

# 9 *System Startup and Shutdown*

Ordinarily, you won't be aware of the complex system startup and shutdown processes, since they happen behind the scenes. Once you enter a startup or shutdown command, the system functions automatically, and you see only the animated icons and the standard prompts. However, it's useful to understand these underlying processes so you can identify and correct problems if they arise.

This chapter describes what happens when you start up and shut down a NeXT computer. It also discusses the system monitor programs, which you can use to get information about the system, to boot and shut down the computer, and to set system parameters.

## Using the System Monitors

Each NeXT computer has two system monitors—the ROM monitor and the NMI (nonmaskable interrupt) mini-monitor. Each monitor has a set of commands that lets you access the system on a basic level. The *ROM monitor* controls the system directly, while the *NMI mini-monitor* is used while the operating system is running. The following sections describe how to use each of these monitors.

### ROM Monitor

The ROM monitor automatically performs system self-test diagnostics and initiates the boot procedure when the system is turned on or restarted. During startup, it also loads the system files from the boot device. Normally, you don't need to use the ROM monitor in your day-to-day work. However, you may want to use the ROM monitor

interactively to perform certain specialized functions, such as setting the hardware password or booting from a nondefault device.

**Note:** Older NeXT computers might have an earlier version of the EPROM chip, which means that the ROM monitor commands may not exactly match the ones discussed here.

You can access the ROM monitor in several ways:

- When the system is first turned on, hold down the Command bar and press the ~ key (without pressing Shift) *immediately* after the `^Testing System^` message is replaced by the `^Loading from disk^` message. (On keyboards with two Command keys, hold down the right Command key and press the ~ key.) The ROM monitor window is displayed containing the prompt `^NeXT>^`.

**Note:** If you're a little slow and the Restart/Power-Off panel appears instead, press the Power key and try again.

- When the system is running, you can access the ROM monitor through the NMI mini-monitor (described later in this chapter). Access the NMI mini-monitor by pressing the ~ key (without pressing Shift) while holding down the Command bar (hold down both Command keys on keyboards that have two). When you see the `^nmi>^` prompt, type **halt** at the prompt. The **halt** command attempts to shut the system down gracefully, saving all files to disk.

**Warning:** Although the **monitor** (or **mon** or **m**) command also takes you to the ROM monitor, it does *not* perform a graceful shutdown. If you power off the computer or reboot after using **m** to reach the ROM monitor, you can cause file system corruption.

- If no default boot device is set, you'll see the ROM monitor prompt when the computer is turned on. From here, you need to enter one of the boot commands (see `^Booting from a Device,^` later in this chapter).

A set of special commands allows you to interact with the ROM monitor. To see the list of monitor commands, type a `^?^` (question mark) at the `^NeXT>^` prompt.

Many of the ROM monitor commands control system test diagnostics and are not discussed in this manual. The following sections describe the ROM monitor commands most commonly used for system administration purposes.

## Inspecting or Modifying Configuration Parameters

You can inspect or change several system parameters with the **p** command. Execute this command by typing **p**

(in lowercase) at the `^NeXT>` prompt. All parameters are stored in nonvolatile RAM and will survive if the power is turned off.

When you type **p**, the ROM monitor begins an interactive session that prompts you for a value for each parameter. If you respond by pressing Return, the parameter remains at the current value.

**Note:** If the hardware password is set, you cannot change the values of any parameters without that password. (See `^Setting the Hardware Password` later in this chapter.)

Example:

```
p
boot command: od? sd                (Make the SCSI drive the default boot device.)
DRAM tests: yes?
perform power-on system test: yes?
    sound out tests: no? yes          (Enable sound tests: system beeps after power-on.)
    SCSI tests: no?
    loop until keypress: no?
    verbose test mode: no?
boot extended diagnostics: no?
serial port A is alternate console: no?
allow any ROM command even if password protected: no?
allow boot from any device even if password protected: no?
allow optical drive #0 eject even if password protected: no?
enable parity checking if parity memory is present: no?
```

## Boot Command

This parameter sets the default boot device. Possible values are:

en	Ethernet (either thin-wire or twisted-pair)
fd	Floppy disk (internal only)
od	Optical disk (internal only)
sd	SCSI disk
tp	Twisted-pair Ethernet

**Note:** The value **en** automatically switches the boot attempt between the thin and twisted-pair Ethernet connectors until the system finds a connection to a functioning network. The thin connector is checked first. If you want to boot from the twisted-pair connector *without* checking the thin connector (for example, for testing purposes), use the value **tp**. Otherwise, use **en**.

