You have three options: Format, to provide a quick format with no checks for bad blocks; Check, which is slower but checks the hard drive for bad blocks; and No, to bypass formatting the partition. Choose Check and then OK to format this partition.

NOTE Checking for bad spots on your hard drive probably isn't necessary for newer (that is, IDE) drives, because the drives take care of remapping bad spots. ■

Next, the Setup program asks about inode density. Inodes are explained in Chapter 15, "Understanding the File and Directory System," but basically you need one inode for each file on your Linux system. If you're going to have many small files, you need many inodes. The Setup program displays the following screen:

```
SELECT INODE  DENSITY

Ext2fs default to one inode per 4096 bytes of drive space. If you're
going to have many small files on your drive, then you may need more
inodes (one is used for each file entry). You can change the density
to one inode per 4096 bytes, or even per 1024 bytes. Select '2048' or
'1024', or just hit enter to accept the default of 4096 bytes.
NOTE: If you are going to run from CD using a small (<60MB) partition,
use 1024 to be safe. Each link uses an inode and it's easy to run out
of space.
```

▶ **See** "Directories and Physical Disks," **p. 291**

Select the second entry, 4096, and choose OK. The Setup program then displays an information screen indicating the formatting choices you've made.

CAUTION

Watch the drive in-use light. Don't press any keys until the program displays the next screen with an option you can select. If you press any keys during an operation, the Slackware installation program will use these keys as your next selection when the next screen appears. Thus, you may inadvertently make a selection you didn't mean to make.

Next, if Setup detects any DOS or OS/2 High Performance File System (HPFS) partitions, Setup asks whether you want to make these partitions visible to Linux. If you have any such partitions on your drive, answering yes is a good idea because you can access the files on these partitions from Linux. Make your choice (yes or no) to continue.

If you choose yes, the CHOOSE PARTITION screen appears, listing the various partitions you can select. In the entry box, type the name of the partition you want to make visible to Linux and press <Return>. If you have more than one partition available, the Setup program continues to ask for a choice until you enter <q>.

After you enter the name, Setup asks for a place in the directory tree to mount the partition. What this means is that the entire partition is accessed as though it were a subdirectory. You must specify where this subdirectory starts. The following information appears:

```
SELECT MOUNT POINT

Now this new partition must be mounted somewhere in your directory tree.
Please enter the directory under which you would like to put it. For
instance, you might want to reply /dosc, /dosd, or something like that.
NOTE: This partition won't actually be mounted until you reboot.

Where would you like to mount /dev/hda1?
```

In the entry box, enter the directory where you want to mount the partition. If this is a DOS partition, use /dosc for drive C, /dosd for drive D, and so on. If you're mounting an OS/2 partition, you can respond with /os2c, /os2d, and so on. Also, make sure that you specify the correct slash mark in directory names—/ (forward), not the DOS backslash (\).

After entering the mount point, the Setup program displays an information screen called CURRENT DOS/HPFS STATUS to let you know which partitions will be mounted. After dismissing this screen by choosing OK, you're returned to the CHOOSE PARTITION screen.

If you have other partitions to mount, you can repeat the preceding procedure until all desired partitions are mounted. When all are processed, press <q> to move on to the next phase of installation, selecting what media to install from.

Choosing from Where to Install Linux

After formatting and mounting your directories, the Setup program asks you to continue with installation with the SOURCE selection. This selection lets you pick from where you'll install the Slackware distribution. If you're using the accompanying CD-ROM, that choice is CD-ROM. If you have the distribution on floppies, you use the floppy drive to install. Choose yes to display the SOURCE MEDIA SELECTION screen:

```
SOURCE MEDIA SELECTION
Where do you plan to install Slackware Linux from?
1      Install from a hard drive partition
2      Install from floppy disks
3      Install via NFS
4      Install from a pre-mounted directory
5      Install from CD-ROM
```

If you're using the CD-ROM, select choice number 5 and choose OK, which displays the Installing from CD-ROM screen:

```
INSTALLING FROM CD-ROM

In order to install Slackware from a CD-ROM, it must contain the
distribution arranged beneath a source directory in the same way
as if you were to install it from a hard drive or NFS. The source
directory must contain subdirectories for each floppy disk. Your
CD-ROM should be compatible with this format if it contains a mirror
of the Slackware FTP site.

What type of CD-ROM drive do you have?
```

```
1      SCSI [/dev/scd0 or /dev/scd1]
2      Sony CDU31A [/dev/sonycd]
3      Sony 535 [/dev/cdu535]
4      Mitsumi [/dev/mcd]
5      Sound Blaster Pro (Panasonic) [dev/sbpcd]
```

Select your CD-ROM drive from those provided and choose OK.

NOTE If you have problems getting the installation program to detect your CD-ROM, you may have to abort the installation and create a new root disk with another kernel. See The CDROM-HOWTO in the /doc/HOWTO directory for more information on using CD-ROMs.

You now need to look up the source directory you saved in the section "Repartitioning the Drive" earlier in this chapter. The Setup program displays the SELECT SOURCE DIRECTORY screen.

NOTE The following screens may appear differently because Linux changes so rapidly. Every effort has been made to assure that this book is in step with the accompanying CD-ROMs, but this isn't always possible.

```
SELECT SOURCE DIRECTORY

Now we need to know which directory on the CD contains the Slackware
sources. This location may vary depending on the cd you have. There
are default selections for the Slackware Professional CD (including
an option to run mostly from the CD), InfoMagic CD, TransAmeritech CD,
and the Linux Quarterly CD-ROM. There may be other directories
containing other versions - enter a custom directory name if you like.
Which option would you like?
```

If you're using the accompanying Slackware CD-ROM, immediately select the type of install—slackware, slaktest, and so on. Make your selection and continue. The next screen asks you to pick the packages to install on your system.

NOTE If you enter the wrong directory or don't remember the directory name, the Setup program alerts you and prompts for the correct directory name. If you're using the accompanying CD-ROM, the /slackware directory is the correct directory on the CD-ROM. If you're using another installation media or CD-ROM and forgot the directory name, you need to exit Setup and look on the media to find the directory. You can usually recognize the directory structure because the subdirectories have names similar to the package names–for example, /a, /ap, /oop, and so on.

Selecting the Series to Install

After you specify the installation media and source directory, the Setup program displays the SERIES SELECTION screen:

```
SERIES SELECTION
```

```
Use the spacebar to select the disk sets you wish to install.
You can use the UP/DOWN arrows to see all the possible choices.
Press the ENTER key when you are finished. If you need to install
a disk set that is not listed here, check the box for custom
additional disk sets.

CUS   Also prompt for CUSTOM disk sets
A     Base Linux system
AP    Various Applications that do not need X
D     Program development (C, C++, Lisp, Perl, etc)
E     GNU Emacs
F     FAQ lists, HOWTO documentation
K     Kernel sources
N     Networking (TCP/IP, UUCP, Mail, News)
Q     Extra Linux kernels with UMSDOS/non-SCSI CD drivers
T     TeX
TCL   Tcl/Tk/TclX, Tcl language, and Tk toolkit for X
X     XFree-86 2.1.1 X Window System
XAP   X Applications
XD    XFree-86 2.1.1 X11 Server Development System
XV    XView 3.2 release 4. (OpenLook window Manager, apps)
Y     Games (that do not require X)
```

Simply move through the list with the arrow keys and mark the desired packages with the space bar. When you make a selection with the space bar, an `X` appears next to the selection. When you've made all your selections, press <Return> to continue.

