Chapter 10

Using The Command (PROM) Monitor

This chapter describes the Command (PROM) Monitor programs, which control the boot environment for all Silicon Graphics workstations or servers. With the Command Monitor, you can boot and operate the CPU under controlled conditions, run the CPU in Command Monitor mode, and load programs (for example, the operating system kernel, */unix* or special debugging and execution versions of the kernel).

The *prom*(1) reference page, accessible with the command:

```
man prom
```

contains valuable information about the PROM monitor on your system.

PROM stands for Programmable Read-Only Memory. Most PROM chips are placed in your computer at the factory with software programmed into them that allows the CPU to boot and allows you to perform system administration and software installations. The PROMs are not part of your disk or your operating system; they are the lowest level of access available for your system. You cannot erase them or bypass them.

Since PROMs are not normally changed after the manufacture of the system, newly added features will not be present on older systems. Some systems have PROM firmware that responds to new programming when the operating system is updated. See your hardware *Owner's Guide* for more information on these types of PROMs.

Note that there are numerous minor differences between machines, and you should refer to your *Owner's Guide* and Release Notes for information specific to your machine.

Newer systems use a PROM called the ARCS prom. ARCS stands for Advanced Risc Computing Standard. This PROM provides a graphical interface and allows the use of the mouse to control booting and execution. ARCS systems allow the use of the keyboard, and the older key syntaxes have been retained for compatibility. Systems that use the ARCS prom include the Indigo R4K, Indy, Indigo^2, CHALLENGE, Onyx, some Crimson systems, and others.

This chapter contains information on the following topics:

- Basic instruction on entering the Command Monitor. See “How to Enter the Command (PROM) Monitor” on page 152.
- A summary of the commands available through the general Command Monitor. See “Summary of Command Monitor Commands” on page 154.
- How to get help while using the Command Monitor. See “Getting Help in the Command Monitor” on page 156.
- Instructions for convenient use of the Command Monitor. See “Using Command Monitor Commands” on page 156 and “Running the Command Monitor” on page 159.
- Instructions for manipulating the Command Monitor Environment. See “The Command Monitor Environment” on page 162.
- Instructions for booting programs from the Command Monitor. See “Booting a Program from the Command Monitor” on page 170.

How to Enter the Command (PROM) Monitor

To get into the Command Monitor on most machines, follow these steps:

1. Reboot the system with the *reboot* command, or if it is already switched off, turn it on.

   On server systems without graphics capability, you see the following prompt:

   ```
   Starting up the system....
   ```

   ```
   To perform system maintenance instead, press <Esc>
   ```

   On systems with graphics, you see similar messages, displayed on your screen as shown in Figure 10-1.

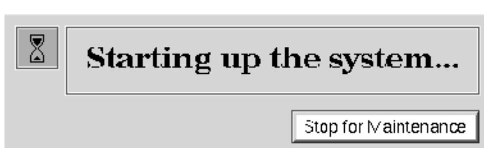

Figure 10-1 ARCS System Startup Message

The procedures are substantially the same for graphical or text usage, except that you need not press the Escape key, you can use your mouse cursor to click a button labeled *Stop for Maintenance*.

2. Press the **<Esc>** key or click the button. You see the following menu, or a similar menu:

```
System Maintenance Menu
1   Start System
2   Install System Software
3   Run Diagnostics
4   Recover System
5   Enter Command Monitor
6   Select Keyboard Layout
```

The menu items have the following effects:

Start System
: This option starts the default operating system.

Install System Software
: This option brings up the standalone version of *inst*(1M). See *IRIX Admin: Software Installation and Licensing* for further information on this option. If your system has the ARCS PROM, it allows you to interactively select the type of device you will use to perform the installation (for example, tape drive, network connection, or CD-ROM drive) and then select the specific device from those of the specified type.

Run Diagnostics
: This option runs a diagnostic program on your system hardware. See your system *Owner's Guide* for further information on this option.

Recover System
: This option allows you to read a previously made system backup onto the system disk. See *IRIX Admin: Backup, Security, and Accounting* for further information on this option. If your system has the ARCS PROM, it allows you to interactively select the type of device you will use to perform the recovery (for example, tape drive, network connection, or CD-ROM drive) and then select the specific device from those of the specified type.

Enter Command Monitor
: This option allows you to enter the Command Monitor program, described in this chapter.

Select Keyboard Layout
This option allows you to select from several different keyboard layouts for common languages.