Optional flags and device selectors can be appended to the boot command in the format:

*device* *[[ctrl,unit,part)] [file] [flags] [kvars]]*

For information on these arguments, see <sup>a</sup>Booting from a Device<sup>o</sup> later in this chapter. The boot command parameter may be up to 12 characters long.

## DRAM Tests

Because these tests check main memory and take the longest amount of time, they can be enabled separately from the other system test options. A **yes** or **y** response enables them; a **no** or **n** response disables them.

## Power-on System Test

If this parameter is set to **no**, none of the other diagnostic tests are run. If it's set to **yes**, the four additional parameters determine which tests are performed:

- `sound out tests`

This parameter causes the sound tests to be performed.

- `SCSI tests`

This parameter enables more comprehensive testing of the SCSI interface (but only if a SCSI device is installed either internally or externally).

- `loop until keypress`

This parameter runs the diagnostics repeatedly until interrupted.

- `verbose test mode`

This parameter causes the name of each test to be displayed as it's run. Note that this parameter not only displays the test names, but also the system messages that are printed at each step of the boot process. Verbose mode can be extremely useful in diagnosing system problems because these messages can help you pinpoint just where a problem occurs.

## Boot Extended Diagnostics

This parameter allows secondary diagnostics to be booted from the default boot device. When set to **yes** or **y**, a reboot attempts to load the file **diagnostics** from the root directory of the default boot device. This program is not shipped with the system and is for use by NeXT Authorized Service Providers.

## Alternate Console

If the ROM monitor detects that the display monitor or keyboard is disconnected when the computer is turned on, the ROM monitor attempts to communicate with a terminal connected to serial port A if this parameter is set to **yes**.

## Password-protected Commands

The next three parameters determine which ROM monitor commands can be used if the hardware password is set (see the section <sup>a</sup>Setting the Hardware Password<sup>o</sup> later in this chapter):

- `allow any ROM command even if password protected`

If set to **yes**, any ROM monitor command can be executed without entering the hardware password (the one exception is the **P** command). This is useful if you want to set the hardware password without disabling any other ROM commands. Booting in single-user mode always requires the hardware password, if set.

- `allow boot from any device even if password protected`

If set to **yes**, any boot device can be specified with the **b** command without entering the hardware password. This can be used to allow booting from any device while disabling other ROM commands.

- `allow optical drive #0 eject even if password protected`

If set to **yes**, the disk in optical drive 0 (assuming you have one) can be ejected without entering the hardware password. If the computer is installed in a public access area, it may be important to prevent the optical disk from being ejected. In this case, you should leave this parameter set to **no**.

## Parity Checking

The last parameter is used to turn on error checking. If you have installed parity memory, set this parameter to **yes**.

## Booting from a Device

The ROM monitor lets you boot the system according to parameters that you specify. To do this, use the command:

**b**[*device* [(*ctrl*, *unit*, *part*)] [*file*] [*flags*] [*kvars*]]

If you execute **b** with no arguments, the system boots the default kernel file from the default device. If you include the *device* argument, the system boots from that specified device. You can also boot from an alternate partition on that device with *ctrl*, *unit*, and *part*. If you include the *file* argument, the system boots from the kernel specified by *file*. The *flags* and *kvars* (kernel variables) provide additional boot options.

For information on setting the default boot device, see the preceding section, <sup>a</sup>Inspecting or Modifying Configuration Parameters.<sup>o</sup>

Possible values for *device* are:

en	Ethernet (either thin wire or twisted-pair)
fd	Floppy disk (internal only)
od	Optical disk (internal only)
sd	SCSI disk
tp	Twisted-pair Ethernet

The optional device selectors are:

<i>ctrl</i>	Controller number (default 0).
<i>unit</i>	SCSI logical unit number (default 0). Almost always set to 0.
<i>part</i>	Partition number (default 0). These are 0-7, corresponding to partitions a-h.

For SCSI drives, the controller number is the logical device number, not the actual target number of the drive. For example, if the first drive is set to target 0 and the second drive target 5, then controller number 1 refers to the second drive.

Possible values for *flags* are:

-a	Ask for the name of the root device.
-b	Boot without running <b>/etc/rc.boot</b> .
-s	Boot in single-user mode instead of multiuser mode.
-i	Ask for the name of the <b>init</b> program (the default is <b>/etc/init</b> ).
-p	Don't automatically reboot after a system panic.

The kernel variables (*kvars*) useful for system administration are:

rootdev=xxx	Use the device specified by xxx (such as <b>sd1</b> , <b>en0</b> , <b>fd0</b> , <b>od0</b> ) as the root device when obtaining the kernel from a different boot device.
rootrw=1	Initially mount the root file system as read/write. Normally, the root file system is initially mounted as read-only. Then, if <b>fsck</b> indicates the file system is clean, it's remounted as read/write.