Your selections are based on personal preference and the type of hardware you have—that is, if you don't have the hardware to run XFree86, you shouldn't install the packages that require X (`iv`, `x`, `xap`, `xd`, and `xv`). You must install package A, the base Linux system. Also, if you're interested in programming, you should install the various programming packages, such as `d`, `oop`, `tcl`, and `xd`. If you aren't interested in programming, you really don't need these packages. If you want to access the Internet, you want to install package `n`. Installing the `f` package, which contains FAQ lists and HOWTO documentation, is strongly recommended, because the package contains a wealth of information you'll need about Linux.

For each package you select, you're led through a series of screens for each package, and each package has programs that it must add, programs that it highly recommends you add, and programs you can skip. Simply follow the instructions on each screen to install the system. For this example, you install the following packages: `a`, `ap`, `d`, `e`, `f`, `n`, `t`, `tcl`, `x`, `xap`, and `y`.

After you select the desired packages, choose OK to continue. A simple screen appears telling you that you're about to enter the INSTALL section and that if you haven't made all the appropriate selections, you're returned to the main selection menu. Choose yes to continue with the Setup process.

Installing Your Selections

The Setup program next prompts you for the type of prompting you want to have while Setup goes through each package, installing programs. Each package contains a set of tagfiles that indicate how the file should be treated. For a first-time install, select the HELP mode indicated in the SELECT PROMPTING MODE screen, to help you decide which mode to use.

```
SELECT PROMPTING MODE
Now you must select which type of prompting you would like to use while
installing your software packages. If you're not sure which to use,
read the help file.

Which type of prompting would you like to use?

Normal   Use the default tagfiles
Custom   Use custom tagfiles in the package directories
Path     Use tagfiles in the subdirectories of a custom path
None     Use no tagfiles - install everything
HELP     Read the prompt mode help file
```

After reading the HELP file, select the Normal prompting mode and choose OK. Then choose INSTALL.

NOTE The HELP file indicates a Prompting mode, but as you can see from the menu selections, there's no such item. Normal is the closest matching item, which is why you should use that selection for the installation procedure.

At this point, the Setup program is on autopilot and goes through each package you selected earlier to install the various programs. Programs marked as `ADD` in the tagfiles are automatically added. Setup displays a screen indicating what package it's installing and a brief message about what the package is. You can't stop the Setup program from installing a program marked as `ADD` in the tagfile.

When the Setup program comes across a program marked as `OPT`, `REC`, or `SKIP`, it displays a screen telling you what it's about to install, whether it's recommended that you install the program, how much space is required to install the program, and then a list of choices. The choices—accessible with the arrow keys—are usually yes, no, or an option to abort the entire program. Usually, the default selection is `yes, install package` *`xxx`*, where *`xxx`* is the name of the package being installed.

NOTE Be careful with your answers. If you make an incorrect choice, you can't go back and change that choice. If you accidentally install a package you didn't want, things aren't so bad; you may lose some disk space and gain another program to experiment with. However, if you don't install a package you need, the best you can do is write down the missed package and then later run `pkgtool`, explained in Chapter 13, "Upgrading and Installing Software," to install the desired package. You can also abort the current installation and start over, but that's a rather harsh and time-consuming option.

▶ **See** "Using *pkgtool*," **p. 253**

Configuring Your System

The Setup program has finished loading all the software components you specified. Now it must configure your system. The Setup program displays

```
CONFIGURE YOUR SYSTEM

Now it's time to configure your Linux system. If this is a new system,
you must configure it now or it will not boot correctly. Otherwise,
you can back out to the main menu if you're sure you want to skip this
step. If you've installed a new kernel image, it's important to
reconfigure you system so that you can install LILO (the Linux Loader)
or create a bootdisk using the new kernel. Do you want to move on to
the CONFIGURE option?
```

Because this is your first time through, you need to configure your system. Choose yes to continue. The next screen asks you to make a boot disk. You should create a boot disk, even if you use LILO. The screen displayed is as follows:

```
MAKE BOOT DISK

It is HIGHLY recommended that you make a standard boot disk for your
Linux system at this time. Such a disk can be very handy if LILO is
ever improperly installed. Since the boot disk will contain a kernel
that is independent of LILO and the kernel on your hard drive, you'll
still be able to use it to boot your system no matter what you do to
LILO or your hard drive kernel. Would you like to make a standard
boot disk?
```

Be sure to have a formatted floppy ready, choose yes, and press <Return>. Setup displays the BOOT DISK CREATION screen. Simply put the formatted floppy into the drive and choose yes to create the boot disk. If you skip boot disk creation, Setup displays this warning message:

```
SKIPPED BOOT DISK CREATION

Boot disk creation skipped. I hope you already have a boot disk.
If you don't, you have to install LILO if you haven't already,
or you'll have a hard time booting your machine. :^)
```

It's highly recommended that you create a boot disk. If anything goes wrong with the preceding installation or the following configuration—especially when installing the Linux Loader, LILO—you'll have an extremely hard time booting your system. In fact, you should also make a boot disk for any other operating systems you may have resident on your system. Typically, with a boot disk you can boot from a floppy if things go bad and correct the problem.

Configuring Your Modem

Next, Setup asks you to configure a modem. You should do this now, even if you don't plan to use a modem at this time. Setup displays the following screen if you want to configure your modem:

```
MODEM CONFIGURATION

This part of the configuration process will create a link in /dev from
your callout device (cua0, cua1, cua2, cua3) to /dev/modem. You can
change this link later if you put you modem on a different port.
```

Choose yes to continue. Next, you need to specify the serial port that your modem is hooked to. You do this via the SELECT CALLOUT DEVICE screen. The items cua0-3 represent your serial ports, with cua0 representing COM1, cua1 being COM2, and so on. Select the appropriate COM port and the choose OK.

Configuring the Mouse

You next go through a similar process to configure your mouse. If you have a mouse with your system, you should go ahead and configure it at this time. Choose yes from the MOUSE CONFIGURATION screen to continue. You're then presented with a screen containing six selections. If you have a mouse that's Microsoft-compatible and not listed in the choices in Table 4.10, you can select option 1 and have a good chance of your mouse working.