3. Enter the numeral 5 and press <Return> or click on the appropriate button. You see the Command Monitor prompt:

   ```
   >>
   ```

   You have entered the Command Monitor.

Summary of Command Monitor Commands

Table 4-1 summarizes the Command Monitor commands and gives each command's syntax.

Table 10-1 Command Monitor Command Summary

Command	Description	Syntax
auto	Boots default operating system (no arguments). This has the same effect as selecting Start System from the PROM Monitor initial menu.	**auto**
boot	Boots the named file with the given arguments.	**boot** [**-f**][**-n**] *pathname*
date	Displays or sets the date and time.	**date [***mmddhhmm***[***ccyy***|***yy***] [.***ss***]]**
eaddr	Prints the ethernet address of the built-in ethernet controller on this system.	**eaddr**
exit	leave Command Monitor and return to the PROM menu.	**exit**
help	Prints a Command Monitor command summary	**help** [*command*] **?** [*command*]

Table 10-1 (continued) Command Monitor Command Summary

Command	Description	Syntax
hinv	Prints an inventory of known hardware on the system. Some optional boards may not be known to the PROM monitor.	**hinv**
init	Partially restarts the Command Monitor noting changed environment variables.	**init**
ls	List files on a specified device.	**ls** *devicename*
off	Turns off power to the system.	**off**
passwd	Sets PROM password	**passwd**
pathname	Given a valid file pathname, the system attempts to find and execute any program found at that path.	***pathname***
printenv	Displays the current environment variables	**printenv** [*env_var_list*]
resetenv	Resets all environment variables to default	**resetenv**
resetpw	Resets the PROM password to null (no password required).	**resetpw**
setenv	Sets environment variables. Using the *-p* flag makes the variable setting persistent, that is, the setting will remain through reboot cycles.	**setenv** [**-p**] *variable value*
single	Boots the system into single user mode.	**single**
unsetenv	Unsets an environment variable	**unsetenv** *variable*
version	Displays Command Monitor version	**version**

Getting Help in the Command Monitor

The question mark (**?**) command displays a short description of a specified command. If you do not specify a command, the **?** command displays a summary of all Command Monitor commands. To get help, type either **help** or a question mark (**?**).

help [*command*]

? [*command*]

Using Command Monitor Commands

The following sections cover these subjects:

- The command syntax notation that this chapter uses
- The function of the commands listed in Table 10-1

Using the Command Line Editor in the Command Monitor

You can edit on the command line by using the commands shown in Table 4-2.

Table 10-2 Command Monitor Command Line Editor

Command	Description
<ctrl-h>, **<del>**,	Deletes previous character
<ctrl-u>	Deletes entire line; question mark (**?**) prompts for corrected line
<ctrl-c>	If a command is executing, kills current command
!!	Repeat the last command

Syntax of Command Monitor Commands

The Command Monitor command syntax is designed to resemble the syntax of commands used with the IRIX operating system. This chapter uses IRIX notation for command descriptions:

- **`Boldface`** words are literals. Type them as they are shown.
- Square brackets ([]) surrounding an argument means that the argument is optional.
- Vertical lines (|) separating arguments mean that you can specify only one optional argument within a set of brackets.
- *file* means that you must specify a file name. A file name includes a device specification as described in "Syntax of Command Monitor File Names" on page 157.

Syntax of Command Monitor File Names

When you specify file names for Command Monitor commands, use this syntax:

device`([`*cntrlr*`, [`*unit*`[,`*partition*`]]])`*file*

- *device* specifies a device driver name known to the PROM.
- *cntrlr* specifies a controller number for devices that may have multiple controllers.
- *unit* specifies a unit number on the specified controller.
- *partition* specifies a partition number within a unit.
- *file* specifies a pathname for the file to be accessed.

If you do not specify *cntrlr, unit*, and *partition*, they default to zero. The notation shows that you can specify only a *cntrlr*, a *cntrlr* and *unit*, or all three variables. The commas are significant as place markers. For example, the root partition (partition 0) on a single SCSI disk system is shown as:

```
dksc(0,1,0)
```

where:

- dksc indicates the SCSI driver
- The first 0 indicates SCSI controller 0

- The 1 indicates drive number 1 on SCSI controller 0
- The final 0 indicates partition 0 (root partition) on drive 1 on SCSI controller 0.