You can press any key to stop booting from the Ethernet. To stop booting from any other device, hold down the Command bar and press the ~ key (without pressing Shift). On keyboards with two Command keys, hold down the right one and press the ~ key.

Here are some example boot commands:

b	Boot from the default boot device.
bsd	Boot from the SCSI disk.
ben	Boot from the Ethernet.
bod test	Boot <b>test</b> from the optical disk.
bfd -s	Boot in single-user mode from the internal floppy disk.
ben mach -as	Boot <b>mach</b> in single-user mode from the network, and prompt for the root device.
bsd(1,0,0)	Boot from the second SCSI disk.
bod - rootdev=sd0	Boot, taking the kernel from the optical disk, then use <b>sd0</b> as the source for the root device. The <sup>a</sup> - <sup>o</sup> is required if the kernel name is not given.

## Setting the Hardware Password

You can set a hardware password, which allows you to limit the ROM monitor commands that can be used without first supplying the password. If the hardware password is set, the three parameters described in the earlier section, <sup>a</sup>Inspecting or Modifying Configuration Parameters,<sup>o</sup> determine how the ROM commands will be limited.

**Warning:** If you set the hardware password, protect it carefully. If it should be lost, recovery is a complex procedure.

To set the password, type **P** (in uppercase). Enter the new password (no more than 6 characters) at the prompt <sup>a</sup>New password.<sup>o</sup> You're then prompted to confirm the new password by retyping it.

Once the hardware password is set, you're prompted for the password in these circumstances:

- When booting the system in single-user mode
- When changing the default boot device with the Preferences application
- When issuing a ROM monitor command that is password protected
- When changing the hardware password

You are only required to enter the hardware password once per session. Once you have successfully entered the hardware password, you can execute any protected command without having to supply the hardware password again.

## Displaying Memory Configuration

You can display information about system memory configuration by entering the command **m**. This command displays information about the memory installed in memory sockets 0 through 15.

The output from the **m** command will be something like this:

```
Memory sockets 0 and 1 (front) have 8MB of page mode SIMMs installed
(0x40000000-0x47fe000)
Memory sockets 0 and 1 (back) have no SIMMs installed (0x0-0x0)
Memory sockets 2 and 3 (front) have no SIMMs installed (0x0-0x0)
Memory sockets 2 and 1 (back) have no SIMMs installed (0x0-0x0)
```

The ROM monitor reserves some space in the last SIMM to store its internal information.

## Displaying Error Codes

For debugging purposes, you can display the last two system test error codes recorded in nonvolatile memory. These error codes can be produced during the power-on self test routines. This command can be helpful in tracking down intermittent errors that can't be produced on demand.

To display these error codes, enter:

```
ec
```



For a list of possible error codes, see Appendix E, “System Test Error Codes.”

## Ejecting Optical Disks

You can eject the disk from the internal optical disk drive by entering the following command:

`ej`

The command `eo` has the same effect.

## Ejecting Floppy Diskettes

You can eject a floppy diskette from the internal floppy drive by entering the following command at the ROM monitor prompt:

`ef`

## Resuming Execution

The **continue** (or `c`) command resumes execution at the point where it left off, if you reached the ROM monitor by entering `m` in the NMI mini-monitor. The screen display is not restored when you resume execution, which means that most of the objects on the screen will be invisible until they are redrawn. One way to do this is to drag the (invisible) main menu back and forth across the entire screen, causing all the objects to be redrawn. In general, avoid using `c` from the ROM monitor.

## NMI Mini-Monitor

While a NeXT computer is running, you can produce a nonmaskable interrupt (NMI) that opens the NMI mini-monitor, giving you access to the set of NMI commands. Although the NMI mini-monitor is not part of the ROM monitor, it's similar. The NMI mini-monitor allows low-level inspection and debugging of the operating system, while the ROM monitor allows low-level inspection of the system hardware.

To access the NMI mini-monitor, press the ~ key (without pressing Shift) while holding down both the left Alternate key and the Command bar (on keyboards with two Command keys, hold down both and press the ~

key). You can type this key sequence while in any application; you don't have to open a special window. This displays a window that contains the NMI mini-monitor prompt:

```
nmi>
```

You can also arrive in the NMI mini-monitor window through a system failure. If this happens, you'll see the prompt `^panic>` instead of `^nmi>`. The available commands are the same, although the **continue** command will continue a panic (which is only useful in rare circumstances).

