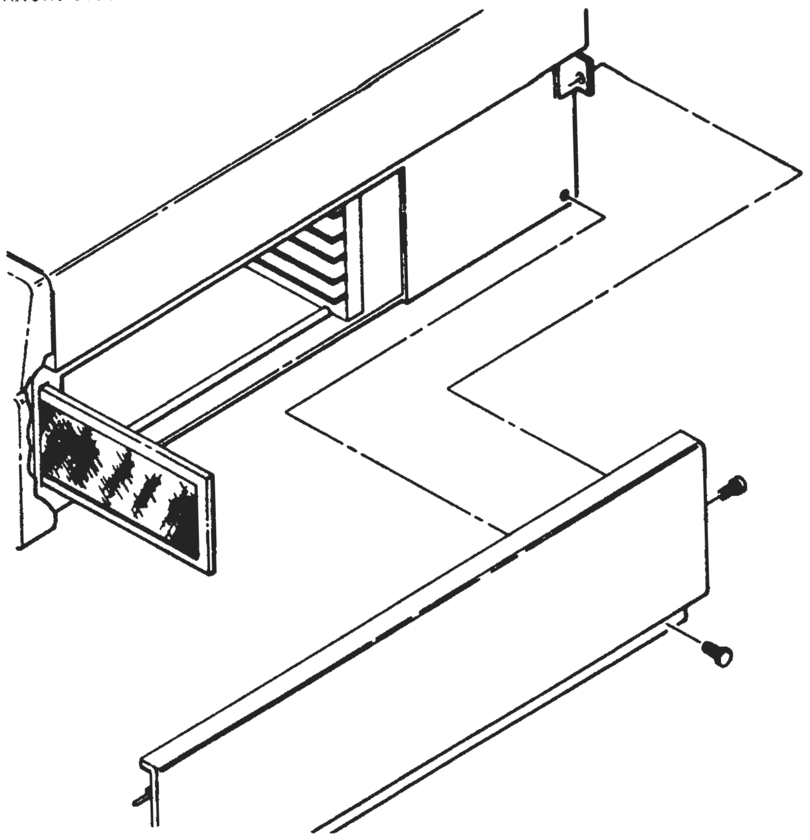

Figure 12-1. Fan Removal, Older P/35s

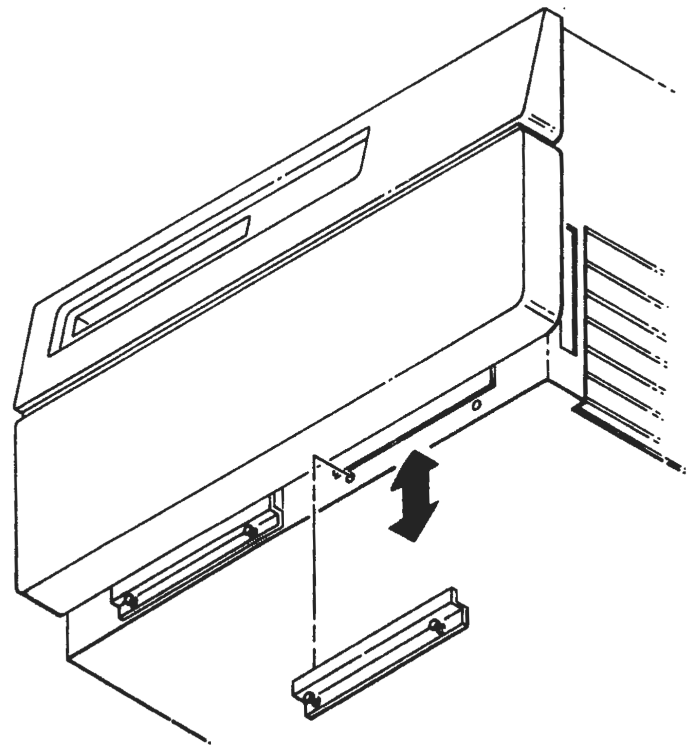

Figure 12-2. Fan Removal, Newer P/35s

12.2.3.1 Removal of Front-Mounted Filters

To remove front mounted filters do the following:

1. Raise the front of unit until it is at least six inches above the mounting surface and block in place.
2. At the front bottom edge of chassis, grasp both mounting pins of the right filter mount and pull outwards. When pins release, do not pull away from the chassis but hold it in place until one hand is free to catch the filter.
3. Remove the mount and catch the filter with your free hand.
4. Repeat steps 2 and 3 for the left mount.

NOTE

A few of the newer units were produced with a single, screw mounted, mounting flange extending across the bottom edge of the chassis. To remove the flange, remove its two mounting screws.

12.2.3.2 Removal of Rear-Mounted Filter

To remove a rear-mounted filter, do the following:

1. Remove the right side cover.
2. The filter aperture is located at the right front corner of the chassis. The end of the filter can be easily seen. A plastic tab is attached to the end of the filter.
3. Pull the plastic tab outward; the filter should slide easily from mounting slot.

12.2.3.3 Replacement of Filters

To replace either type of filter, perform the reverse of the removal procedure.

12.2.4 Replacing the Clock Battery

The system clock is maintained by a 3.6-volt battery mounted to the backplane (Figure 12-3). Accuracy of the system clock is vital to file system integrity; each time you bring up your system, verify the accuracy of the system time. If discrepancies occur, reset the system time using the **date** (1) command.

To keep the processor battery charged, run the system for at least 48 hours once every 60 days. Should the processor battery lose its charge due to extended system shutdown, it will recharge once the system is up and running. Remember to reset the system clock before using the system.

As a preventive maintenance procedure, the processor battery should be replaced every two years to ensure that it keeps the system clock current when system power is removed.

Use the following procedure to change the clock battery:

1. Obtain a replacement battery (Use a General Electric Nickel-Cadmium DS 35D, 3.6V or equivalent).
2. Shut down the system as described in the chapter, "Changing Initialization States."
3. Remove side and top covers. To remove the P/35 side and top covers, perform the following sub-procedure:
   a. At right side of unit, remove two mounting screws located along bottom of cover.
   b. At rear of unit, remove single mounting screw from rear flange of right cover.
   c. Pull cover to rear until the front retaining guidepost of the cover is disengaged from its mounting hole.
   d. Lift cover away from chassis and place it in a safe storage area.
   e. Removal of right cover exposes retaining screws (2) along bottom edge of top cover. Remove these two screws if the top cover is to be removed.

   CAUTION

   Removal of the right cover exposes the entrance to the unit's card cage and any cables connected to the front edge of the mounted PC boards. Care must be taken to prevent damage to the mounted PC boards and any exposed cables.

   f. At left side of unit, remove left side cover in the same manner as for the right side cover (i.e. steps a through d).
   g. Removal of left cover exposes retaining screws (2) along bottom edge of top cover. Remove these two screws to remove the top cover.
   h. To remove top cover, grasp it at both sides and lift rear of cover slightly upwards. Pull cover backwards to disengage locking latch located at front center of cover. Lift top cover away from unit and store in a safe place.
4. Remove mounting screws (17) from chassis top panel. Lift panel from chassis and store in a safe place.