Table 4.10 Mouse Types Supported by Linux

Option	Description
1	Microsoft-compatible serial mouse
2	C&T 82C710 or PS/2 style mouse (auxiliary port)
3	Logitech bus mouse
4	ATI XL bus mouse
5	Microsoft bus mouse
6	Mouse Systems serial mouse

If you select a mouse that requires a serial port, Setup requires you to specify the port. Like modems, Linux refers to the serial ports with a different name than COM1, COM2, and so on. Linux refers to the ports as ttyS0 through ttyS3. Select the appropriate serial port from the SELECT SERIAL PORT screen and then choose OK to continue with the installation.

Configuring Linux to Use *ftape*

Setup detects whether you loaded the `ftape` package and asks whether you want to start the program as soon as Linux boots. It doesn't hurt to start the program at boot time, so if you loaded the `ftape` package, you should have Setup start the program at boot time. Choose yes from the FTAPE CONFIGURATION screen to continue.

Configuring Your Modem's Baud Rate

Next, if you installed the `gp9600` package, the Setup program displays the SET YOUR MODEM SPEED screen. Simply select the appropriate baud rate and choose OK. If you don't see a rate high enough for your modem needs, you can later use the `setserial` program to set the baud rate to whatever value you require.

Installing LILO

LILO stands for the LInux LOader. LILO is a program executed at system startup that lets you choose which operating system is used to boot the computer. You can use LILO to boot several different operating systems, such as Linux and MS-DOS. With LILO, you also can specify a default operating system to boot and a default time limit before it boots that system. For example, if you had MS-DOS and Linux on your computer, you can configure LILO to boot either one. You could then tell LILO to boot MS-DOS if no one intervenes after 30 seconds. Before that 30 seconds is up, however, a user can specify another operating system to boot instead of the default. You can press the <Ctrl>, <Alt>, or <Shift> keys to stop the timed process. Press <Tab> to get a list of operating systems LILO can boot.

You specify all this information while configuring LILO. Although you can directly edit the lilo.conf file located in the /etc directory, the LILO INSTALLATION screen provides a better interface for editing the file.

After you configure your system, Setup lets you install LILO. Setup displays

```
LILO INSTALLATION

LILO (the Linux Loader) allows you to boot Linux from your hard drive.
To install, you make a new LILO configuration file by creating a new
header and then adding at least one bootable partition to the file.
Once you've done this, you can select the install option, Alternately,
if you already have an /etc/lilo.cfg, you may reinstall using that.
If you make a mistake, you can always start over by choosing 'Begin'.
Which Option would you like?

Begin          Start LILO configuration with a new LILO header
Linux          Add a Linux partition to the LILO config file
OS/2           Add an OS/2 partition to the LILO config file
DOS            Add a DOS partition to the LILO config file
Install        Install LILO
Recycle        Reinstall LILO using the existing lilo.conf
Skip           Skip LILO installation and exit this menu
View           View your current /etc/lilo.cfg
Help           Read the Linux Loader Help file
```

To begin, you should select the Help option to read the help file. After reading the help file, you should start at the beginning—the Begin option. If you don't want to install LILO at this time but instead want to use a boot floppy, you can select the Skip menu option.

CAUTION

If you skip installing LILO, you must have a bootable floppy. If you skipped creating a boot floppy earlier, you should install LILO or, when back at the main menu, re-enter the configuration selection and create a boot floppy. If you do leave Setup without a way to boot your system, you're forced to configure your system at a later time with the boot and root disks you originally created.

Configuring the Kernel with the *append*= Parameter

You have the option to pass the Linux kernel command-line flags to configure the kernel before execution begins. The extra flags might be needed with certain SCSI hard drives and IBM motherboards. LILO allows you to specify these command-line flags via the append= parameter. If you need to pass any command-line parameters, enter them in the edit box of the displayed screen. If you have no parameters to pass, press <Return> to continue.

Setting a Target Location

Next, you must choose where to place the LILO program. You can place the program in the master boot record of your first hard drive, in what's called the *superblock* of your root Linux partition, or on a floppy disk. If you choose the Floppy Disk option, you need to place a formatted floppy in the drive. You should probably use the master boot record for LILO.

Setting the Delay Option

The next screen lets you set the amount of time LILO waits before booting the default operating system. Select one of the following options and then choose OK:

Option	Description
None	Don't wait at all—boot straight into the first operating system
5	5 seconds
30	30 seconds
Forever	Present a prompt and wait until a choice is made

Choosing Your Default Operating System and Adding All Partitions

When you return to the main LILO INSTALLATION screen, you must choose your default operating system. This is the first operating system located in the lilo.conf file. If you want Linux to be the default operating system, for example, you should select the Linux menu option; if you want MS-DOS to be the default, select the DOS menu option.

After you select the default OS, the installation program presents a screen consisting of all the possible partitions that you can boot from. Enter the name of the partition just as it appears under the Device column heading on the SELECT *XXX* PARTITION, where *XXX* indicates the type of partition you're using. For example, if you had selected DOS as your default operating system, the screen reads SELECT DOS PARTITION and displays all available bootable DOS partitions. After you enter the correct device name, choose OK to continue.

Next, you must choose a short name to help identify the operating system when someone presses the <Tab> key at the LILO prompt. This is the name a user must enter to select that operating system as the boot operating system from LILO. Examples of names are DOS, LINUX, OS/2, and so on. The name must be a single word.

After selecting your default operating system, you can continue to add various operating systems to LILO by using the Linux, OS/2, and DOS menu options. Just remember to add Linux.

After you add all the appropriate operating system partitions, you should use the View option to examine your current /etc/lilo.conf file. For this example, assume that you've specified DOS as the default operating system and have added an entry for Linux. You also specified a 30-second delay before booting into DOS. In that case, your lilo.conf file looks like

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
#compact        # faster, but won't work on all systems.
delay = 300
vga = normal    # force sane state
ramdisk = 0     # paranoia setting
# End LILO global section
# Linux bootable partition config begins
image = /vmlinuz
root = /dev/hda4
label = linux
# Linux bootable partition config ends
# DOS bootable partition config begins
other = /dev/hda1
label = dos
table = /dev/hda
# DOS bootable partition config ends
```

After you add all the needed partitions, select the Install option to configure LILO.

Uninstalling LILO

If you're running LILO version 0.14 or newer, you can uninstall LILO with the following command:

```
opus:~# lilo -u
```

If you have a previous version, you must remove or disable LILO from its primary partition. You can use Linux's or MS-DOS's FDISK program to make another partition active.

If you placed LILO within the MBR (master boot record), you must replace it with another MBR from another operating system. With MS-DOS 5.0 or above, the command