You can display the list of available NMI commands by typing a `^?>` (question mark) at the `^nmi>` or `^panic>` prompts. These commands are described in the following sections.

## Resuming Execution

The **continue** (or **c**) command resumes execution at the point where it left off (that is, the point at which you produced the nonmaskable interrupt). It's not a great idea to use this command after a system failure, since the system will be returned to exactly the point at which it failed. This will undoubtedly result in a repeat failure.

## Exiting to the Debugger

The **gdb** command allows you to exit to the GNU debugger. This allows the system to be debugged using a cooperating computer connected over serial port A.

## Displaying the Kernel Message Buffer

The **msg** command displays the kernel message buffer that contains messages from the system. This is useful because system messages aren't normally displayed on the screen. (If you want such messages to be displayed, choose the Console command from the Tools menu in the Workspace Manager.) This command allows you to examine those messages even if they haven't been displayed. If you enter **msg=*n***, only the last *n* messages will be displayed.

**Important:** This command is especially useful after a system panic, since you can use it to check the kernel messages that occurred during this event. When diagnosing the cause of a system panic, it's important to record what these messages are.

## Exiting to the ROM Monitor

The **monitor** (or **mon** or **m**) command takes you to the ROM monitor. The ROM monitor commands are described earlier in this chapter. Remember that the **monitor** command does not shut down the system gracefully, and does not save files to disk before it exits to the ROM monitor. To halt the system, use the **halt** command (see <sup>a</sup>Halting the System<sup>o</sup> later in this section).

## Rebooting the System

The **reboot** (or **r**) command saves all files to the disk and restarts the system with the last command used to boot the system.

## Halting the System

The **halt** (or **ha**) command saves all files to the disk and halts the system. Unlike the **mon** command, which simply stops the system, **halt** attempts a clean shutdown. In almost all circumstances, it's better to use **halt** rather than **mon** (use **mon** only if **halt** doesn't work).

**Note:** The recommended way to halt the system is to press the Power key.

## Retaining the NMI Window

Normally when you use the NMI **continue** (or **c**) command to resume execution, the NMI window disappears from the screen. If you enter the **stay** (or **s**) command before giving the **continue** command, the NMI window will remain on the screen instead.

**Warning:** The display of other windows may conflict with the display of the NMI window through use of the **stay** command. Also, although **stay** does retain a display of the NMI window and any messages that may appear in it, you may have trouble entering commands into the window.

To remove the NMI window once you've given the **stay** command, you must reboot the computer.

## Enlarging the NMI Window

You can enlarge the NMI window using the **big** command. When you enter this command, you're automatically

returned to the interrupted process, as if you had entered **continue**. Subsequently, when you return to the NMI mini-monitor, the window will be bigger.

## Observing the Boot Process

The boot process on a NeXT computer goes through three phases:

- **Bootstrap**—The ROM monitor loads the Mach kernel into memory, either from a local disk or from the network. If the system is booting from the network, the ROM first broadcasts a request to the network to locate a server, and the first server to respond provides the kernel.
- **Startup**—Internal initialization of the Mach kernel takes place. The system hardware is reset, the kernel data structures are initialized, the kernel threads are started, and the root file system is mounted. Finally, the kernel executes **init**.
- **init**—The **init** program brings up the system in multiuser mode by running initialization (or **rc**) scripts, which start the daemon processes and mount the file systems.

One of the design goals of the booting sequence is to provide autonomy among the three phases. Thus, the kernel needs no information about the device it was loaded from to become fully functional. The **rc** scripts need no information from the kernel to bring the system up in multiuser mode.

Here's a brief description of how the **rc** scripts are used during startup:

1. When the **init** program executes during startup, it first runs the shell script **/etc/rc.boot**. This script performs the most basic system initialization, such as checking the file systems for consistency (if needed) and initializing the network interface.
2. If the system is being brought up in single-user mode (with the **-s** flag), **init** will run an interactive shell (**/bin/sh**) on the console. When this shell exits (or if you're not starting up in single-user mode), **init** will run the shell script **/etc/rc**.
3. The **/etc/rc** script performs some general system startup operations, calls **/etc/rc.swap**, mounts the local file systems, and starts the system daemons.
4. During the mount process, local file systems are mounted according to entries found in **/etc/fstab**. Later,

**/etc/rc** starts NetInfo. Once it starts, NetInfo is used as the database for file system mounting, and **/etc/fstab** is not used again for mounting. To complete the mount process, **/etc/rc** performs a second mount for NFS partitions using the NetInfo database.