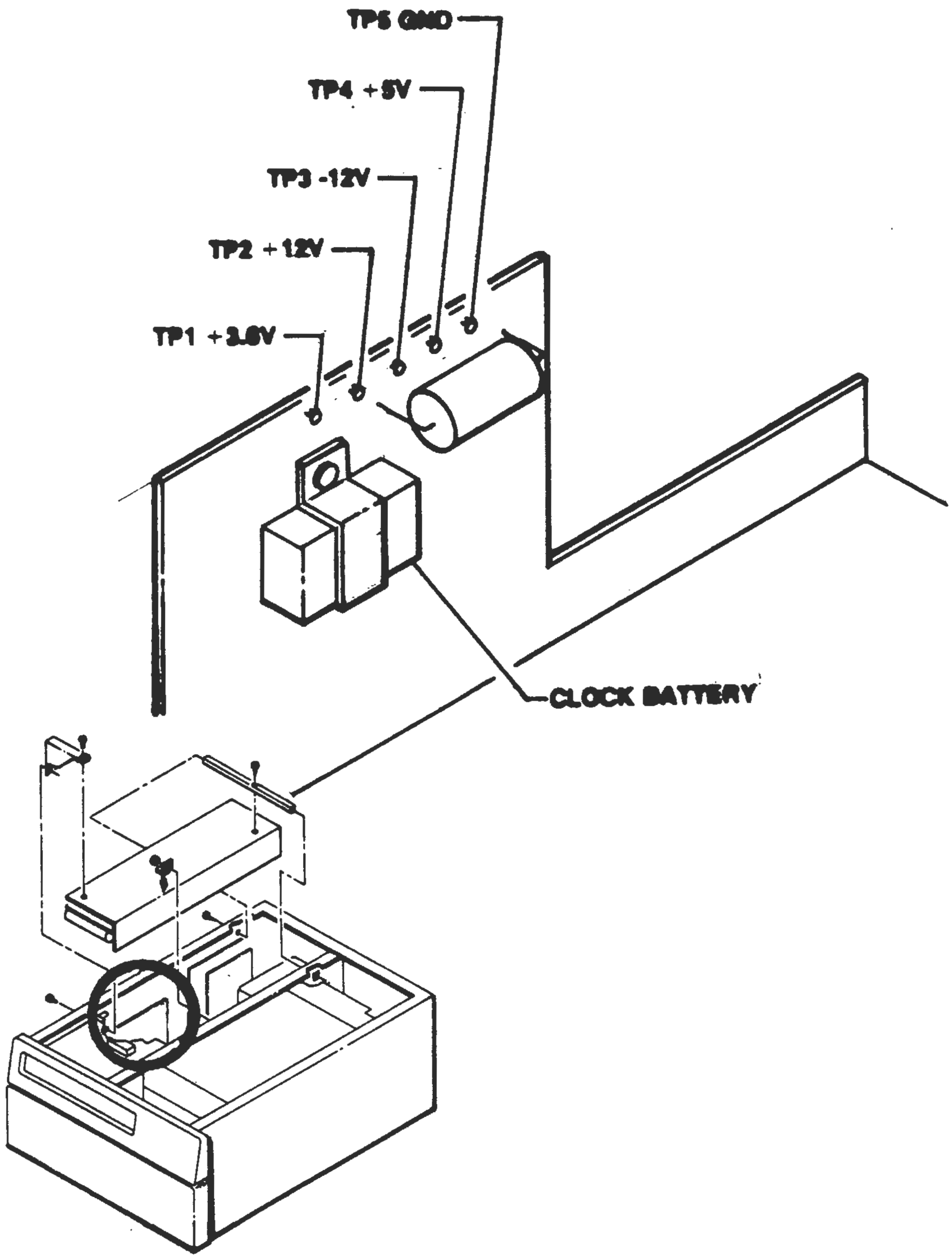

Figure 12-3. Location of Clock Battery

5. P/35s are equipped with either a 300 or a 400 watt power supply. If you have a 300 watt power supply, you do not need to remove it to gain access to the clock battery. If you have a 400 watt power supply (most likely) remove system power supply as described in the sub-procedure below:

    a. Remove locknut securing power supply hold-down bracket to centerbrace.

    b. Remove and store bracket.

    c. Remove screw securing power supply to front angle bracket.

    d. Remove angle bracket mounting screws (2) from left side of unit. Remove and store bracket.

    e. Remove mounting screws (2) securing power supply to rear mouning bar.

    f. Remove mounting screw (1) securing end of rear mounting bar to left side of chassis.

    g. Slide rear mounting bar through fitted hole in centerbrace and remove from unit.

    h. Standing at the left side of unit, pull power supply towards the left side of the chassis approximately ½ inch. Lift left edge of power supply upwards until unit is lying upside down on the disk unit. Do not disconnect any of the wires.

6. Locate the battery attached to the backplane (see Figure 12-3).
7. Loosen the single screw holding battery mounting bracket.
8. Slide the mounting bracket to the side and off.
9. Pull the old battery straight out from backplane; discard battery.
10. Install the new battery by pushing it firmly into place on the backplane, brace the backplane during installation.
11. Reinstall battery mounting bracket.
12. Reattach the power supply, reversing the steps of the power supply removal sub-procedure.
13. Reattach the top-access plate.
14. Install the chassis top and side covers.

12.2.5 Cleaning System Exterior

When the system's exterior becomes soiled, clean it using Miller Stephenson Chemical Co MS-260, Windex, or an equivalent commercial-grade plastic cleaner. Put some cleaner on a soft cloth and gently wipe all soiled surfaces.

CAUTION

Do not spray cleaner directly onto system and do not use excessive amounts of cleaner, or contamination of interior components may result. Moisten cloth with conservative amount of cleaner and wipe system gently.

Chapter Thirteen

CRISIS RECOVERY

13.1 System Crashes

If the power fails without warning while your Plexus system is busy, it can damage the file systems. This section describes how to recover from this damage.

CAUTION

> Attempting to start a damaged file system can damage it even more. Use the procedures in this section to protect file systems from further damage.

To shut down the system gracefully when there is advance warning, use the procedures in the chapter, "Changing System Initialization States."

The act of recovery after an unscheduled system shutdown includes three phases. These are:

a. Restarting the system

b. Checking and repairing damaged file systems

c. Recovering lost data

These three aspects of system recovery are discussed in the remaining parts of this chapter

13.1.1 Recovering from Ungraceful Shutdowns

After an unexpected power failure, first the power should be restored, and then the system should be brought up to single-user mode (init state 1) so that the file systems can be checked and repaired. This process is described in the chapters, "Startup, Shutdown and

Tape Drive Operation" and "Changing System Initialization States."

CAUTION

> The system is in init state 1 when the single-user prompt (#) is displayed. Do not take the system to multi-user mode until the recovery is completed.

13.1.2 Repairing File Systems

The program for checking and repairing damaged file systems in **fsck**. It is described in detail in the *Plexus Sys3 UNIX Programmer's Manual.*

Before running **fsck**, two precautions are advised. These are:

a. Create room in **/lost+found,** which is a directory in root. Copy (**cp**) a few files into **/lost+found,** then remove (**rm**) them.

b. If there is more than one file system, verify that all file systems are listed in **/etc/checklist.** This is particularly necessary if **fsck** is to be used without specifying file systems on a multi-file system installation. (The default value for **fsck** with no parameters is the list in **/etc/checklist**).

The command **fsck** should be run either:

a. once with no parameters, or

b. once for each file system, with the file system name as the only parameter.

To run the command, enter either:

```
fsck
```

or

```
fsck /dev/dkX
```

where X is the disk file system number.

The **fsck** program prompts for information when it finds damage.

After **fsck** has completed, you may want to recover lost data before taking the system to multi-user mode (state 2). This process is described later in this chapter.

When ready to return the system to normal, multi-user mode, enter:

```
/etc/init 2
```

This results in the message:

```
#cron started
 initializing ICPs
 multi-user
 type ctrl-d
```

`Type` **control d** (hold the <ctrl> key and type the character **d**) to send login messages to the terminals.

13.1.3 Recovering Lost Data

When **fsck** is done, depending on the extent of damage, some files and directories may be missing. The **fsck** program reports the names of some files it erases, but not all. To recover any erased files, there are two choices:

a. Look in **/lost+found** -- The **fsck** program puts files here when it can't figure out where they belong. Copy or move files found there back to their proper locations.

b. Recover files from backup tapes -- In the course of repairing the file systems, **fsck** removes the damaged parts. A good backup or archive system is the best insurance. The procedures for recovering data from **dump** tapes are in the next section of this chapter.