The */usr* partition (partition 3) on the same disk would be written as:

```
dksc(0,1,3)
```

The Command Monitor defines the devices shown in Table 4-3.

Table 10-3 Device Names for Command Monitor Commands

Device Name	Description
dkip	the ESDI disk controller (ips in IRIX)
dksc	the SCSI disk controller (dks in IRIX)
tpsc	the SCSI tape controller (tps in IRIX)
xyl	the SMD disk controller (xyl in IRIX)
ipi	the IPI disk controller (ipi in IRIX)
tty	CPU board duart
tty(0)	the local console
tty(1)	the remote console
gfx	the graphics console
console	the ‘‘pseudo console’’ which may be one of gfx(0), tty(0), or tty(1).
bootp	Ethernet controller using bootp and TFTP protocols
tpqic	the quarter-inch QIC02 tape drive

The PROM device notation is different from IRIX device notation. Certain environment variables (such as *root* and *swap*) are passed to higher level programs, and often require IRIX notation for the */dev* device name. For example, in PROM notation, an ESDI disk partition most commonly used for swap is written:

```
dkip(0,0,1)
```

In IRIX notation, the same disk is:

```
ips0d0s1
```

Syntax of ARCS PROM File Names

Systems that use the ARCS prom (including Indy, Indigo^2, Indigo R4K, CHALLENGE, Onyx, and others) use a slightly different syntax for specifying pathnames and disk partitions.

ARCS pathnames use the same syntax as the hardware inventory. The pathnames are written as a series of "type(unit)" components that parallel the hardware inventory format.

Old style pathnames are automatically converted to new style pathnames, so the old names can still be used. The PROM will match the first device described by the pathname, so full pathnames are not always required. Some examples of common pathnames are shown in Table 10-4.

Table 10-4 ARCS File Names

ARCS Naming Convention	Pathname or Device
scsi(0)disk(1)partition(1)	dksc(0,1,1)
disk(1)part(1)	dksc(0,1,1)
scsi(0)cdrom(5)partition(7)	dksc(0,5,7)
network(0)bootp()*host*:*file*	bootp()*host*:*file*
serial(0)	first serial port
keyboard()	graphics keyboard
video()	graphics display

Running the Command Monitor

This section describes the commands that you use to run the Command Monitor. The Command Monitor accepts the commands listed in Table 10-1, “ Command Monitor Command Summary,” on page 154.

Reinitializing the Processor from the Command Monitor

The *init* command reinitializes the processor from PROM memory, and returns you to the monitor program.

Setting a PROM Password

Your system has a facility that allows you to require a password from users who attempt to gain access to the Command Monitor. To set the PROM password, perform the following steps:

1. Select option 5 from the System Maintenance Menu to enter the Command Monitor. You see the Command Monitor prompt:

   ```
   Command Monitor. Type "exit" to return to the menu.
   >>
   ```

2. Enter the command:

   `help`

3. Issue the *passwd* command:

   `passwd`

   You see the prompt:

   ```
   Enter new password:
   ```

4. Enter the password you want for your machine and press **`<Return>`**. You see the prompt:

   ```
   Confirm new password:
   ```

   Enter the password again, exactly as you typed it before. If you typed the password the same as the first time, you next see the Command Monitor prompt again. If you made a mistake, the system prints an error message and you must begin again. If you see no error message, your password is now set. Whenever you access the Command Monitor, you will be required to enter this password.

It is very important that you choose and enter your password carefully, because if it is entered incorrectly or forgotten, you may have to remove a jumper on the CPU board of your system. This procedure is different for each system type, and is described in your *Owner's Guide*. Some systems, though, allow you to reset the PROM password from IRIX by logging in as *root* and issuing the following command:

```
nvram passwd_key ""
```

The quotation marks with no characters or space between them are essential to remove the PROM password. You must be *root* to perform this operation.

The *resetpw* command within the Command Monitor also resets the PROM password.

Copying Hard Disks From the Command Monitor

You can copy a hard disk onto another hard disk easily through the Command Monitor. You may want to do this to create a backup disk in case of failure, or perhaps you have a specific software setup that you wish to copy for a new system. In order for this procedure to work correctly, and for the new disk to be useful, the disks must be of identical size and manufacture. Also, the system that is to receive the new disk must use the same CPU and graphics board set that the existing system uses, because the kernel is custom configured for CPU and graphics type, and the kernel will be copied exactly to the new disk.