5. The **/etc/rc** script also calls **/etc/rc.local**. Use this script to specify local startup procedures.

The **rc** scripts have been designed to work on all system configurations. They operate in conjunction with the file **/etc/hostconfig**, which is modified by SimpleNetworkStarter and HostManager. Typically the only startup files you will modify are **/etc/hostconfig** and **/etc/rc.local**.

The daemon processes started by the **rc** scripts operate in the background to perform routine system tasks. There are numerous daemons controlling many system functions, and starting them is one of the most significant tasks the **rc** scripts perform. For more information, see "The **rc** Scripts" later in this chapter.

The following table lists the daemon processes.

Daemon	Function
mach-task	Kernel threads
kernel idle	Idle loop (runs when nothing else needs to be run)
init	Starts the system initialization procedures
mach_init	Controls system service port inheritance
kern_loader	Loadable device driver server
syslogd	Logs system error and status messages
nmserver	Supports networking of Mach interprocess communication and acts as name server for Mach IPC ports
portmap	Converts RPC program numbers into Internet port numbers
nibindd	Starts NetInfo service; spawns <b>netinfod</b> daemons
netinfod	NetInfo server (one for each domain served)
ypserv	NIS server process
ypbind	NIS client-to-server bind process
lookupd	Communicates with network information services, such as NetInfo, NIS, and DNS

routed	Network daemon for maintaining routes to other networks
biod	Asynchronous block I/O daemons for network file system operation
ntpd	Network time protocol daemon
inetd	Initial Internet daemon
sendmail	Mail daemon
lpd	Line printer daemon
pbs	NeXT pasteboard server
npd	NeXT printer daemon that intercedes between applications and <b>lpd</b> , determining whether to send PostScript directly to the NeXT printer or to be spooled
FaxDaemon	Supports sending and receiving faxes
autonfsmount	Automatic mounting daemon for NFS directories
nfsd	Daemons that handle client file system requests if this host is an NFS server
rpc.mountd	NFS mount request server
bootpd	Configuration server (boot protocol) daemon
rpc.bootparamd	Boot parameters server
snmpd	Simple Network Management Protocol daemon
atalkd	AppleTalk protocol daemon
ashared	AppleShare daemon
nucmessaged	NetWare message daemon
autoNetWaremount	Automatic mounting daemon for NetWare servers
Connector	Allows ISDN connections
named	Domain Name Service (DNS) server
update	Updates disks by running <b>sync</b> every 30 seconds

cron	Executes automated system processes at specified times
------	--

Once the system has started up, **loginwindow** is run. The **loginwindow** program is the login front end for the console of the NeXTSTEP environment. It allows the user to log in by entering a user name and password. Before executing the Workspace Manager program, **loginwindow** reads the user parameters (including those specified with the Preferences application) from various sources. These parameters are put into effect for the duration of the user's login session. It then executes the Workspace Manager as that user.

If there's an account for the user **me** and that account has no password, then the **loginwindow** program doesn't go through the login procedure; it immediately executes the Workspace Manager for **me**.

## Customizing Login and Logout

The **loginwindow** program can be run with optional arguments. These arguments permit you to tailor what **loginwindow** does during login and logout. The arguments are:

- **LoginHook** *prognam*e ⌘ Specifies a program for **loginwindow** to run before a user logs in. You should specify the program *prognam*e with an absolute pathname (beginning with <sup>a/</sup>). The **LoginHook** program is run after the user enters the account name and password in the login window. The specified program won't be run if the default user (**me**) is logged in without being asked for a password.
- **LogoutHook** *prognam*e ⌘ Similar to **LoginHook**, except that the specified program is run after a user has logged out. Like **LoginHook**, the program *prognam*e should be specified by its full pathname. It isn't run when the default user (**me**) is logged out.
- **HostName** *hostname* ⌘ Specifies a name to be displayed in the login window. If the name specified is **localhost**, then the computer's host name is displayed.
- **ImageFile** *filename* ⌘ Provides an alternate file to use as the background of the login window. Normally, **loginwindow** displays the following file as the background for the login window:

`/usr/lib/NextStep/loginwindow.app/English.lproj/nextlogin.tiff`

If you provide a different image file, it should contain an image 650 pixels wide by 230 pixels high. The image should contain two white rectangles, each 156 pixels wide by 23 pixels high: one at location (242,100) within the image, and one at location (242,53) within the image (these are the rectangles where the name and

password are entered). Subject to these restrictions, the rest of the image can contain whatever you want. **IconBuilder**, located in **/NextDeveloper/Apps** on the extended release, can be used to create TIFF image files.