13.1.4 Running restor

The purpose for making dump tapes is to create copies of file systems on tape, to be used if data on the disk is lost. This can happen when the system is powered off without warning or when users accidentally erase data. Dump tapes are then used to recover the lost data.

Before using dump tapes to recover data after an ungraceful shutdown, see the section "Recovering from Ungraceful Shutdowns." In many cases, these procedures can be used to restore the system to its original condition, making the **restor** operation unnecessary.

Restoring part or all of a system from a backup tape is controlled by the command **restor**. This command is described in detail in the *Plexus Sys3 UNIX Programmer's Manual — vol 1A*.

There are two different restore procedures: one for restoring single files or directories and another for restoring entire file systems. Both are provided below.

13.1.4.1 Restoration of Files and Directories

To restore files and directories, do the following:

1. If the data is missing because of a system failure, restore file system integrity as described in the section "Recovering from Ungraceful Shutdowns".

2. Obtain the latest full dump tape and all incremental dump tapes taken since. If you know when the file or directory was last changed, use the incremental dump tape from that day.

3. Load the full dump tape into the tape drive.

4. Enter:

   dumpdir > **temp**

   This creates a file "temp" in the current directory. This file contains a list of all the files and directories on the dump tape with their inode numbers.

5. Enter:

   restor x file(s)

   where "file(s)" is the name of the files (or directories) to be recovered. Use the entire pathname except for the name of the mounted file system.

   EXAMPLE: For the file /usr/bin/stuff, use bin/stuff.

6. If the system responds:

```
bin/stuff: inode XXXX
Mount desired tape volume: Specify volume #:
```

   where XXXX is the inode number.

   Enter the tape volume number.

NOTE

The volume number is always 1 if the dump is only on one tape. If the dump is on two or more tapes (because it didn't fit on one tape) load the last volume and attempt the **restor**. If a prompt complains that it can't find the file, load the next volume, etc. When **restor** finds the file, use that volume number.

7. Unload the full dump tape and store it. Load the earliest incremental dump tape in the tape drive.

8. Enter:

   dumpdir >> **temp**

   This command adds the inode numbers from the incremental dump tape to the file "temp".

9. Enter:

restor x file

exactly as above. Any files which were changed between when the full dump was taken and when the incremental dump was taken are updated with the new information. Unchanged files (from the full dump) are left unchanged.

10. Repeat the above two steps with all the incremental dump tapes, working from the oldest to the newest.
11. Look at the contents of the file "temp".
12. Look for the name of each file or directory that was restored. Write down their inode numbers.
13. When **restor** restores individual files or directories, it writes them in the current directory using their inode numbers for file names. Use the inode numbers from the file "temp" to identify files, then use the command **mv** or **cp** to move them to their desired locations.
14. Unload the last incremental dump tape and store it.

If enough of a file system is damaged, it may be easier to restore the entire file system using the following procedure. Use this procedure carefully; it overwrites the entire file system.

CAUTION

If the root file system is damaged, use the standalone procedures in Chapter Eleven, "The Standalone Environment," to repair it before proceeding.

13.1.4.2 Restoring Entire Disk File System

To restore an entire file system, do the following:

1. Obtain the latest full dump tape and all the incremental dump tapes taken since then.
2. If recovering from a system crash or an ungraceful shutdown, restore file system integrity as described in the section "Recovering from Ungraceful Shutdowns". If recovering accidentally erased data, skip this step.
3. Run the program **mkfs** to create a new, empty file system.
4. Load the latest full dump tape into the tape drive.
5. Enter:

   restor r /dev/dkX

   where X is the name of the newly created file system.

   EXAMPLE: /dev/dk2
6. Remove the full dump tape from the tape drive and load the incremental dump tape taken just after the full dump.

7. Enter:

   restor r /dev/dkX
   where X is the name of the file system used above.

8. When the dump is finished, remove the tape from the tape drive and store it.

9. Starting with the next oldest incremental dump tape, load it in the tape drive and repeat the last three steps. Do this with all the incremental dump tapes starting with the oldest and working towards the most recent.

   Since incremental dumps store only files that have changed, all files changed since the last full dump are progressively updated to the level of the last incremental dump. Files unchanged since the full dump remain unchanged.

13.2 Trouble-Shooting Guide

Often when a system exhibits failure symptoms, the problem is caused by a simple oversight such as a blown fuse, a switch in the wrong position or a cable not connected correctly. This section provides a checklist to help locate and correct these problems.

For initial system-level troubleshooting, verify the following:

Check:	Refer to:
System level:	
System power cord PLUGGED IN	*Installation Guide*
System Power On indicator lit	*Installation Guide*
System power keyswitch ON	*Installation Guide*
Main system fuse OK	*Installation Guide*
On tape drive:	
Tape heads clean	"Periodic System Routines"
Tape cartridge NOT in SAFE position	
IMSP board properly seated	*Installation Guide*
On processor module:	
Baud rate set correctly	*Installation Guide*
Board properly seated	
On disk drive;	
Actuator UNLOCKED	*Installation Guide*
Disk-to-controller cables attached	

13.3 Recording a Core Dump

When a hardware or software problem causes the system to stop in the system debugger, call Plexus Field Service so the hardware problem can be corrected or the software problem diagnosed. If it is not a hardware problem and there is no software solution, Plexus Field Service may request that you take memory and ICP dumps to send to Plexus for evaluation.

If your system did not crash, but one or more of your ports hung or the system is operating improperly, follow the procedure below to provide more information:

1. Have all users who had normal output from their terminals log off. Conversely, do not have users log off who will change the condition you are reporting.
2. Record the present state of the processing environment by issuing the following command:

   ps -efl > psefl[datetime]

   where [datetime] contains no special connectors or separators.
3. Update the superblock by typing:

   sync;sync
4. Continue onto the next procedure *starting at step 5.*
5. At step 10 in the following procedure, modify the **tar** command to include the **ps** files you prepared in this procedure. Your **tar** command should look like this:

   tar -cvb 20 icpdmp* psefl* usr/lib/dnld/* etc/inittab etc/rc sys3

   The following is the procedure for preparing the dump tape when the system has hung up or gone into debugger mode.

1. Write down any error messages exactly as displayed.
2. Press the <RETURN> key and observe whether the console echoes the command.
3. If the error is a bus error and the system reports that it is in "SYS DEBUG" then type:

   r 903400 100

   and write down the last eight (8) bytes of the resulting output.
4. If the operating system has reported that it is in "SYS DEBUG" then type **x** and write down all registers and the PC.
5. Dump the main memory as follows:
   a. Procure a second person to help.
   b. Have one person press the <RESET> button while the other immediately hits the console's exclamation point (!) immediately after the "P" of "PLEXUS SELFTEST" appears on the console screen.
   c. Insert a cartridge (set not "SAFE") into the tape slot.
   d. On the console keyboard, type:

      td(0,0)

      This command dumps the main memory onto the tape.
6. Make sure your autoboot switch is off.
7. When the memory dump has completed (it may take awhile), bring the system up into single-user mode. Dump each ICP with the command:

```
/etc/icpdmp /dev/icX icpdmpX
```

wherein "X" is 0, 1, 2, or 3 for ICPs 1, 2, 3, or 4, respectively. This will perform an ICP memory dump to the file icpdmpX.

8. Position the tape to append to the memory dump without erasing it by typing:

   ```
   /usr/plx/tape srcheof 1
   ```

9. Ensure you are in the root directory by typing:

   ```
   cd /
   ```

10. Place the ICP dumps and other pertinent information onto the tape by entering the following command:

    ```
    tar -cvb 20 icpdmp* usr/lib/dnld/* etc/inittab etc/rc sys3
    ```

11. To aid Plexus Field Service write down the revision level of each printed circuit board in your system's card cage. To do so, follow this sequence:

    a. Shutdown the system completely (i.e., turn it off).

    b. Expose the card cage.

    c. Remove any cables that obstruct the boards from removal, taking care to note the connect points for reconnection.

    d. Using the built-in ejectors, slide each board far enough out of the cage to enable reading and recording the serial numbers, revision levels, and manufacturing lots of each board in the cage.

    e. Re-insert the boards, ensuring that the edge connectors are firmly seated in the backplane sockets.

    f. Reconnect any cables that you disconnected in step C, above.