```
c:\>fdisk /mbr
```

restores the MS-DOS MBR.

When LILO is removed from the active partition or the MBR, you're free to remove the files from /etc/lilo.

▷ **See** "Removing Files or Directories," **p. 307**

Configuring Your Network Next, Setup allows you to configure your network. You may not have all the information available yet, but go ahead and configure as much as possible. The Setup program tries to configure your system, but if it can't (which is very likely the first time), you can configure the network later with the `netconfig` command. Choose yes to start the configuration, and choose OK on the NETWORK CONFIGURATION screen to begin.

The first item you need is a name for your machine. This is a personal name, and you can be as creative as you want. At the `ENTER HOSTNAME` prompt, enter your selected name and press <Return>.

NOTE It's a good idea to stick with lowercase letters, because UNIX and Linux are case-sensitive and most commands and interactions are done with lowercase letters. ■

The next prompt asks for something called a *domain name*. If you understand this term and have one available, enter the domain name for your system. If you don't have a domain name yet or don't understand what one is, don't worry. (Later chapters of this book, such as Chapter 26, "Understanding the Internet," explain what an Internet domain name is and how to configure your network with one.) If you don't have a domain name, enter the following as a placeholder:

```
tristar.com
```

▶ **See** "Internet Names," **p. 525**

NOTE You must enter a domain name to continue the installation process. ■

The next question deals with using *TCP/IP* through *loopback*. (If you don't understand these terms, check out Chapter 23, "Configuring a TCP/IP Network," for this information before returning to this section.) By answering yes to the `Only use loopback?` prompt, you can skip several layers of configuration. Answering yes to this question and configuring your network later is a good idea. If you choose yes, the network configuration is complete and you can continue with the rest of the configuration.

▶ **See** "The/etc/hosts File," **p. 474**

▶ **See** "Configuring the Software Loopback Interface," **p. 478**

Using the *selection* Program If you have a mouse, you can use the `selection` program to cut and paste commands from your terminals. This screen asks whether you want to start the selection program automatically at boot time. If you have a mouse, you should start `selection` at boot time by answering yes at this screen. However, if you have a bus mouse, there are potential problems using `selection` and XFree86. If you have a bus mouse, you shouldn't start `selection` automatically.

If you don't choose to start `selection` at boot time, you can execute the program at any time with the following command:

```
selection -f &
```

Configuring *sendmail* Next, Setup asks you to choose a configuration file for `sendmail`. Chapter 29, "Using Electronic Mail," provides more information on using e-mail, but for starters you might want to preload one of the supplied sendmail configuration files. If you know how you're connecting to the Internet with PPP, via Ethernet and so on, you can choose the SMTP-BIND or SMTP menu option. If you plan to use a modem and UUCP, select the UUCP menu option. If you don't plan on using a network, select the SKIP menu option. If you're unsure, the SKIP menu option is a good choice because you can reconfigure the system at a later time.

▶ **See** "Understanding E-Mail," **p. 570**

Selecting a Time Zone Next, Setup requires you to select a time zone that Linux uses to keep track of the date and time. Look through the list of available time zones and choose the most appropriate one. If you're in the United States, several selections begin with US. The same applies for Australia, Canada, and those systems that use Greenwich Mean Time or Universal Time. Select your time zone and press <Return>.

Replacing/etc/fstab /etc/fstab is a file-system table that represents each of your partitions. If you modify the table—with LILO, for example—while using the Setup program, it may ask you to replace the old one with the new one. This screen appears only if, for some reason, you stop and restart the Setup program or the configuration process. If you've made no changes to the partition table, answer no to this question; otherwise, answer yes.

Modifying the Kernel with *rdev*

When configuration is complete, you can rerun Setup to add new items, or you can use the `pkgtool` program to view, add, or delete packages.

▶ **See** "Using *pkgtool*," **p. 253**

You might have to change items germane to your kernel, such as the video mode used. One solution is to reinstall Linux—a major undertaking, as you've just seen. Or you can recompile and build a new kernel from scratch, but that's not for the novice. Another option is to modify the current kernel, which, fortunately, Linux allows you to do with the `rdev` program. `rdev` is found in /sbin and should be used only when you're logged in as root.

For a complete listing of options, invoke `rdev` with the `\h` or `\?` parameter.

Your kernel file—that is, the actual software—is found in a file named vmlinuz. The reason for this name is historical; most UNIX systems store the kernel in a file named vmunix, and Linux is based on UNIX. Note the installed kernel is vmlinuz, where the z signifies a compressed kernel. When a kernel is built, the decompressed version is made and called vmlinux; however, the decompressed version isn't bootable. A parameter to `rdev` is usually this file name. You can use `rdev` to fix problems such as root and swap partitions and video modes.

Building a New Kernel

Sometimes a problem has only one solution—a new kernel. The Linux kernel is the core operating system for Linux. Although not for the faint of heart, sometimes downloading a new kernel from the Net and building it is necessary. If you have some programming experience and know your way around the C programming language, you should be able to build and install a new kernel; if not, you can skip this section.

You may have to install a new kernel for various reasons:

- A new patch is provided to run new hardware.
- You want to remove features from the kernel you don't use, thus lowering the memory requirements for your system.

The starting point is to determine what kernel version you're now running. You can find out the kernel version with the following command:

```
uname -a
```

The response indicates which version of the kernel is now running and when it was created. The version numbers are in the form of

MajorVersionNumber.MinorVersionNumber.PatchLevel

Linus Torvalds is the official release point for new kernels, although anyone can modify Linux (due to the GPL). The fact that Linus is the official release point gives the Linux development and user community a common baseline from which to work and communicate.

NOTE Be sure to read the Kernel HOWTO before actually trying to build and configure a new kernel! The Kernel HOWTO is in /doc/HOWTO.

To build a new kernel, you need to have the source files in the /urs/src/linux directory. You also need to have the C compiler package loaded, which is disk set `d`. If you didn't install that package during installation, use `pkgtool` to do so now.

First, you must get the new kernel sources or patches. The new sources are usually found on the Internet (check out **sunsite.unc.edu** for the latest and greatest kernels). The source files are usually in a `tar` format and will need to be unarchived. If you're modifying your current kernel, obtaining the new sources is, of course, unnecessary.

NOTE It's a *very* good idea to make a backup copy of your current kernel with the following commands:

```
cd /usr/src
cp linux linux.sav
```

These commands copy the entire linux source directory to another directory called linux.sav.

Next, you should use the `patch` command to apply any patch files. After preparing the source files, you can configure and build your new system. Start by entering the following command from the /usr/src directory:

```
make config
```

This command asks you various questions about the drivers you want to install or configure. Pressing <Return> accepts the default value for each question; otherwise, you must supply the answer. Table 4.11 shows some of the questions; you may have to answer other questions depending on the version of the kernel you're installing or the patches you've applied.

Table 4.11 Some Config Questions

Configuration Option	Description
Kernel Math Emulation	Asks if the kernel should emulate a math co-processor.
Normal Hard disk Support	Enables drivers for all standard hard drives.
XT Hard disk Support	Applies only if your machine uses an XT class controller class instead of an AT controller.
Networking Support	Answering yes enables networking support within the kernel.
SCSI Support	Enables support for SCSI controllers.
CD_ROM Drivers	A series of questions dealing with CD-ROMs, especially those not supported within the standard SCSI Support package.
Filesystems	A series of questions dealing with file systems the kernel should support. If your kernel doesn't have support for the ISO9660 file system, you won't be able to use a CD-ROM.
Parallel Printer Support	Enables printer support via the parallel port.
Mouse Support	Bus mice need their own support from within the kernel.
Sound Card Support	A series of questions about the hardware and software configuration of a sound card.

After you answer the various questions to configure your new kernel, you must compile it.

NOTE The build process can take anywhere from 15 minutes to many hours. So relax and order a pizza! □

The following commands will build the new kernel:

```
make dep
make clean
make
```

When compilation completes, you can create a new boot disk as discussed earlier in the section "Creating the Boot and Root Disks." You can copy this kernel to a new diskette, or use LILO to boot the new kernel.

Upgrading from a Previous Version of Slackware

The current version of Slackware (version 3.0) contains the Linux files in a special format, called Executable and Linking Format (ELF). Earlier versions of Slackware were in a format called a.out. You can't mix these types of executables on the same system, so if you plan to upgrade from an earlier version, you must reinstall from scratch to be on the safe side. This is especially important when dealing with the A and N packages. The `pkgtool` program can uninstall programs as well as install them, but the safest route is to back up your important configuration files and reinstall Linux from scratch.

If you're upgrading from similar formats—that is, from an ELF-based distribution to an ELF-based distribution—you can use `pkgtool` to remove those packages you want to update, and then use the tool to install the newer versions.

▶ **See** "Using *pkgtool*," **p. 253**

Going Back to the Beginning

After you complete the setup and configuration of your system, the Setup program returns you to the main menu. From there, you can choose the `EXIT` option to leave Setup. If you want to change options, you can do so here. But Chapter 13, "Upgrading and Installing Software," provides information on updating and installing software after your initial installation. Choose `EXIT` to leave the Setup program.

Choosing `EXIT` returns you to the system prompt, indicated by the `#` sign. You're now in Linux and can issue simple commands, such as `ls` for a directory listing of files. At this time, though, you should reboot the system so that all your setup and configurations can take effect.

Rebooting Linux is more involved than rebooting DOS. You can't turn off the power and turn the system back on. If you do so in Linux, you can damage the file structures and systems. Linux tries to repair itself on bootup. Don't turn off the power while running Linux. To exit Linux, use the following command:

```
shutdown [-r] time
```

The optional `-r` flag indicates that the system should reboot after shutting down. *`time`* indicates the time that the system should shut down; you can use `now` in place of *`time`* to indicate immediate shutdown. Linux also recognizes the warm-boot keys used by DOS to reboot the machine, <Ctrl-Alt-Delete>, which Linux interprets as the command

```
shutdown -r now
```

▶ **See** "Shutting Down Linux" **p. 218**

Make sure that you've removed the all floppy disks from the drive and reboot your new Linux machine.

Resolving Problems

After rebooting your machine, the LILO prompt should appear. Make sure that you can boot to your old operating system if you left it on the hard drive. If that system was DOS, press the <Shift> key and then type the short word you used to identify the DOS partition when you installed LILO. If you enter an invalid word, press <Tab> to get a list of valid operating system types. If you're having problems at this point, place your DOS boot disk in the boot drive and reboot.

You should be able to boot from your boot disk. When your system is up and running under DOS, try the Linux boot disk you created during installation—not the ones you created to originally install the entire system. If that boot disk doesn't work, you may have to reinstall Linux. Potential problems to initially check are the kernels and your hardware. Before starting over, make sure that you have the appropriate hardware. If you made notes during the installation process, check which kernel you installed against what hardware you have. If you have a SCSI CD-ROM, did you install the `idekern` instead of the `scsikern`? But do make sure that you have hardware supported by Linux.

From Here...

After you have your system up and running, you can read the following chapters for further information about Linux:

- Chapter 5, "Running Linux Applications," gets you up to speed on the various programs you just installed.
- Chapter 7, "Using X Windows," is interesting if you've installed the X system.
- Chapter 13, "Upgrading and Installing Software," provides instructions of how to reinstall packages you may have left out during the initial setup of your Linux system.
- Chapter 26, "Understanding the Internet," briefs you on the basics about how to get to the Internet.

CHAPTER 5

Running Linux Applications

Now that you've installed your Linux system, this chapter presents a brief introduction to setting up a user account for you to use and some basic commands to get you moving around your new system. This is your very own multitasking, multiuser system; experimenting is encouraged, so go ahead and play with your system. You may never get this type of opportunity on a typical UNIX system.

However, just playing with an operating system is no fun; it doesn't get your daily job done. After all, you don't use DOS all day, right? You use applications. Linux provides access to literally thousands of applications from around the world. You've installed several from the Slackware or Red Hat distribution from the accompanying CD-ROMs. There are plenty more where they came from, too. Programs that rival those costing hundreds of dollars for the PC platform are readily available for Linux. ▨

- **Using basic file and directory commands**

  Like UNIX, Linux makes extensive use of files, so you need to know how to work with files.

- **Adding new users and executing programs**

  Like Windows NT, Linux allows many different users to access the system at the same time. To gain access, new users must be added to the system.

- **Using the *minicom* modem communications program**

  `minicom`, a communications program available with Red Hat and Slackware, is similar to the DOS-based PROCOMM program.

- **Playing the various games available**

  Linux provides various text-based and graphical games for your enjoyment.

- **Using emulators that let you run DOS and Windows programs under Linux**

  Linux doesn't exist in a vacuum, and the myriad developers realize plenty of DOS and Windows programs are out there that users simply don't want to abandon. So several groups are working on emulators with which you can use your DOS and Windows programs under Linux.

Maneuvering Through Linux

After installing Linux and rebooting, you're faced with a system prompt based on the name you gave your system during installation. The prompt looks similar to

```
Red Hat Linux release 4.0 (Colgate)
Kernel 2.0.18 on an I486
web login:
```

The prompt may indicate a different version of Linux, however, because Linux is an evolving system.

You must now supply a user name and a password. A user name identifies you to the operating system because Linux can support many different users, both at different times and concurrently. An account also provides each user with a default directory, called the *home directory.* Many accounts are also set up to restrict users to certain directories on the system and to prevent them from using certain commands, primarily to protect the files of one user from the prying eyes of another.

Entering Commands

You enter commands in Linux much as you do in DOS and other command-line-oriented operating systems. Linux, like UNIX, is case-sensitive; if Linux doesn't know a command, make sure that you've spelled it correctly and that you've entered it in the proper case. Most commands are executed after you press <Return>.

Recalling Command History

Linux also provides a history function to recall previous commands. This history is kept across sessions, too. You can press the <↑> key to recall previous commands, and then press <Return> to activate that command. To get a complete listing of all the prior commands you've entered, you use the `history` command:

```
[tackett@web~]$ history
1 clear
2 adduser
3 history
```

When you have the preceding history list, you can repeat the command by using the <↑> key and cycling through the commands until the proper one appears on the command line, or you can press <!> (the *bang* character) and enter the number of the command you want to re-execute. For example, if you wanted to repeat the `adduser` command in the previous list, enter

```
[tackett@web~]$ !2
```

The number of entries in the history list is user-defined in the user account's .profile configuration file. See Chapter 17, "Understanding Linux Shells," for more information on the .profile configuration file.

NOTE Linux provides many different command shells, some of which don't provide the history functions.

Making Selections

If you have a mouse with your system and installed the `selection` program, you can also use your mouse to copy text from other areas of your screen to the command line. To select the text, simply move the mouse cursor (which appears as soon as you click the left mouse button) by holding down the left mouse button as you drag the cursor across the desired text, and then press the right mouse button to copy the text to the command line. This is useful if you need to enter a long file name on the command line.

Completing Commands

Linux also offers another nice feature when entering commands. You can start to type a file name and then press <Tab>. Linux searches the directory for a file beginning with the same letters you've typed and completes the file name it finds. If Linux can't find a unique file name, it beeps and completes the file name to the last common character. For example, if you wanted to copy a file called todo_monday to todo_today, you type **`cp to`** at the prompt and then press <Tab>, Linux beeps and fills out the command line like so:

```
[tackett@web~]$ cp todo_
```

If you now typed an `m` and pressed <Tab>, Linux would place the entire todo_monday file name on the command line.

Managing Users

On many systems, the person responsible for maintaining the user accounts is referred to as the *systems administrator*. The systems administrator sets up user accounts and performs other duties. For more information on the various aspects of systems administration, check out the chapters in Part II, "Systems Administration." On your Linux system, you're the systems administrator, so it's your responsibility to set up accounts for yourself, family, and friends.

To add an account for yourself, you must create that account as the systems administrator. Systems administrators are also sometimes referred to as *superusers* because they have so much control over the system. To begin your trek through Linux, you must first log in as the superuser via the root account.

Logging In and Out

To log in as root, enter **`root`** at the login prompt. Linux asks for a password.

By using a password, you prevent unauthorized users from logging in to any account. Linux wants to make sure that the user name is in fact the correct user. You shouldn't share your passwords with just anyone. Linux protects the password you type by not *echoing*—that is, not displaying—the letters on screen, so make sure that you enter the correct password.

If you enter an invalid user name or password, Linux gives the following error message and starts the process over:

```
web login: jack
Password: password
Login incorrect

web login:
```

Because this is your first time logging in to the system since installation, the root account has no password set, so after entering **`root`**, you're presented with a command prompt. You now can enter Linux commands. Most commands are entered in the same way as in DOS: Type the command with any needed parameters and press <Return>.

NOTE The default installation of the Slackware distribution provides a small "fortune cookie" message every time someone logs in to the system. This message displays a short–sometimes amusing, sometimes not–adage about life. You can also get a fortune at any time by using the `fortune` command, provided that you've installed the text-based games.

The Red Hat distribution doesn't provide this functionality by default. To get a fortune cookie under Red Hat, type **`/usr/games/fortune`**.

To log out, enter **`logout`**. This command returns you to the login prompt. If this command doesn't work, try the `exit` command.

Adding Users Under Slackware

After you log in as root, you should add an account for yourself. To add an account, enter the following command and follow the prompts:

```
[root@web~]# adduser

Adding a new user. The user name should be not exceed 8 characters
in length, or you many run into problems later.

Enter login name for new account (^C to quit):
```

Look at this screen for a second. Notice the command prompt after which you entered the command. The prompt begins with the host name of the computer. This is the name you entered while installing the `n` package of disks. The next item is the ~ (tilde) character. Linux uses this character to refer to the account's home directory (described later). Here, it represents the directory the user is now located in. If you issued the `adduser` command from the /usr/bin directory, the prompt reads

```
[root@web~]#/usr/bin#
```

The next character is the pound sign (#). This prompt, by convention, belongs to any superuser account. A normal user account usually has `$` (dollar sign) as a prompt.

Next, you may have noticed the misspellings and improper grammar in the prompts—that is, `should be not` and `you many run`. These errors don't affect the performance of the system, but they help highlight the fact that Linux, while fully functional and a great system, isn't a commercial venture.

Now enter a user name of up to eight characters and press <Return>. An example session to create an account for Jack Tackett follows:

```
Enter login name for new account (^C to quit): jack

Editing information for new user [jack]

Full Name: Jack Tackett, Jr.
GID[100]:<Return>

Checking for an available UID after 500
501...
First unused uid is 502

UID [502]:<Return>

Home Directory [/home/jack]:<Return>

Shell [/bin/bash]:<Return>

Password: opus

Information for new user [jack]:
Home directory: [/home/jack] Shell: [/bin/bash]
Password: [opus] uid: [502] gid: [100]

Is this correct? [y/N] :y

Adding login [jack] and making directory [/home/jack]

Adding the files from the /etc/skel directory:
./ .kermc -> /home/jack/ ./ .kermc
./ .less -> /home/jack/ ./ .less
./ .lessrc -> /home/jack/ ./ .lessrc
./ .term -> /home/jack/ ./ .term
./ .term/termrc -> /home/jack/ ./ .termrc
./ .emacs -> /home/jack/ ./ .emacs

[root@web ~]#
```

As you move through the process, you must enter a full name for the user to help identify the user account further. Next, you're asked to enter a group ID and a user ID. Don't worry about these items at this time. Linux uses them to determine the directories and files that you have access to by default. You can safely accept the default values (within the brackets) by simply pressing <Return> after each request.

Next, you're asked to enter a home directory for the user. This is where the user is automatically placed when he or she first logs in. This is the user's account area for storing files and for working storage. Linux provides a default directory based on the user's name. If this default directory is acceptable, press <Return>; otherwise, enter a directory and press <Return>. Accept, for now, the defaults offered by the `adduser` command.

You're now asked to specify a shell for the user. The shell is a command interpreter much like COMMAND.COM is for DOS. The shell accepts the input and runs specified commands. You've been using a shell called `bash` since installing Linux. For the time being, simply accept the default `bash` option.

▶ **See** "Understanding Shells," **p. 325**

The final parameter is the password for the account. It's highly recommended that you provide every account with a password. Linux then displays all the information entered and asks whether it's correct. If the information isn't correct, enter **`n`** (or simply press <Return>, because No is the default choice); you must go back and correct the errors. If everything is correct, enter **`y`**.

Linux displays a series of files it copies from a skeletal user account located in the ./etc/skel directory to the new user's home directory. These files are configuration files for such items as the user's terminal and how such programs as `emacs` and `less` run from their accounts. The users can modify these files at any time to change the default behavior of the programs.

After adding the account, you can verify its existence in one of two ways; the quickest is to use a utility called `finger` to see whether the user has an account. The general form of the command is `finger` *`name`*. For example, you can test for the account you just created by entering