- **DefaultUser** *username* — Designates the default user (if this option isn't used, **me** is the default user). If the default user has no password, **loginwindow** automatically logs that user in without requesting a password when the computer is booted.
- **PowerOffDisabled true** — Prevents users from casually turning off the computer. This option can be useful if the system is providing a shared service (for example, a print server). If this option is set, then pressing the Power key has no effect while the login window is displayed. In addition, if a user presses the Power key while logged in, the user is logged out but the system is not turned off. To turn off the system when this argument is set, exit to the NMI mini-monitor and type **halt** to exit to the ROM monitor. From the ROM monitor, press the Power key, then type **y**.

The options to **loginwindow** can be set in **/etc/ttys**:

1. Log in as **root**.
2. Open **/etc/ttys** and locate the line beginning with `^console^` (although it may appear on two lines, this is actually a single entry containing no Return characters):

```
console    /usr/lib/NextStep/loginwindow      NeXT      on secure
window=/usr/lib/NextStep/WindowServer onoption="/usr/etc/getty std.9600"
```

3. Modify the line to include arguments to **loginwindow**. This example disables the Power key and changes the default user to **george**. Note the required quotes around **loginwindow** and its arguments:

```
console "/usr/lib/NextStep/loginwindow -PowerOffDisabled true -DefaultUser george" NeXT on
secure window=/usr/lib/NextStep/WindowServer onoption="/usr/etc/getty std.9600"
```

4. Reboot the system.

For more information about **loginwindow** and its arguments, see the UNIX manual page for **loginwindow**.

## The rc Scripts



What follows is an explanation of each **rc** script (including **rc.boot**, **rc**, **rc.swap**, **rc.local**, **rc.cdrom**, and **rc.uucp**).

**Note:** The **rc** scripts have been thoroughly annotated. For further information about the initialization process, read the files themselves.

## **/etc/rc.boot**

This script is run by **init** before the system enters single-user mode, and before **rc** is run during a normal multiuser boot. It runs **fsck** to check all file systems specified in the **fstab** file, creates a **swapfile** if one doesn't already exist, creates **mtab** entries for all file systems mounted by the kernel, and initializes the Ethernet network interface with **ifconfig**.

**Note:** An exact copy of this file is provided in **/etc/rc.boot.standard**. If you ever need to return to the release version of **rc.boot** after it has been modified, make a copy of **rc.boot.standard**.

## **/etc/rc**

This script is run by **init** when the system enters multiuser mode. It performs general startup functions such as mounting file systems, starting the system log daemon, running **rc.local**, starting **cron** and **update**, and turning on accounting.

**Note:** An exact copy of this file is provided in **/etc/rc.standard**. If you ever need to return to the release version of **rc** after it has been modified, make a copy of **rc.standard**.

## **/etc/rc.swap**

The script **/etc/rc.swap** is run by **/etc/rc**. If a swap disk is present and properly configured, it will be mounted on **/private/swapdisk** and used for paging and temporary file space.

## **/etc/rc.local**

This script is run by **/etc/rc**. It's used to start locally installed software (that is, software that is not provided by NeXT or not supported by NeXT). If you have local daemons of your own to start, place them in this script. For example, you might add a command to run **rc.uucp**.

## **/etc/rc.cdrom**

This script is run by **/etc/rc** if the system is being booted from a CD-ROM. Booting from CD-ROM is used to install system software on the hard disk.

## **/etc/rc.uucp**

This script is provided as a convenience for those who want to run UUCP. This script should be executed by **rc.local**. Note that UUCP is not supported in this release (it's provided with no guarantees). However, you can refer to Chapter 12, "Using UUCP," for more information on UUCP.

# **Shutting Down the System**

You can shut down your system in several ways. The preferred way to shut down is to use the Power key on your keyboard. However, you can also shut down using a UNIX command or the NMI mini-monitor. If possible, any shutdown procedure you select should be "clean," with system processes halted in an orderly fashion and all files saved to disk.

After a clean shutdown, the file system is in a consistent state—it can account for all of its data blocks. By contrast, after a "dirty" shutdown the file system may be in an inconsistent state and (rarely) data can be lost. Most of the time, the file system can recover from a dirty shutdown without loss of data. You can use the **fsck**

program to check for file system inconsistencies and repair them if they are minor (for details, see <sup>a</sup>Troubleshooting,<sup>o</sup> later in this chapter, and the UNIX manual page for **fsck**).

## Shutting Down with the Power Key

To shut down your system using normal procedures, first quit any applications to make sure any open files are saved.