12. Write a short description of the problem you encountered, including:

    - What programs were running at the time.
    - What devices were attached to the ICP (e.g., CRTs, printers).
    - Any unusual circumstances such as high number of processes, power fluctuations, strange system behavior before the error messages appeared or the crash occurred.

    Your description is important for Plexus Field Service to quickly identify the nature of the problem.

APPENDIX A: Hardware/Firmware Diagnostics

Overview

When you turn on or reset a Plexus system, the boot PROMs instruct the main processor to test itself, its memory, and the other processor boards — the ICP (Intelligent Communications Processor for controlling serial and parallel ports and the IMSP (Intelligent Mass Storage Processor for controlling disk and tape drives). If any of the components on these processor boards fails the initial self-test, the system writes error codes to the console and/or causes certain LEDs (Light-Emitting Diodes) on the board in question to light or flash to signal the problem. If the system encounters a hardware error it prevents the system from booting up. This protects your software from being inadvertently destroyed by component failures.

The self test sequence consists of two phases within each of the processor boards tested, phase 0 and phase 1. Phase 0 is a short basic test that ensures that the particular board is stable enough to attempt to run the more sophisticated tests of phase 1. Phase 1 tests are more thorough; they check out each component on each board before returning the `PLEXUS SELFTEST COMPLETE` message.

If any aspect of the system fails selftest, the system tries to print a message to the console. It also lights a pattern of LEDs on the faulty board to help indicate the problem.

This appendix gives the test names, test numbers, and error numbers that a user might encounter during selftest. The numbering scheme for error numbers also indicates what kind of processor board has had a failure:

Error numbers 0x1 through 0xff (hex) are reserved for the CPU, also known as the main processor board. Error numbers 0x101-0x1ff indicate an ICP error; error numbers 0x201-0x2ff indicate an IMSP error.

Lists of ICP error messages and codes occur later in this appendix. The equivalent lists of IMSP failures will be supplied in a future version of this document.

The system also uses LEDs on the processor boards themselves to indicate the status and progress of the selftest. While the selftest is in progress, the LEDs light in patterns representing binary codes that indicate which individual test within selftest is currently underway. This appendix also shows what these LED codes indicate. Reading the LEDs requires that you are able to see the front edges of the processor boards as they rest in the card cage. This introduction includes a procedure for gaining access to the card cage.

This appendix is provided for reference only. Plexus does not necessarily require end-user customers to involve themselves with the computers at the hardware/firmware interface level. We provide the diagnostic codes here, however, to aid customers when talking to field service if your system returns a diagnostic message during selftest. Having this information in hand can speed up communication with Plexus Field Service and shorten the time to repair the system when your field service representative arrives at your installation site.

Accessing the Card Cage

If you intend to look at the diagnostic LEDs or set any switches on the processor boards themselves, you must be able to gain access to the system card cage. To access the card cage of a P/35 and/or remove any cards from the card cage, follow the procedure immediately below:

1. At the right side of unit as you face the front, remove the two mounting screws located along the bottom of the cover.
2. At the rear of the unit, remove the single mounting screw from the rear flange of the right cover.
3. Pull the cover toward the rear until the front retaining guidepost of the cover is disengaged from its mounting hole.
4. Lift the cover away from its chassis and place it in a safe storage area.

CAUTION

Removal of the right cover exposes the entrance to the unit's card cage and any cables connected to the front edge of the mounted PC (printed circuit) boards. Care must be taken to prevent damage to the mounted PC boards and any exposed cables.

5. You now have the card cage exposed. Diagnostic LEDs for each of the boards face outward toward you. They are mounted near the edges of the boards closest to you.

   The 8-position DIP switch on the CPU is also on the front edge of that card. You probably need not extract this board to change or check switch settings.

NOTE

If you need to remove any of the PC boards to set switches or jumpers, follow the rest of the procedure as it continues below. If you want to simply examine the LEDs for diagnostic purposes, stop at this point.

6. Remove the PC board retaining bracket. It is a metal brace centered in front of the card cage opening.
7. Remove the I/O cable(s) from the PC board you wish to remove. To remove an I/O cable, pull the locking levers at either end of the card-mounted cable jack outward to release the cable connector. Then, disengage the cable connector.
8. To remove a PC board, pull the card ejector levers at each corner of the board, simultaneously, towards you. This disengages the board from the cage connector.
9. Pull the board straight out. Do NOT bend or flex the board in any way while removing it from the cage.
10. You may have to remove (and later reconnect) additional cables.

NOTE

Cables and mating connectors are marked to prevent incorrect pin connections. On all Plexus cables, pin 1 is indicated by a triangle on the cable connectors. Most Plexus cables also indicate pin 1 by a red stripe at the pin 1 edge of the cable. Match this stripe to pin 1 of the header.

11. To re-insert the PC board, reverse the above steps.

Card Cage Slot Assignments

The card cage PC board slot assignments are shown in the following diagram:

```
Slot #                                                   Board Assignment
=========================================================
|    +--------------------------------------------+  |
|  1 |                                            |  |  Memory
|    +--------------------------------------------+  |
|  2 |                                            |  |  Second memory*
|    +--------------------------------------------+  |
|  3 |           ===                  .....       |  |  Processor
|    +--------------------------------------------+  |
|  4 |===  ..........           ..........        |  |  Second ICP*
|    +--------------------------------------------+  |
|  5 |===  ..........           ..........        |  |  ICP
|    +--------------------------------------------+  |
|  6 |   ..........  .....  ..........  .....     |  |  IMSP
|    +--------------------------------------------+  |
=========================================================
```

* Second memory and second ICP are optional. The dots (....) represent the position of front edge (J) connectors.

=== This symbol represents the approximate location of the status LEDs on the CPU and ICP. The specific location of the LEDs can vary somewhat. ICP LEDs may be in front of, alongside, or behind the metal brace that runs the length of that processor board.

68000-based CPU Diagnostics

The CPU performs two levels of selftest, phase 0 and phase 1. In phase 0, the system checks the PROM checksums, tests the cache data, the memory map, and the cache dirty bit and page clear functions. Phase 0 is executed from assembler code, and corresponds to the the `PLEX` part of `PLEXUS SELFTEST X.X` message that appears on the console screen when you turn the system on or press the reset button. Under ordinary circumstances, if the system fails phase 0 of selftest, the selftest ends at that point, the console displays an error message, and the CPU's LEDs display a binary representation of the error encountered. The other function of phase 0 is to ensure that the hardware environment can execute phase 1 from C code.

In phase 1, the CPU performs 16 more tests from the C language; two of these tests are more thorough versions of phase 0 tests. Phase 1 corresponds to the display of `US SELFTEST X.X` portion of the `PLEXUS SELFTEST X.X` console display on power-up or reset. Unlike phase 0, phase 1 selftest does not halt at the first error encountered; it continues as long as the hardware permits. If phase 1 encounters failing components, selftest outputs the test name, test number, and failure number (error code) to the console.

In addition, the CPU board has 8 LEDs. LEDs 0 through 3 display in binary the current test in progress. If there are any failed tests, LEDs 4 through 7 display the binary representation of the number of the most recently failed test.

Hardware Dependencies

For the information in this appendix to be useful, certain characteristics of the main processor board must be present. The diagnostic/boot PROM set must be at a minimum revision level shown below. Further, the CPU has an 8-position DIP (Dual In-line Processor) switch near the front edge of the board. The settings of the switch determine several factors regarding the behavior of the console port and must be set to a workable pattern to establish a system/user interface.