Follow these steps:

1. Bring your system down and install the new disk in the space provided for an additional disk. Select a SCSI device number that is not currently in use for the new disk. For this example, we will use device 2 on SCSI controller 0 (the integral SCSI controller) for the new disk, and device 1 on SCSI controller 0 for the disk to be copied.
2. Boot the system to the System Maintenance Menu.
3. Select option 5 from the System Maintenance Menu. You will see the Command Monitor prompt:

   ```
   >>
   ```
4. Give the *sash* or *boot* command to load the sash:

   ```
   sash
   ```
5. From the sash prompt, give the command:

   ```
   cp -b 128k disk(1)part(10) disk(2)part(10)
   ```

You see a "read error" message when the copy is complete. This is the normal message that tells you that the copying software has read the entire disk. If you see a "write error" message, there was an error copying the disk and you will probably have to start over or install the disk by more conventional means.

The Command Monitor Environment

The Command Monitor maintains an environment, which is a list of variable names and corresponding values (the values are actually text strings). These *environment variables* contain information that the Command Monitor either uses itself or passes to booted programs. The system stores some environment variables—those that are important and unlikely to change frequently—in non-volatile RAM (nvram). If you turn off power to the machine or press the **Reset** button, the system remembers these variables. When you change the setting of these variables using the *setenv* command, the PROM code automatically stores the new values in non-volatile RAM.

You can also use the */etc/nvram* command to set or print the values of non-volatile RAM variables on your system. For complete information on the *nvram* command, see the *nvram*(1) reference page.

Table 10-5, " Variables Stored in Non-volatile RAM," on page 163 shows a list of the environment variables that the system stores in non-volatile RAM.

The ARCS Prom defines some variables not found in older PROMS, and so an additional list is provided in Table 10-7.

Several environment variables also exist that affect IRIX's operation. These are not stored in non-volatile RAM, but they do affect the operation of the PROM and of IRIX. See Table 10-6.

Table 10-5 lists non-volatile RAM variables:

Table 10-5 Variables Stored in Non-volatile RAM

Variable	Description
netaddr	Specifies the local network address for booting across the Ethernet. See the *bootp* protocol.
dbaud	Specifies the diagnostics console baud rate. You can change it by setting this variable (acceptable rates include 75, 110, 134, 150, 300, 600, 1200, 2400, 4800, 9600, and 19200), or by pressing the **<Break>** key. IRIS uses the *dbaud* rate for the diagnostics console during the entire system start-up. Pressing the **<Break>** key changes the baud rate only temporarily; the baud rate reverts to the value specified in *dbaud* or *rbaud* when you press the reset switch or issue an *init* command.
rbaud	Specifies the remote console baud rate. The list of acceptable baud rates is the same as for *dbaud*, above.
bootfile	Specifies the name of the file to use for autobooting, normally a stand-alone shell (sash). This variable is valid for pre-ARCS PROMs only. ARCS proms store this information in the *OSLoader* variable.

Table 10-5 (continued) Variables Stored in Non-volatile RAM

Variable	Description
bootmode	Specifies the type of boot in pre-ARCS PROMs. ARCS PROMs store this information in the *AutoLoad* variable. The options have these meanings: c - performs a complete cold autoboot, using the file pointed to by the bootfile variable to boot the kernel; boots sash, then boots kernel; runs power-on diagnostics. m - (default) goes straight to the Command Monitor; clears memory; runs power-on diagnostics. d - go straight to the Command Monitor; do not clear memory; do not run power-on diagnostics (on IRIS-4D 100, 200 and 300 series systems, this has the same effect as *bootmode m*).
boottune	Selects the boot music string. A value of *0* randomizes the selection each time. *1* is the default value. (Supported only on Power Indigo^2)
autopower	Allows systems with software power control to automatically reset after a power failure if set to *y*.
console	Specifies which console to use. The options have these meanings: G - graphics console with the Silicon Graphics, Inc., logo in the upper left corner g - (default) graphics console without the Silicon Graphics logo