To power off, simply press the Power key on the keyboard. A panel appears asking you if you really want to turn off the computer. Click Turn it Off. This is a clean shutdown.

## Shutting Down with a UNIX Command

To shut down the system using a UNIX command:

1. Open a shell window and use **su** to gain **root** access.
2. Enter either of the following commands:

```
shutdown  
or  
halt
```

The **shutdown** command takes you to single-user mode, while the **halt** command takes you to the ROM monitor.

3. Press the Power key.
4. A prompt appears asking <sup>a</sup>really power down?<sup>o</sup> Type **y**. This is a clean shutdown.

**Note:** Entering **halt -p** will turn off the computer.

## Shutting Down with the NMI Mini-Monitor

If none of the earlier shutdown procedures work, you can shut down by using the NMI mini-monitor.

1. Access the NMI mini-monitor by holding down both the left Alternate key and the Command bar while pressing the ~ key (without pressing Shift). On keyboards with two Command keys, hold down both and press the ~ key.
2. Enter **halt**. The monitor saves all files to disk, and takes you to the ROM monitor.
3. Press the Power key.
4. A prompt appears asking <sup>a</sup>really power down? Type **y**. This is also a clean shutdown.

**Important:** If entering **halt** doesn't work, you can enter **mon** (or **monitor**). This takes you directly to the ROM monitor, where you can use the Power key to power off. This is *not* a clean shutdown.

## Shutting Down with the Restart/Power-Off Panel

Another shutdown method is the Restart/Power-Off panel.

1. Hold down the Command bar (on keyboards with two Command keys, hold down the right one), then press the ~ key (without pressing Shift). The Restart/Power-Off panel appears.
2. Press the Power key.

This is a clean shutdown.

## Emergency Shutdown

If none of these procedures is successful, you can do an emergency shutdown.

The two procedures described here do *not* result in a clean shutdown. They are also potentially damaging to an optical disk if the drive is in the middle of a write operation. Be aware that after such a shutdown, your computer will take a long time to reboot because it must check for file system damage.

In an emergency, you can shut down your computer with either of the following:

- Unplug the computer.
- Hold down both the left Alternate key and the Command bar (the left Command key on keyboards with two Command keys) while typing the asterisk (\*) key on the numeric keypad. This will not power off the computer, but will do a hard reset of the CPU, restarting the computer at the beginning of the power-on system test procedure.

## Troubleshooting

You can receive several types of error messages during system startup. This section covers **fsck** error messages, which are produced if file system corruption is found, and boot error messages, which might be caused by damage to the root file system or by an incorrect network configuration

If problems arise that are unrelated to these types of diagnostic checks, you can look for additional messages by placing the system in verbose test mode, which displays startup messages on the screen (see <sup>a</sup>Inspecting or Modifying Configuration Parameters,<sup>o</sup> earlier in this chapter). You can also use the NMI mini-monitor **msg** command to view the kernel messages (see <sup>a</sup>Displaying the Kernel Message Buffer<sup>o</sup> earlier in this chapter). For more information, see Chapter 15, <sup>a</sup>General Troubleshooting.<sup>o</sup>

**Tip:** If you're experiencing boot problems, be sure to boot from the ROM monitor, or set the <sup>a</sup>verbose mode<sup>o</sup> parameter from the ROM monitor. This will let you see all the system messages as the computer is booting, including any error messages.

## Error Messages from fsck

During the boot process, **init** calls **/etc/rc.boot**, which runs the **fsck** program to check the disks. If file system corruption is found during this process, you'll receive **fsck** error messages.

The **fsck** program checks the file system and repairs any easily fixed inconsistencies it finds. If it finds other types of inconsistencies, it exits with an error condition, and the reboot fails.

## Running fsck

If the boot fails because the root file system is corrupted, or you need to repair some other file system, you can run **fsck** <sup>a</sup>by hand.<sup>o</sup>

**Note:** If an expert UNIX system administrator is available, you may want to ask the expert to assist you with this procedure.

1. Boot the system in single-user mode. (If you're repairing a file system other than the root file system, you needn't be in single-user mode, but you must make sure the file system isn't mounted.) From the ROM monitor, enter something like:

```
bsd -s
```

See <sup>a</sup>Booting from a Device<sup>o</sup> earlier in this chapter for the appropriate boot command.

2. If prompted, enter the hardware password.
3. At the single-user mode prompt (<sup>a</sup>#<sup>o</sup>), enter the following:

```
fsck -y /dev/rsd0a
```

If you're correcting a disk other than the boot disk, replace */dev/rsd0a* with the appropriate device name. The **y** option tells **fsck** to repair any inconsistencies it finds without waiting for a response from the user.