PROM Dependencies

For the MC68000-based processor board (CPU), the boot/diagnostic PROM set must be the following Plexus part numbers and checksums:

Part Number	*Checksum (Hex)*
66-00207-2	6F4A
66-00208-2	B600

Switches and LEDs

The main processor board contains an 8-position DIP switch, a bank of eight small red LEDs, and five larger LEDs located in a row along the front edge of the board as you face it in the card cage. In this section we are concerned only with the DIP switch and the bank of eight small red LEDs.

The DIP switch sets the console baud rate, enables or disables Autoboot mode, selects which console port is active, and enables or disables the diagnostic mode.

CAUTION

Diagnostic mode is NOT recommended for anyone but Plexus-trained field engineers. Use of diagnostic mode by untrained users could result in further damage to your system.

The settings of each of the eight switches on the DIP switch are as shown in Figure A-1.

Set Baud Rate	{	SW0	See baud	
		SW1	rate	
		SW2	settings, below.	
			Enable	Disable
Auto Boot Mode	{	SW3	ON	OFF
Console Port P3	{	SW4	ON	OFF
Console Port P4	{	SW5	ON	OFF
(unused)	{	SW6		
Diag. Monitor	{	SW7	ON	OFF

Figure A-1. Switch S1 Configuration

Switch 7 enables diagnostic mode which should be turned on and used only by trained Plexus personnel. Switch 6 is not used. Switches 5 and 4 enable console ports P3 and P4 respectively. To enable both ports, set both switches to OFF. Turning switch 3 ON enables Autoboot mode. Switches 0 through 2 set baud rates 110 through 19200 via the ON/OFF combinations shown in the table below. The Plexus default is 9600 baud. The Plexus factory setting for all switch positions is:

Switch Number:	7	6	5	4	3	2	1	0
Plexus Setting:	OFF	OFF	OFF	ON	OFF	ON	ON	OFF

Switch: 2	1	0	Baud Rate
OFF	OFF	OFF	110
OFF	OFF	ON	300
OFF	ON	OFF	600
OFF	ON	ON	1200
ON	OFF	OFF	2400
ON	OFF	ON	4800
ON	ON	OFF	9600
ON	ON	ON	19200

The bank of eight LEDs provides binary patterns to represent the test numbers failed and/or of test numbers in progress. The LED error reporting scheme for phase 0 is somewhat different than that for phase 1. In phase 0, all eight LEDs may be used to report a single error. In phase 1, LEDs 0 through 3 report the test number in progress while LEDs 4 through 7 indicate the test number of the most recently failed test.

Self-Test Sequence

The console prints the following messages upon power up or reset:

```
< line feed >
< carriage return >
PLEXUS SELFTEST REV X.X COMPLETE
< line feed >
< carriage return >
SYSTEM CONFIGURATION: {im} {is} {rm} {ex} {ic0..ic4} xxxx Mb
```

Within the initial selftest message, certain letters or combinations represent successful completion of progressive steps in the self-test routine. The console displays the `PLEX` of `PLEXUS` first during phase 0 and then `US SELFTEST REV X.X` during phase 1. There will not be a significant time delay between the two messages output (that is visible to the user). The P is printed after the PROM checksum, the L is printed after the cache data test, the E is printed after the map test and the X is printed after the cache dirty bit and page clear test. If any errors are encountered during phase 0, the readout halts without printing `US SELFTEST REV X.X`, which is printed after successful completion of phase 0 and a C environment is created.

`COMPLETE` is printed only if the complete (phase 0 and phase 1) selftest passes. If the selftest fails then the message `SELFTEST FAILED` is printed on the line following the last error message.

For completed selftests, the other controller boards represented by the mnemonics within brackets ({ }) are output if the devices they drive are determined to be present in the system. Driver codes represent the following system components.

pd - IMSP disk controller
xy - Xylogics disk controller
pt - IMSP tape controller
rm - Tapemaster tape controller
ex - Excelan Ethernet controller
ic0-ic4 - ICP comm controller 0-4

xxxx Mb - memory size

A failure of one of the four initial tests (1,2,4, or 8; `P, L, E,` or `X`) causes the selftest to stop (provided that switch 7 of the CPU board is off) since the system is not functioning sufficiently to proceed with the selftest. It flashes the failing test number in the bank of LEDs on the processor board. All failing subtests detected are displayed on the system console with a brief error description. All failing subtests detected are also sequentially displayed in the LEDs 0-3 until you enter a <del> (go to monitor proper) or ! (go to PROM). The diagnostic monitor commands can be used to get a more detailed explanation of the failure but are intended for use by trained Plexus Technical Staff.

The LEDs briefly flash at the start of the program to verify their functionality. Then the positions of the DIP switch are displayed in the LEDs. The test number (see test description) is displayed in the LEDs 0-3 during program execution and the last failing test is displayed in LEDs 4-7.

The following is the order of the selftests, their names, and test numbers. Descriptions of the each of the tests occur after that. The phase zero tests (tests 4 & 8) contain a limited version of the corresponding Phase 1 tests (tests 4 & 8) just to ensure that a sufficient amount of the board is functional to execute C code. The phase zero tests are listed first followed by the rest of the tests.

TABLE A-1. Test Number List

Phase 0 Tests

LED	*Test#*	*Letter*	*Test Name*
●●●●●●●●	ff		LED and switch test
□□□□□□□●	1	P	PROM checksum
□□□□□□●□	2	L	Cache data (limited test)
□□□□□●□□	4	E	Map tests (limited test)
□□□□●□□□	8	X	Cache page and dirty bits cleared
		U	
		S	
		S	
		E	
		L	
		F	
		T	
		E	
		S	
		T	

Phase 1 Tests

LED	*Test#*	*Test Name*
●●●●●●●●	ff	LED and SWITCH test
□□□□□□□●	1	PROM CHECKSUM test
□□□□□□●□	2	CACHE DATA test
□□□□□□●●	3	EPCI test
□□□□□●□□	4	MAP test
□□□□□●□●	5	CLOCK MEM test
□□□□□●●□	6	CLOCK INT test
□□□□□●●●	7	SCRATCH RAM test
□□□□●□□□	8	CACHE DIRTY BIT and PAGE test
□□□□●□□●	9	REGISTER test
□□□□●□●□	a	MAP ID and PRIVILEGE test
□□□□●□●●	b	MAIN MEMORY test
□□□□●●□□	c	MAPPER test
□□□□●●□●	d	ECC test
□□□□●●●□	e	WRITE protect test
□□□□●●●●	f	PROCESSOR errors

TABLE A-2. Test Descriptions

Test descriptions:

Test # (hex)	*Test* Name	Description
ff	LED	Lights all LEDs and then copies switches into LEDs.
1	PROM CHECKSUM	Does a word sum on each pair of PROMs. Expects the sum to be zero.
2	CACHE DATA	Does standard memory test (ones,zeros,address, sliding bit,byte) to cache data bits. After completion of test, stack is put in high cache.
3	EPCI	Each EPCI is tested in the data loopback mode with a march and sliding bit pattern. The EPCIs are reinitialized before any error messages are printed out then initialized to continue the test.
4	MAP	The map is tested with 0's,1's,address,sliding bit, byte.
5	CLOCK MEM	The test saves the memory and registers for restoration after the test is complete. The memory is tested with 0's,1's,address,sliding bit.
6	CLOCK INT	The clock chip is initialized to interrupt the CPU. If the CPU can recognize the interrupt then the test passes.
7	SCRATCH RAM	The scratch RAM is tested with 0's, 1's, address and sliding bit.
8	CACHE DIRTY	The cache dirty bit and page numbers are tested with 0's, 1's, address and sliding bit.
9	REGISTER	Reads/writes registers, reads read only register and compares to expected value, writes write only register if effect can be seen in another register.