Table 10-5 (continued) Variables Stored in Non-volatile RAM

Variable	Description
keybd	Specifies the type of keyboard used. The default is ''df.'' Available settings depend on the exact PROM revision, but may include some or all of: USA, DEU, FRA, ITA, DNK, ESP, CHE-D, SWE, FIN, GBR, BEL, NOR, PRT, CHE-F. or US, DE, FR, IT, DK, ES, deCH, SE, FI, GB, BE, NO, PT, frCH on systems with the keyboard layout selector. On some systems, JP is also acceptable to specify a Japanese keyboard.
diskless	Specifies that the system is diskless and must be booted over the network. On ARCS systems, diskless system environment parameters should be set as follows: **`diskless=1`** **`SystemPartition=bootp()`***host:/path* **`OSLoader=`***kernelname*
monitor	Specifies the monitor resolution on Indy systems when an unrecognized brand of monitor is used. Set this variable to *h* or *H* to specify a high resolution monitor, the default is a low resolution monitor.
nogfxkeybd	Specifies that the keyboard is not required to be connected if set to *1*.
notape	Specifies that no tape drive is attached to the system. If a tape drive is attached to the system, this variable must be set to *1* (true) in order to access a tape drive on another system on the network.
volume	Specifies the system speaker volume numerically.

Table 10-5 (continued) Variables Stored in Non-volatile RAM

Variable	Description
pagecolor	Specifies the background color of the textport using a set of 6 hexadecimal RGB values.
prompoweroff	On Indy systems only, this variable specifies that the system should return to the PROM monitor before powering down on shutdown if set to *y*.
rebound	Specifies that the system should automatically reboot after a kernel panic if set to *y*.
sgilogo	Specifies that the Silicon Graphics logo and related information will be displayed on the PROM monitor graphical screen if set to *y*.
diagmode	Specifies the mode of power-on diagnostics. If set to *v*, then diagnostics are verbose and extensive. .

Table 10-6 lists Command Monitor environment variables that directly affect the operating system. Note that these variables are not stored in non-volatile RAM and are discarded if the machine is powered down.

Table 10-6 Environment Variables That Affect the IRIX Operating System

Variable	Description
showconfig	Prints extra information as IRIX boots. If set through *setenv*, its value must be *istrue*.
initstate	Passed to IRIX, where it overrides the initdefault line in */etc/inittab*. Permitted values are s and the numbers 0-6. See *init*(1M).

Table 10-6 (continued) Environment Variables That Affect the IRIX Operating System

Variable	Description
swap	Specifies in IRIX notation the swap partition to use. If not set, it defaults to the partition configured into the operating system, which is normally partition 1 on the drive specified by the root environment variable.
path	Specifies a list of device prefixes that tell the Command Monitor where to look for a file, if no device is specified.
verbose	Tells the system to display detailed error messages.

When you boot a program from the Command Monitor, it passes the current settings of all the environment variables to the booted program.

The environment variables specific to ARCS Proms are described in Table 10-7.

Table 10-7 ARCS PROM Environment Variables

Variable	Description
ConsoleIn/ConsoleOut	These variables are set automatically at system startup.
OSLoadPartition	The disk partition where the operating system kernel is located. This is also used as the default root partition and is set automatically at system startup.
OSLoader	The operating system loading program. By default, this is SASH (The Stand-Alone Shell). This is set automatically at system startup.
SystemPartition	The disk partition where the operating system loading program is found. This is set automatically at system startup.

Table 10-7 ARCS PROM Environment Variables

Variable	Description
OSLoadFilename	The file name of the operating system kernel. By default, this is */unix*. This variable is automatically set at system startup.
OSLoadOptions	This variable specifies options to the boot command used to load the Operating System. For more information on boot options, see "Booting a Program from the Command Monitor" on page 170.
AutoLoad	This variable specifies whether the operating system will boot automatically after a reset or power cycle. This variable supercedes *bootmode* and can be set to *yes* or *no*.

Displaying the Current Environment Variables

The *printenv* command displays the Command Monitor's current environment variables.

printenv [*env_var_list*]

To change (reset) the variables, see the next section.

Changing Environment Variables

The *setenv* command changes the values of existing environment variables or creates new environment variables.

setenv *env_var string*

env_var is the variable you're setting, and *string* is the value you assign to that variable. To see the current monitor settings, use *printenv.*

When you use *setenv* to change the value of one of the stored environment variables in Table 10-5, the system automatically saves the new value in non-volatile RAM. You do not need to re-enter the change the next time the machine is turned off and then on again.