As each inconsistency is found and corrected, one or more lines are printed identifying the file system on which the correction takes place and the nature of the correction. After **fsck** successfully corrects a file system, it prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

4. Repeat the **fsck** command until it finds no inconsistencies. If you're prompted to reboot the system, do so before running **fsck** again.
5. Reboot the system.

For further information, see the UNIX manual page for **fsck**.

## System Boot and Network Startup Errors

A few messages which appear to be errors are really just status messages. Some of these are described in this section, along with messages which, in fact, indicate errors during booting.

## NetInfo Errors

If your network is configured incorrectly, you might see the message:

```
No response from network configuration server.  
Type CTRL-C to start up computer without a network connection.
```

This message occurs when no server responds with configuration information. Possible causes include:

- The network servers are down, or otherwise unreachable.
- The physical network has hardware problems (for example, an incorrectly terminated Ethernet cable).
- A **bootpd** process is not running on the configuration server. A computer obtains its Internet address and host name at boot time from the **bootpd** and **rpc.bootparamd** daemons running on the configuration server. This information is stored in the NetInfo database when the host is initially added to the network.
- The configuration server does not have information about your computer (specifically, the Ethernet address).
- An error occurred during automatic host addition, if it's enabled (for example, another computer was being added at the same time you were adding yours), resulting in incorrect or incomplete information in the NetInfo database.

Before typing Control-c and starting without a network connection, try to determine the cause of the message. For example, the configuration server might be in the process of rebooting. If this is the case, it responds to the request from your computer after it finishes.

The following message might indicate NetInfo problems:

```
Still searching for parent network administration (NetInfo) server.  
Please wait, or press `c' to continue without network user accounts.  
See your system administrator if you need help.
```

This message is displayed when your computer is looking for the parent of its local domain, and the NetInfo server for the parent domain doesn't respond. If you type **c** at this point, your local domain is not bound to its parent domain, and all configuration information residing in that domain (including users, aliases, and machines) is not available. For more information, see Chapter 10, <sup>a</sup>Configuring a Large Network.<sup>o</sup>

This message can occur for various reasons:

- All servers for the parent NetInfo domain are unavailable.
- Your computer's Internet address is not recognized by a server for its parent NetInfo domain.
- The local domain is referencing a computer as a server for its parent domain which is not, in fact, such a server.
- The local domain is referencing a nonexistent computer as the one to serve its parent NetInfo domain.
- Although servers for the parent NetInfo domain are running, no servers responded to the request.

A server for a domain might not respond to a request for one of two reasons:

- The server or the network is overloaded.
- The server is ignoring all NetInfo requests.

When a NetInfo server is started, it checks the consistency of its database; during this operation, it does not respond to NetInfo requests. If the system crashed, and if you have a large NetInfo database or large NetInfo network, it can take a long time (several minutes) to complete these consistency checks. When the checks are completed, the server for the parent domain responds to your computer's request, and your computer will bind correctly to its parent NetInfo server.

## System Boot Errors

If various system components are missing, the system will not be able to boot. Lack of some of these components produces, at best, a very cryptic error message; sometimes, the system will simply freeze during booting. Some of these messages are:

### Message

`unknown binary format`

`filename not found`

`Load of filename failed, errno n`

### Meaning

The kernel (**`mach`**, **`sdmach`**, **`odmach`**, **`fdmach`**) is not a proper executable file.

The kernel file could not be found.

The indicated file could not be loaded. Typically, this file will be **`/etc/mach_init`** or **`/etc/init`**. The error number, *n*, corresponds to the system call error numbers listed in **`/usr/include/bsd/sys/errno.h`**.



`init exited with 1`

**/etc/init** is missing, or is not executable (check the file permissions).

If you unexpectedly find yourself in single-user mode, verify that **/etc/rc.boot** and **/etc/rc** are present.

If the boot process otherwise appears normal, but none of the expected messages from any of the **rc** files are being displayed, **/dev/console** might be missing, or might be an ordinary file rather than a device file. (If it's an ordinary file or is missing, you can use **/usr/etc/MAKEDEV** to create an appropriate device file.)

If your computer is part of an NIS domain, it will be unable to boot if it can't contact an NIS server for the domain. If this happens, messages indicating that **lookupd** is timing out will appear. These errors occur if all NIS servers for the domain are down, if there are network problems, if you've configured your computer as part of a non-existent NIS domain, or if you remove your computer from the network. In the latter case, if you want to run your computer, you'll have to edit **/etc/hostconfig** in single-user mode to set the YPDOMAIN parameter to -NO-.