a	MAP ID & PRIV	The functionality of the id register and the privilege bits are tested. The scratch RAM is used for this testing(write protect uses main memory also). A special bus error routine is added to allow for these tests. A pattern is written to the RAM. Two pages are read protected and then they are read and summed. Then a subroutine is loaded into scratch and execute protected. The subroutine is executed. If the first instruction does not generate an error an error flag is loaded and then jumps out of the scratch RAM. The id register is loaded. The processor is put into user mode and then tries to read scratch RAM. A trap call puts the processor back into system mode if the test fails.
b	MAIN MEMORY	The size of main memory is checked and the memory is tested with address, inverted address and sliding bit.
c	MAPPER	All map entries point to same page. A location in each virtual page is incremented (same physical location). Value is compared to expected. If no memory then only mapping to scratch memory is tested.
d	ECC TEST	The ECC chip is tested in diagnostic mode and also single and double bit errors are forced. Syndromes from single bit errors are compared to expected values. If no memory boards are present then test is skipped.
e	WRITE PROTECT	Scratch RAM and main memory are written with a pattern. Some pages are write protected. memory is then cleared. The memory is checked to see if zeros were written to non write protected pages and the pattern remains in the write protected memory. If no memory then test is skipped.
f	PROCESSOR	The processor boards (IMSP and ICP) writes the results of their selftests into 68k memory. The failures if any are output. (See the appropriate functional specification for specific error information for the ICP and IMSP cards.)

Error Codes and Messages

If the CPU encounters an error during selftest, it attempts to create an error message and write it to the console. The syntax of these error messages differs between phase 0 and phase 1. Within both phases of error messages, however, is an error code, a hexadecimal number, that represents what part of the self-test failed. In certain cases, component failure may prevent the error message from appearing on the console screen. In such case the processor board is still able to indicate the number of the test that failed via its bank of eight LEDs.

The syntax of phase 0 error messages is:

<testname><error code><address><data received><data expected>

The syntax of phase 1 error messages is:

<testname><test number>FAILED<error code>

The following are the potential range of errors in the initial phase 0 tests. The board's LEDs display one of the following codes in the event of a phase 0 error.

TABLE A-3. Phase 0 LED Display Codes

LED	*Hex Value*	*Failure*
□□□□□□□●	01	PROM checksum was incorrect
□□□□□□●●	02	Cache data failed memory test
□□□□□●□□	04	Map failed memory test
□□□□●□□□	08	Cache page and dirty bits cannot be cleared
□□□●□□□□	10	EPCI A was selected and timed out
□□●□□□□□	20	EPCI B was selected and timed out
□●□□□□□□	40	Data written to cache could not be read from virtual 0

If the board is not in diagnostic mode, the LEDs flash the error code value. If the board is in diagnostic mode, the LEDs provide a constant display of the error code.

An error message can also be output to the console. Phase 0 error messages are of the form

<testname> <error number> <address> <data received> <data expected>

If any of the fields are not necessary they are output null. The <error number> is found in Table A-4. Only the PROM checksum, the cache data memory test, map memory test, and cache page test output error messages. The two EPCI tests and the virtual 0 read test will not output error messages. (If there is a fault in the EPCI hardware area, the error messages from the first four tests may also not come out.)

TABLE A-4. Phase 0 Error Message Numbers

Error Number	*Description*
1	Checksum of diagnostic PROMs was not 0.
2	Checksum of boot PROMs was not 0.
5	Cache data failed address test
6	Cache data failed sliding bit test

7	Cache data failed inverted address test
a	Map failed address test
b	Map failed sliding bit test
c	Map failed inverted address
d	Cache dirty bit and page could not be cleared

Phase 0 Error Messages

The failure messages of phase 0 tests differ from those of phase 1 since failure information is output before the test is halted. The test message gives an error code also that you can also look up on the error code list to get additional information about the failure. The following failure messages can be output during the phase 0 tests.

Error Message	*Meaning*
PROM [1or2]	PROM failed checksum test. If the test number is 1, the monitor PROMs failed; if the test number is 2 then boot PROMs failed.

NOTE

If the board has been set for 32K PROMs the proper lights will be set but the error message could be incomplete (since the processor cannot access all of the code).

Error Message	*Meaning*
DATA [5-7] x y z	Cache data RAM failed test 5, 6, or 7 at address x when it read value y which should have been z.
MAP [a-c] x y z	MAP RAM failed test a, b, or c at address x when it read value y which should have been z.
PAGE d x y 0	Cache page and dirty bits failed test d at address x when it read y which should have been 0.

Phase 1 Error Codes

The following are the errors in the tests performed by phase 1 of selftest. Phase 1 error messages are output to the console in the form:

<test name><test number>`FAILED`<error number>

If any of the fields are not necessary they are output null. The <error number> is found in Table A-5.

TABLE A-5. Phase 1 Error Codes

Error Number	*Description*
1	Checksum of diagnostic PROMs was not 0.
2	Checksum of boot PROMs was not 0.
3	Cache data failed 1's test
4	Cache data failed 0's test
5	Cache data failed address test
6	Cache data failed sliding bit test
7	Cache data failed inverted address test
8	Map failed 1's test
9	Map failed 0's test
a	Map failed address test
b	Map failed sliding bit test
c	Map failed inverted address
d	Cache dirty bit and page could not be cleared
10	EPCI A failed loopback address test
11	EPCI A timed out during loopback address test
12	EPCI B failed loopback address test
13	EPCI B timed out during loopback address test
14	Clock's calendar RAM failed 1's test
15	Clock's calendar RAM failed 0's test
16	Clock's calendar RAM failed address test
17	Clock's calendar RAM failed sliding bit test
18	Clock's nonvolatile RAM failed 1's test
19	Clock's nonvolatile RAM failed 0's test
1a	Clock's nonvolatile RAM failed address test
1b	Clock's nonvolatile RAM failed sliding bit test
1C	clock interrupt test received improper number of interrupts
1d	ID register failed sliding bit test
20	Scratch RAM failed 1's test
21	Scratch RAM failed 0's test
22	Scratch RAM failed address test
23	Scratch RAM failed sliding bit test
24	Scratch RAM failed byte test
25	Cache dirty bit and page failed 1's test
26	Cache dirty bit and page failed 0's test
27	Cache dirty bit and page failed address test
28	Cache dirty bit and page failed sliding bit test
2b	Read protect test failed
2c	Execute protect test failed
2d	ID privilege test failed
2e	User in system space (user with a23 =1)
2f	Main mem failed address test
30	Main mem failed inverted address test
31	Main mem failed sliding bit test

32	Improper amount of memory or no memory
33	Single bit error
34	Mapper func. test couldn't map into main memory
35	Mapper func. test couldn't map into scratch RAM
36	Memory board did not correct the forced sbe
37	CPU did not receive expected sbe interrupt
38	Incorrect syndrome was received for forced error
39	Expected double bit error; bus error not received
3a	Memory write protect did not function properly
3b	Received different bus error than was expecting
3e	Status register incorrect after buserror
3f	Memory error register incorrect in ecc test
40	Memory error address incorrect in ecc test
41	Unable to unlock multibus
42	Unable to lock multibus
43	Failed cache byte test
44	PROMs set for 128k instead of 64k

The diagnostic tests cannot recover from the following errors and will immediately print an error message and end the test except for 93 (single bit interrupt).