Setting the Keyboard Variable

If the *keybd* variable is set to anything but the default *df*, the appropriate keyboard translation table is loaded from the volume header of the hard disk. If the table is missing or unable to load, then the default table stored in the PROMs is used. The *keybd* variable can be set to any value, but the keyboard translation table should be loaded from the volume header on the hard disk. This variable overrides the normal system mechanism for determining the kind of keyboard installed in the system. You should not change this variable unless you are performing keyboard diagnostics. Table 10-8 lists *keybd* variables suggested for international keyboards:

Table 10-8 *keybd* Variables for International Keyboards

Variable	Description
BEL or BE	Belgian
DNK or DK	Danish
DEU or DE	German
DF	The default
FRA or FR	French
FIN or FI	Finnish
ITA or IT	Italian
JP	Japanese
NOR or NO	Norwegian
PRT or PT	Portuguese
CHE-F or freCH	Swiss-French
CHE-D or deCH	Swiss-German
ESP or ES	Spanish
SE or SWE	Swedish
GB or GBR	United Kingdom (Great Britain)
US or USA	United States (available on all models)

Removing Environment Variables

The *unsetenv* command removes the definition of an environment variable.

unsetenv *env_var*

env_var is the variable whose definition you are removing (see *setenv*, above). Note that variables stored in non-volatile RAM cannot be unset.

Booting a Program from the Command Monitor

This section describes each Command Monitor boot command and shows you how to use it. When you reboot or press the **Reset** button, you start up the Command Monitor. Do not press the **Reset** button under normal circumstances, that is, when the workstation is running IRIX.

Booting The Default File

The *auto* command reboots the operating system. It uses the default boot file as though you were powering up the CPU. At the Command Monitor prompt (>>), type:

auto

The PROM's environment variable *bootfile* specifies the default boot file. In addition, you must set the environment variable *root* to the disk partition that IRIX uses as its root file system. The *auto* command assumes that the desired image of IRIX resides on the partition specified by *root* of the drive specified in the environment variable *bootfile*.

The *bootfile* name can contain no more than 14 characters. To select a different boot file, see "Changing Environment Variables" on page 168.

Booting a Specific Program

The *boot* command starts the system when you want to use a specific boot program and give optional arguments to that program. The syntax of the *boot* command is:

boot [**-f** *program*] [**-n**] [*args*]

-f specifies the program you want to boot. The program name must contain fewer than 20 characters. If you do not specify this option, the environment variable *bootfile* specifies the default program. *boot* normally loads *sash*.

When you specify a program, you can include a device specification. If you don't, the Command Monitor uses the device specifications in the environment variable *path*. The Command Monitor tries in turn each device that you specify in *path*, until it finds the program you request, or until it has tried all the devices listed in *path*.

- **-n** means no go: it loads the specified program, but does not transfer control to it. Instead, **-n** returns you to the Command Monitor command environment.
- *args* are variables that the Command Monitor passes to the program you're booting. For an *arg* that starts with a hyphen (-), you must prepend an additional hyphen so that the Command Monitor doesn't think that the argument is intended for itself. The Command Monitor removes the extra hyphen before it passes the argument to the booted program. For more information, see "Booting the Standalone Shell" on page 171.

For example, to boot the disk formatter/exerciser program (*fx*) from the cartridge tape drive, use this command:

```
boot –f SCSI(0)tape(7)partition(0)fx
```

Without any arguments, *boot* loads the program specified in *bootfile*.

Booting the Standalone Shell

The Command Monitor has been designed to keep it independent of operating systems and as small as possible. Therefore, the Command Monitor cannot directly boot files residing in IRIX or other operating system file trees. However, the Command Monitor does provide a two-level boot mechanism that lets it load an intermediary program that does understand file systems; this program can then find and load the desired boot file. The program is called the *standalone shell*, and is referred to as *sash*. *sash* is a reconfigured and expanded version of the Command Monitor program, and includes the modules needed to handle operating system file structures. It also has enhanced knowledge about devices.

After the system software is installed, a copy of *sash* is located in the volume header of the first disk. The header contains a very simple file structure that the Command Monitor understands. You can also boot *sash* from tape or across the network if need be. To boot *sash* from your disk, shut down the system, and when you see the message:

```
Starting up the system...
To perform system maintenance instead, press Esc
```

Press the escape key. You may have to enter your system's Command Monitor password, if your system has one. Next, you see a menu similar to the following:

```
System Maintenance Menu
      (1) Start System
      (2) Install System Software
      (3) Run Diagnostics
      (4) Recover System
      (5) Enter Command Monitor
```

Select option 5, "Enter Command Monitor" from the System Maintenance Menu. You see the following message and prompt:

```
Command Monitor. Type "exit" to return to the menu.
>>
```