70	Received multibus interrupt 0
71	Received multibus interrupt 1
72	Received multibus interrupt 2
73	Received multibus interrupt 3
74	Received multibus interrupt 4
75	Received multibus interrupt 5
76	Received multibus interrupt 6
77	Received multibus interrupt 7
83	Received clock interrupt
90	Received powerfail interrupt
93	Received single bit or dma error interrupt
95	USART a interrupt
96	USART b interrupt
98	Illegal interrupt
a1	Address exception
a2	Buserror exception
a3	Chkinst exception
a9	Illegal instruction
b0	CPU privilege exception
b3	Spurious interrupt
b4	Trapv instruction
b6	Trace exception
ba	Zero divide
bd	Zero divide

Phase 1 Error Messages

The remainder of the tests will have error messages printed out in the following format:

<test name>(t<test number>) `FAILED` (<failure number>)

NOTE

> If a problem causes the processor to get lost the test listed will be the last test executed. This will most likely occur in the last test since if an error occurs during the boot process the last test will be listed.
>
> The exception to this is the message printed for failing boards in the system which is of the format:

<driver name> `FAILED` (<failure number>)

ICP Card Diagnostics

The ICP is the circuit board that controls interaction between the system and serial and parallel I/O devices such as printers, terminals, and modems. The diagnostics built into the ICP can function in one of two ways: It can communicate certain messages to the main processor and subsequently to the system console, or it can offer more detailed diagnostics in a standalone mode wherein a terminal becomes a dedicated console by connecting it to the ICP's tty0 port.

This section, being oriented toward the end-user, contains diagnostic information relative to system interface diagnostics only. The ICP's standalone diagnostics are reserved for Plexus-trained field service personnel.

Even in system interface mode, however, the ICP generates some diagnostic codes during selftest in the form of LED patterns on the ICP itself. This appendix supplies the meanings of those LED patterns as well.

Hardware Dependencies

For the diagnostic information in this appendix to be correct, the ICP PROMs and 8-position DIP switch (not to be confused with the CPU's PROMs and DIP switch) must conform to the information contained below:

PROM Dependencies

For the Intelligent Communications Processor board (ICP), the diagnostic PROM set must be the following Plexus part numbers and checksums, or later versions of the part numbers.

Part Number	*Checksum (Hex)*
66-00110-1	(4BF4)
66-00111-1	(0C00)
66-00112-1	(1FB1)
66-00113-1	(4F00)

Switches and LEDs

The ICP has a bank of four small red LEDs that are located somewhere in the vicinity of the front left corner of the board as you face the front of the card cage. ICP LEDs may be in front of, alongside, or behind the metal brace that runs the length of that processor board.

The ICP also has an 8-position DIP switch whose settings may affect the transmission of diagnostics to the host processor's console. You must extract the ICP board from the card cage to check or set the DIP switch. The switch settings are as follows:

Function		*ON*	*OFF*
Memory Save	SW0	Save memory contents	Perform destructive memory test
Diagnostic Reporting	SW1	Report errors to CPU	Standalone
External Clock 0	SW2*	Connect synchronous modem to tty 0—3	Disabled
External Clock 1	SW3*	Connect synchronous modem to tty 4—7	Disabled
(unused)	SW4		
(unused)	SW5		
(unused)	SW6		
(unused)	SW7		

* The settings for switches 2 and 3 depend on what sorts of devices are connected to this particular ICP. If you connect a synchronous modem (i.e., one with its own clock) to the ICP's tty port 0, 1, 2, or 3, you must set switch 2 to ON. If you connect a synchronous modem to the ICP's tty port 4, 5, 6, or 7, you must set switch 3 to ON. If either of these switches are set incorrectly relative to your board's I/O configuration, your entire system will not boot properly.

ICP Error Numbers and Meanings

The following error codes and brief descriptions correspond to the syntax of phase 1 board failures as described in the CPU section of this appendix:

<driver name> `FAILED` <error code>

For example, if ICP 0 failed the sliding bit test, the message written to the system console would be:

```
icp0 FAILED 107
```

The ICP error numbers and their meanings are as follows:

ICP Error List

Number	*Test*	*Meaning*
101	1	diagnostic PROM had incorrect checksum
102	1	system PROM had incorrect checksum
103	1	read illegal wakeup address (board not set for icp0 or 1)
104	1	inconsistent read of wakeup address
105	2	failed word address test
106	2	failed byte address test
107	3	failed sliding bit test
108	4	ctc0 channel 0 could not write/read time constant
109	4	ctc0 channel 1 could not write/read time constant
10a	4	ctc0 channel 2 could not write/read time constant
10b	4	ctc0 channel 3 could not write/read time constant
10c	4	ctc1 channel 0 could not write/read time constant
10d	4	ctc1 channel 1 could not write/read time constant
10e	4	ctc1 channel 2 could not write/read time constant
10f	4	ctc1 channel 3 could not write/read time constant
110	4	ctc2 channel 0 could not write/read time constant
111	4	ctc2 channel 1 could not write/read time constant
112	4	ctc2 channel 2 could not write/read time constant
113	4	ctc2 channel 3 could not write/read time constant
114	4	ctc3 channel 0 could not write/read time constant
115	4	ctc3 channel 1 could not write/read time constant
116	4	ctc3 channel 2 could not write/read time constant
117	4	ctc3 channel 3 could not write/read time constant
118	4	ctc0 channel 0 decremented incorrectly
119	4	ctc0 channel 1 decremented incorrectly
11a	4	ctc0 channel 2 decremented incorrectly
11b	4	ctc0 channel 3 decremented incorrectly
11c	4	ctc1 channel 0 decremented incorrectly
11d	4	ctc1 channel 1 decremented incorrectly
11e	4	ctc1 channel 2 decremented incorrectly
11f	4	ctc1 channel 3 decremented incorrectly
120	4	ctc2 channel 0 decremented incorrectly
121	4	ctc2 channel 1 decremented incorrectly
122	4	ctc2 channel 2 decremented incorrectly
123	4	ctc2 channel 3 decremented incorrectly
124	4	ctc3 channel 0 decremented incorrectly
125	4	ctc3 channel 1 decremented incorrectly
126	4	ctc3 channel 2 decremented incorrectly
127	4	ctc3 channel 3 decremented incorrectly
128	5	ctc0 channel 0 incorrect number of interrupts generated
129	5	ctc0 channel 1 incorrect number of interrupts generated
12a	5	ctc0 channel 2 incorrect number of interrupts generated

12b	5	ctc0 channel 3 incorrect number of interrupts generated
12c	5	ctc1 channel 0 incorrect number of interrupts generated
12d	5	ctc1 channel 1 incorrect number of interrupts generated
12e	5	ctc1 channel 2 incorrect number of interrupts generated
12f	5	ctc1 channel 3 incorrect number of interrupts generated
130	5	ctc2 channel 0 incorrect number of interrupts generated
131	5	ctc2 channel 1 incorrect number of interrupts generated
132	5	ctc2 channel 2 incorrect number of interrupts generated
133	5	ctc2 channel 3 incorrect number of interrupts generated
134	5	ctc3 channel 0 incorrect number of interrupts generated
135	5	ctc3 channel 1 incorrect number of interrupts generated
136	5	ctc3 channel 2 incorrect number of interrupts generated
137	5	ctc3 channel 3 incorrect number of interrupts generated
138	6	sio 0 improper write/read of interrupt register
139	6	sio 1 improper write/read of interrupt register
13a	6	sio 2 improper write/read of interrupt register
13b	6	sio 3 improper write/read of interrupt register
13c	7	sio0 channel A did not generate single interrupt
13d	7	sio0 channel B did not generate single interrupt
13e	7	sio1 channel A did not generate single interrupt
13f	7	sio1 channel B did not generate single interrupt
140	7	sio2 channel A did not generate single interrupt
141	7	sio2 channel B did not generate single interrupt
142	7	sio3 channel A did not generate single interrupt
143	7	sio3 channel B did not generate single interrupt
14c	7	sio0 channel A received incorrect character on loopback
14d	7	sio0 channel B received incorrect character on loopback
14e	7	sio1 channel A received incorrect character on loopback
14f	7	sio1 channel B received incorrect character on loopback
150	7	sio2 channel A received incorrect character on loopback
151	7	sio2 channel B received incorrect character on loopback
152	7	sio3 channel A received incorrect character on loopback
153	7	sio3 channel B received incorrect character on loopback
154	8	sio0 channel A test did not generate ctc interrupt
155	8	sio0 channel B test did not generate ctc interrupt
156	8	sio1 channel A test did not generate ctc interrupt
157	8	sio1 channel B test did not generate ctc interrupt
158	8	sio2 channel A test did not generate ctc interrupt
159	8	sio2 channel B test did not generate ctc interrupt
15a	8	sio3 channel A test did not generate ctc interrupt
15b	8	sio3 channel B test did not generate ctc interrupt
15c	8	sio0 channel A dma count register did not count down properly
15d	8	sio0 channel B dma count register did not count down properly
15e	8	sio1 channel A dma count register did not count down properly
15f	8	sio1 channel B dma count register did not count down properly
160	8	sio2 channel A dma count register did not count down properly
161	8	sio2 channel B dma count register did not count down properly