To boot the standalone shell (*sash*), enter the command:

`boot -f sash`

sash operates in interactive command mode. You see the sash prompt:

```
sash:
```

To use the multi-level boot feature, set the PROM environment variable *bootfile* to refer to a specific copy of *sash*. In normal configurations, setting *bootfile* to *dkip(0,0,8)sash* tells the Command Monitor to load *sash* from the ESDI disk controller 0, disk unit 0, partition 8 (the volume header). Use this syntax:

ESDI drives	**`setenv bootfile "dkip(0)disk(0)partition(8)sash"`**
SCSI drives	**`setenv bootfile "scsi(0)disk(1)partition(8)sash"`**
SMD drives	**`setenv bootfile "xyl(0)disk(0)partition(8)sash"`**
IPI drives	**`setenv bootfile "ipi(0)disk(0)partition(8)sash"`**

Then issue a boot command, as in this example for an ESDI drive:

`boot dkip()unix initstate=s`

The following actions take place:

- *boot* loads *dkip(0)disk(0)partition(8)sash*, as specified by *bootfile*, since the boot command doesn't contain a **-f** argument. (A **-f** argument would override the default specified by *bootfile*.)
- *sash* gets two arguments: **dkip()unix** and **initstate=s**, which brings the IRIS up in single-user mode. (Note that the Command Monitor removes the leading hyphen [-] from any argument, so if you use the next layer of software, and need an argument with a leading hyphen, you should put two hyphens in front of it.)
- *sash* loads the file specified by the first argument (**dkip()unix**) and passes the next argument to that file.

Do not issue the *auto* command from *sash* with the *bootfile* set as shown above. If you do, the system tries to boot *sash* over itself and will exit with an error.

To be able to use the *auto* command from *sash*, set *bootfile* to refer to the kernel, for example, *dkip()unix*. Even better, return to the PROM level to use the *auto* command.

Booting across the Network

At the heart of the operation of diskless workstations is the *bootp* protocol. The *bootp* protocol is a DARPA standard protocol supported on all Silicon Graphics servers and workstations. One of the devices that the Command Monitor can use for booting is the network. Silicon Graphics provides a TCP/IP boot protocol that lets you boot files that reside on another host in the network, if the other host supports the booting protocol. The network booting protocol is the *bootp* protocol. It is a datagram protocol that uses the User Datagram Protocol (UDP) of TCP/IP to transfer files across the Ethernet network.

To boot across the network, you must first determine the Internet address of the machine you want to boot. The Internet address is a number assigned by the network administrator of the network to which the system is attached. The format of the number is four decimal numbers between 0 and 255, separated by periods; for example:

```
192.20.0.2
```

Use the *setenv* command to set the *netaddr* environment variable to this address; for example:

```
setenv netaddr 192.20.0.2
```

Booting across the Network with bootp

Once you have set the *netaddr* environment variable, you can use *bootp* to refer to a remote file by using a file name of the form:

bootp()[*hostname*:] *path*

- *hostname* is the name of the host where the file resides. The specified host must run the *bootp* server daemon, *bootp*. If you omit *hostname*, *bootp* broadcasts to get the file from any of the hosts on the same network as the machine making the request. The first host that answers fills the request. Only hosts that support *bootp* can respond to the request. It is safe to omit the hostname only when you know that the path is unique to a particular host, or when you know that all the copies of the file are interchangeable.

  hostname can be the name of a host on a different Ethernet network from the machine that you are booting, if a *gateway* on the local Ethernet network provides a route to the remote host. The gateway must be an IRIS workstation running a *bootp* server that you have configured to do cross-network forwarding.

  For more information about booting through gateways, see *bootp*(1M). For more information about the */etc/inetd.conf* configuration file, see *inetd*(1M).

- *path* is the pathname of a file on the remote host. For example, this command:

  boot -f bootp()wheeler:/usr/local/boot/unix

  boots the file */usr/local/boot/unix* from the remote host *wheeler*. The command:

  boot -f bootp()/usr/alice/help

  boots the file */usr/alice/help* from any host on the network responding to the *bootp* broadcast request that has a file of that name.

To configure the gateway to permit cross-network forwarding, follow these steps:

1. Log in as **root** or become the superuser by issuing the *su* command.
2. Edit the file */etc/inetd.conf* on the gateway machine. This file configures the *bootp* server, which is started by the *inetd*(1M) daemon.
3. Change the *bootp* description so that *inetd* invokes *bootp* with the **-f** flag. Find this line:

   ```
   bootp  dgram  udp  wait  root  /usr/etc/bootp  bootp
   ```

   Add the **-f** flag to the final *bootp* on the line:

   ```
   bootp  dgram  udp  wait  root  /usr/etc/bootp  bootp  -f
   ```

4. Change the *tftp* configuration line in one of the following ways:

   Remove the **-s** flag from the argument list for *tftpd*:

   ```
   tftp   dgram  udp   wait  guest   /usr/etc/tftpd  tftpd -s
   ```

   This allows *tftpd* access to all publicly readable directories. If you are concerned about a possible security compromise, you can instead explicitly list the directories to which *tftpd* needs access. In this case, you need to add */usr/etc*:

   ```
   tftp dgram udp wait guest /usr/etc/tftpd tftpd -s /usr/etc
   ```

   See *tftpd*(1M) and *tftp*(1C) for more information.

5. Signal *inetd* to re-read its configuration file.

   ```
   killall -1 inetd
   ```

Booting Across A Larger Network

If you have access to a larger network, and the bootable file you need is sufficiently remote on the network that the *tftp* and *bootp* timeouts and network delays are keeping you from booting successfully, it is possible to use an intermediary host as a *bootp* server.

As an example, consider the following situation. You have a host named *local_host* that needs to boot a kernel found on the remote system *far_host*. But the network is heavily used, resulting in *bootp* and *tftp* timing out before the boot operation can take place. However, a third host, *near_host*, has the optional NFS software and has *automount*(1M) running, allowing access to the files on *far_host*. To boot through this method, perform the following steps:

1. On *near_host*, the system acting as intermediary, log in as *root* and edit the file */etc/inetd.conf*. This file configures the *bootp* server, which is started by the *inetd*(1M) daemon.

2. On *near_host*, change the *bootp* description in the */etc/inetd.conf* file so that *inetd* invokes *bootp* with the *-f* flag. Find this line:

   ```
   bootp  dgram  udp  wait  root  /usr/etc/bootp  bootp
   ```

   Add the *-f* flag to the final *bootp* on the line:

   ```
   bootp  dgram  udp  wait  root  /usr/etc/bootp  bootp  -f
   ```

3. On *near_host*, change the *tftp* configuration line in the */etc/inetd.conf* file in one of the following two ways:

   Remove the *-s /usr/local/boot* string from the argument list for *tftpd*, so that the entry matches the following:

   ```
   tftp   dgram  udp   wait  guest   /usr/etc/tftpd  tftpd
   ```

   This allows *tftpd* access to all publicly readable directories.

   If you are concerned about a possible security compromise, you can instead explicitly list the directories to which *tftpd* needs access. In this case, you need to add */hosts*:

   ```
   tftp dgram udp wait guest /usr/etc/tftpd tftpd -s /hosts
   ```

   See *tftpd*(1M) and *tftp*(1C) for more information.

4. Signal *inetd* to re-read its configuration file.

   killall -1 inetd

5. On *far_host*, the system on the distant subnetwork, use NFS to export the directory containing the needed bootable kernel (in this case, the file is */usr/local/boot/unix*). If you need help exporting a directory, see the *export*(1M) reference page.

6. On *local_host*, the system you are trying to boot, give the command:

   boot -f bootp()near_host:/hosts/far_host/usr/local/boot/unix

   If *bootp* times out, try the command again, as *automount* may require a bit of time to retrieve the files from the remote system.

Booting from a Disk or Other Device

To tell the Command Monitor to load standalone commands from various resources (such as a disk or CD-ROM device), set the *path* environment variable. (See "Changing Environment Variables" on page 168.) Set the *path* variable as follows:

setenv path "*device_name alternate_path*"

For example, issue the following command:

setenv path "dkip(0)disk(0)partition(8)bootp()/altdir/altbootfile"

This causes the Command Monitor to boot the file *dkip(0)disk(0)part(8)altbootfile*. If that file fails, the Command Monitor boots *bootp()/altdir/altbootfile*. If that file also fails, the Command Monitor prints the message "`command not found`". Note that pathnames are separated with spaces. If the device specification is contained within a command or by *bootfile*, the Command Monitor ignores *path*. Only *bootp* or volume headers are understood by the PROM.