162	8	sio3 channel A dma count register did not count down properly
163	8	sio3 channel B dma count register did not count down properly
164	8	sio0 channel A did not receive character
165	8	sio0 channel B did not receive character
166	8	sio1 channel A did not receive character
167	8	sio1 channel B did not receive character
168	8	sio2 channel A did not receive character
169	8	sio2 channel B did not receive character
16a	8	sio3 channel A did not receive character
16b	8	sio3 channel B did not receive character
16c	8	sio0 channel A received incorrect character
16d	8	sio0 channel B received incorrect character
16e	8	sio1 channel A received incorrect character
16f	8	sio1 channel B received incorrect character
160	8	sio2 channel A received incorrect character
161	8	sio2 channel B received incorrect character
162	8	sio3 channel A received incorrect character
163	8	sio3 channel B received incorrect character
174	9	pio loopback received incorrect data
175	a	dma0 channel 0 low address reg failed write/read
176	a	dma0 channel 1 low address reg failed write/read
177	a	dma0 channel 2 low address reg failed write/read
178	a	dma0 channel 3 low address reg failed write/read
179	a	dma1 channel 0 low address reg failed write/read
17a	a	dma1 channel 1 low address reg failed write/read
17b	a	dma1 channel 2 low address reg failed write/read
17c	a	dma1 channel 3 low address reg failed write/read
17d	a	dma2 channel 0 low address reg failed write/read
17e	a	dma2 channel 1 low address reg failed write/read
17f	a	dma2 channel 2 low address reg failed write/read
180	a	dma2 channel 3 low address reg failed write/read
181	a	dma0 channel 0 high address reg failed write/read
182	a	dma0 channel 1 high address reg failed write/read
183	a	dma0 channel 2 high address reg failed write/read
184	a	dma0 channel 3 high address reg failed write/read
185	a	dma1 channel 0 high address reg failed write/read
186	a	dma1 channel 1 high address reg failed write/read
187	a	dma1 channel 2 high address reg failed write/read
188	a	dma1 channel 3 high address reg failed write/read
189	a	dma2 channel 0 high address reg failed write/read
18a	a	dma2 channel 1 high address reg failed write/read
18b	a	dma2 channel 2 high address reg failed write/read
18c	a	dma2 channel 3 high address reg failed write/read
18d	f	improper number of ctc interrupts
18e	f	dma word count not 0xff
18f	b	sio0 channel A improper number of ctc interrupts

190	b	sio0 channel B improper number of ctc interrupts
191	c	sio1 channel A improper number of ctc interrupts
192	c	sio1 channel B improper number of ctc interrupts
193	d	sio2 channel A improper number of ctc interrupts
194	d	sio2 channel B improper number of ctc interrupts
195	e	sio3 channel A improper number of ctc interrupts
196	e	sio3 channel B improper number of ctc interrupts
197	b	sio0 channel A dma address register not 0xff
198	b	sio0 channel B dma address register not 0xff
199	c	sio1 channel A dma address register not 0xff
19a	c	sio1 channel B dma address register not 0xff
19b	d	sio2 channel A dma address register not 0xff
19c	d	sio2 channel B dma address register not 0xff
19d	e	sio3 channel A dma address register not 0xff
19e	e	sio3 channel B dma address register not 0xff
1a0	10	dma sio 0/1 latch test sio timeout
1a1	10	dma sio 2/3 latch test sio timeout
1a2	10	dma sio 0/1 latch test incorrect data read
1a3	10	dma sio 2/3 latch test incorrect data read
1a4	10	dma pio latch test failed
1a7	12	multibus address test of main processor memory failed
1a8	12	multibus sliding bit test of main processor memory failed
1a9	8	sio0 channel A generated incorrect number of interrupts
1aa	8	sio0 channel B generated incorrect number of interrupts
1ab	8	sio1 channel A generated incorrect number of interrupts
1ac	8	sio1 channel B generated incorrect number of interrupts
1ad	8	sio2 channel A generated incorrect number of interrupts
1ae	8	sio2 channel B generated incorrect number of interrupts
1af	8	sio3 channel A generated incorrect number of interrupts
1b0	8	sio3 channel B generated incorrect number of interrupts
1b5	2	parity error address test (integer)
1b6	2	parity error address test (byte)
1b7	3	parity error sliding bit test
1b8	13	parity error not received
1b9	15	character not received on pio dma loopback

1ba	0	phase 0 memory test failed

The following are normally fatal errors that result in the selftest being aborted. If diagnostic error reporting is enabled (switch 7 is ON on CPU board) an attempt is made to send the error to the host processor.

Number	*Test*	*Meaning*
1f	0	extended instruction trap
1f	1	privilege instruction trap
1f	2	system call trap
1f	3	parity error
1f	4	non vectored interrupt
1f	5	power fail

LED Error Indications

The ICP's LEDs can indicate certain failures that may not be able to be reported to the CPU's console. These LED patterns are shown below. Note that in phase 1 some patterns are used twice, once in constant mode to indicate one error, and once in blinking mode to indicate another error.

Phase 0 LED Patterns

If the test halts prior to completion of phase 0 then the LEDs will not flash.

LED Display	*Hex Value*	*Meaning*
□□●□	2	parity error
●□●□	a	failed sliding bit test or address test if diagnostic switch 0 off
●●●□	e	hung during parity initialization
□●●●	7	hung while verifying parity set
□●●□	6	waiting for multibus to unlock or got multibus and no device to give xack (turn switch 2 off if hung on bus)
●●□□	c	failed memory test
●●●●	f flashing	error occurred during phase 0 (failure written to host processor if switch 2 = on)

Phase 1 LED Patterns

The LED descriptions shown below are true only if DIP switch position 1 (relative to 0 — 7) is set to ON.

LED Value (hex)	*Meaning*
f	If on for a while then running selftest with no failures or selftest wandered into left field or waiting for multibus to unlock to write test results (if diagnostic switch 1 is on)
f flashing	Fatal error occurred during selftest (failure written to host processor if switch 1 is on)
1,3,4,5,6,a b,c,d,e	Last phase that failed in LEDs and selftest proceeding or waiting for multibus to unlock if switch 1 is set
1,3,4,5,6,a b,c,d,e blinking	Last phase that failed flashing in LEDs and selftest is complete. Error information sent to host processor if switch 1 is set
9	Error during communication with host processor
9 blinking	Test done and last error was during communication with host processor

NOTE

If the host system is a z8000 or a 68k processor before 60 -00100 rev z then diagnostic switch 2 must be set ON.

PLEXUS COMPUTERS

Reader Comment Form

Company Name :

Your Name (Optional) :

Manual Name : **Plexus P/35 User´s Manual**

Publication Number : **98-05043.6 Rev.A**

Please let us know if anything in this manual is unclear, incomplete, or inaccurate.

1. Should any information be included or removed?

____

____

____

____

2. Please specify the page and nature of any error(s) found in this document.

____

____

____

____

3. Other Comments

____

____

____

____

Please mail this form to :

Technical Publications Department
Plexus Computers, Inc.
3833 North First St.
San Jose, CA 95134