```
[root@web ~]#finger jack

Login: jack   Name Jack Tackett, Jr.
Directory: /home/jack   Shell: /bin/bash
Never logged in.
No Mail.
No Plan.
[root@web ~]#
```

If the user has an account, the appropriate information is displayed; otherwise, a message indicating no such user has an account is displayed.

The next way to verify the account is to actually log in to the account to see whether Linux will let you. You can do this in one of several ways:

- You can log out and then log in as the new user.
- You can use the `su` command, which stands for *switch user*.
- You can use the `login` command.
- You can use one of the six virtual terminals provided by Linux to log in to a new account. Remember, Linux is multiuser.

Table 5.1 presents an overview of each method.

Table 5.1 Logging In to a Newly Created User Account

Command	Description
`logout`	Logs you out of the root account and brings you back to the login prompt. You no longer have access to the root account until you log in as root.
`su` *`username`*	Logs you out of the account, doesn't ask for the user name to log in as, and then prompts you for the password. If you don't specify *`username`*, `su` assumes that you're trying to log in as root and expects you to enter the root password.
`login` *`username`*	Almost the same as `su`, except that leaving off *`username`* merely places you at the normal login prompt.
<Alt-F*x*>	Lets you use the virtual terminals. You can access a virtual terminal by pressing the <Alt> key and one of the function keys (F1 through F6). This takes you to another login screen, where you can log in as the new user. The best feature of using the virtual terminals is that you're still left in the other account and can swap back and forth by using the <Alt-F*x*> keys; you aren't logged out of the other account.

NOTE If you try to add a user later from the account you now create, you may not be able to use the command `adduser` because certain commands can be entered only by the superuser, `adduser` being one of them. If you have trouble adding a user to the system, make sure that you're logged in as root.

Part I Ch 5

Adding Users with Red Hat

Red Hat Linux automates many of the `adduser` functions. To add a new user from the command line, enter the following command:

```
[root@web /root]#adduser jack
```

This command is a shell script located in /usr/sbin. You must be the superuser—that is, root—to issue this command.

▷ **See** "Working with Shell Scripts," **p. 350**

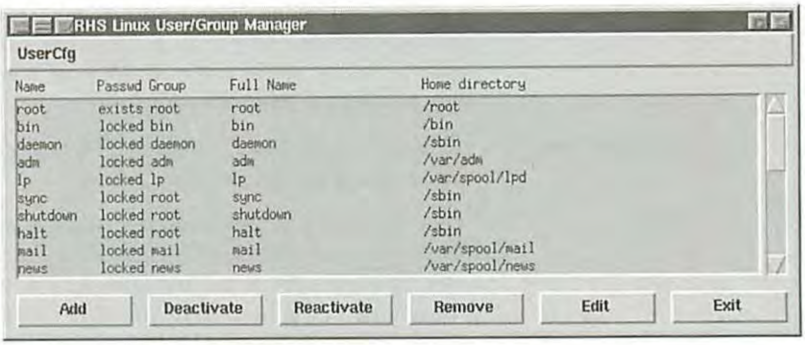
The script, which is just an ASCII file, creates the necessary directories and files needed by the new user. The only thing left is to set the user's password when he or she first logs in. Changing passwords is discussed later in the section "Changing Passwords."

▷ **See** "Viewing the Contents of a File," **p. 309**

Using Red Hat's Control Panel to Manage Users

If you installed XFree86 with your Red Hat installation, you can use the Control Panel's User/Group Manager configuration window (see Figure 5.1) to add users, modify user settings, and delete or deactivate users. To manipulate a user's account, simply select the user in the dialog box and click the appropriate button. Table 5.2 describes each button's function.

FIG. 5.1
The RHS Linux User/Group Manager dialog lets you see the information stored in /etc/passwd as well as manipulate the information.

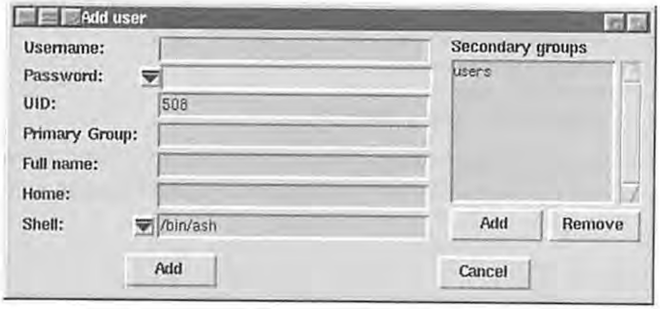


Name	Passwd	Group	Full Name	Home directory
root	exists	root	root	/root
bin	locked	bin	bin	/bin
daemon	locked	daemon	daemon	/sbin
adm	locked	adm	adm	/var/adm
lp	locked	lp	lp	/var/spool/lpd
sync	locked	root	sync	/sbin
shutdown	locked	root	shutdown	/sbin
halt	locked	root	halt	/sbin
mail	locked	mail	mail	/var/spool/mail
news	locked	news	news	/var/spool/news

Table 5.2 The RHS Linux User/Group Manager Buttons

Button	Description
Add	Displays the Add User dialog box, which lets you set the various required attributes for a user, such as home directory and password.
Deactivate	Allows you to deactivate the account of a user who you know will need the account again in the future. You may want to deactivate an account for a user who's on sabbatical or is being disciplined for some infraction. You can choose to compress the user's files to save space on your system until you reactivate that user later.
Reactivate	Allows you to reactivate a user's account.
Remove	Deletes a user from your system. The user's various files and directories will be removed. You may want to back up these files before deleting them.
Edit	Allows you to edit user accounts for such items as passwords (if they forget their password), their groups, or the shell they want to use.
Exit	Exits the RHS Linux User/Group Manager.

Clicking the Add button displays the Add User dialog box shown in Figure 5.2. You can set up the user's account from this dialog box by filling in the information for the various fields. Table 5.3 describes the fields and their function.



Username:
Password:
UID: 508
Primary Group:
Full name:
Home:
Shell: /bin/ash
Secondary groups: users

FIG. 5.2
Red Hat's graphical admin tools make adding new users a breeze.

Table 5.3 The Options of the Add User Dialog Box

Field	Description
Username	The name the user uses to log in to your system.
Password	The user's password. To give the user a password, you must use the combo box arrow and choose Edit from the menu. This displays a new dialog box that lets you enter a new password for the user. The password combo box also lets you blank out the password field by selecting the "none" value, or lock the password.
UID	A field generated by the system. For more information on UIDs and groups, see Chapter 10, "Managing User Accounts."
Primary Group	The primary group to which the user belongs. Groups allow you to put users into similar groupings, all having the same permissions.
Full Name	The user's full name.
Home	The user's home directory. Typically, it's located in a directory under /home or /usr/home.
Shell	The default shell the user's account starts with. This combo box allows you to select any shell offered by the Red Hat system for the user.

▶ **See** "Working with Groups," **p. 225**

Changing Passwords

In the future, you may want to change your password or add a password to an account that doesn't have one, such as the current root account. You should always password-protect the root account.

To change a password under any version of Linux or UNIX, you use the `passwd` command, specify the old and new password, and then verify the new password. If you don't have (or—worse—don't remember) the old password, you can't use the `passwd` command to change your password. The typical sequence for `passwd` follows:

```
[tackett@web ~]$ passwd
Changing password for jack
Enter old password: password
Enter new password: new-password
Re-type new password: new-password
```

If you make an error, Linux informs you that the password hasn't been changed. Linux also requires at least six characters for a valid password, and this minimum is enforced.

CAUTION

Don't forget your passwords! If you forget a user password, you must change the account information. If you forget the root account password, you must use the boot floppy created during installation to boot the system and change the password. Typically, you can set the password to empty by selecting none in the RHS Add/Edit User dialog box, and then let the user set a new password with the passwd command. You can also edit the /etc/passwd file and remove the emncrypted password from the user's record.

▶ **See** "Setting User Passwords," **p. 223**

Using Basic Commands

You need to know some basic commands to get around the system. The following sections provide some of the commands you need to use your Linux system. Finally, many of the "commands" presented in the following sections are actually utility programs that Linux uses to extend its command set. These programs are found in the /bin, /sbin, and /usr/bin directories.

Getting Help for Commands with *man*

To get online help for each of the various Linux commands, you can type **man**. Linux then displays, a screen at a time, any information it has on the command. If you aren't sure of what command to use, you can try the -k parameter and enter a simple keyword that represents the topic of interest. man then searches through its help files (called man, or manual, pages) for a topic that contains the keyword. Linux also provides an alias for this command, called apropos.

If you enter the command **man ls**, Linux provides help on the ls command, including all its parameters. The command man -k cls provides a listing of commands that have the word cls in the help file; the command apropos cls is the same as man -k cls.

Using Directory-Manipulation Commands

Linux provides many commands to work with directories. Like other operating systems you may have used, Linux allows you to create, delete, and move directories, as well as display information about the directory.

Changing the Current Working Directory with *cd* Linux, like DOS and other operating systems, stores files in a tree structure called a directory. You can specify a file via a path from the root directory, specified with the / character, to the file itself. Thus, the configuration file for `emacs` for the user jack can be exactly specified like so:

```
/home/jack/.emacs
```

If you're familiar with the DOS limits of eight characters for a file name and three characters for an extension, you'll be pleasantly surprised to learn that Linux has no such limit on file names.

▶ **See** "Understanding File and Path Names," **p. 288**

Linux also uses the concept of a home directory, which is specified when an account is added to the system. A user's home directory is usually specified with the ~ (tilde) character. You can use the tilde in place of the directory name, where the user wants to copy a file from the current directory \usr\home\jack to his or her home directory:

```
cp .emacs ~
```

To move around the Linux directory structure, you use the change directory command, `cd`. If you enter `cd` without any parameters, Linux immediately returns you to your home directory. To move from one directory to another directory, you use the `cd` command much as you do in DOS—that is, `cd` *`new-directory`*. Linux also uses the single . (dot) to represent the current directory and the .. to represent the parent directory. In fact, it's DOS that emulates UNIX, not UNIX/Linux emulating DOS.

NOTE Be careful how you specify the directory separator. DOS uses as its directory separator the \ (backslash) character, which Linux uses as the character for continuing a command on another line. To separate directory names in Linux, you must use the / (forward slash) character.

Also, although DOS doesn't mind if you fail to use spaces when specifying the . and .. parameters, Linux does. Linux doesn't understand `cd..`, but it understands `cd ..`. Linux needs the space separating the command and the parameter.

Displaying Information About Files and Directories with *ls* `ls` stands for *list* and is used by Linux to display a list of files. This command is the counterpart to the DOS `DIR` command. (Linux also accepts the `dir` command to list files in a directory.) Under Linux, the `ls` command displays all the main files in a directory in color. By default, blue indicates directories and green indicates executable programs. You can change the default colors by modifying the file /etc/DIR_COLORS.

▶ **See** "Listing Files," **p. 302**

`ls` takes many parameters to specify not only how to display a file but what files to display. The most common parameter is `-la`, which tells `ls` to display information in a long format for every file in a directory.

The command `ls -la` lists all information about every file in the current directory. The command `ls .emacs` lists the file .emacs, whereas `ls -l .emacs` lists all information about the file .emacs.

Creating New Directories with *mkdir* Because Linux's file system is based on directories, Linux provides the `mkdir` command so users can create new ones. Unlike DOS, which has an alias for the `mkdir` command called `MD`, Linux requires that the full `mkdir` command be spelled out. You must specify a name for each new directory, as shown in the following example:

```
mkdir backup
```

NOTE Linux does provide a way, via the command shell, to make aliases for command names; thus, if you simply can't live without the DOS `MD` command and hate typing **`mkdir`**, you can alias `md` to the `mkdir` command.

▶ **See** "Aliasing Commands," **p. 349**

Deleting Directories with *rmdir* The `rmdir` command deletes Linux directories. The command takes the name of the directory to delete. This directory must be empty—otherwise, Linux can't remove it.

For example, if the /backup directory had two directories within it, the command `rmdir /backup` fails. The command `rmdir /backup/jack/*` removes all files in the /backup/jack directory, and then `rmdir /backup/jack` removes the now-empty /backup/jack directory.

CAUTION

You can't delete a directory that contains files with the `rmdir` command. Instead, you can use the `-r` flag to the `rm` command. For example,

```
rm -r *
```

deletes everything from the current directory and every directory below the current directory. Be very careful using this command, because the moment you delete a directory, you can't recover the directory or the files that were located in the directory. Make backups.

Using File-Manipulation Commands

Because Linux treats directories and files similarly, it provides similar commands for manipulation.

Copying Files with *cp* The `cp` command is similar to the DOS `COPY` command. You use this command to copy one or more files from one directory to another directory. The syntax of `cp` is

```
cp from-filename to-filename
```

You must supply a *`from-filename`* and a *`to-filename`* for the files to be copied. If you want to preserve the file name, use the dot (.) as a placeholder for the *`to-filename`* parameter. This is in contrast to DOS, where you could leave off the *`to-filename`*.

The command `cp fred1 fred1.old` copies the file fred1 to a backup file named fred1.old, whereas the command `cp ~fred1.old /backup/jack` copies the file fred1.old from the home directory to the /backup/jack directory. (The ~ character represents the user's home directory.)

Moving Files with *mv* The `mv` command, which is similar to the DOS `MOVE` command, allows you to move files from one directory to another directory. When you move a file, it has the same effect as if you had copied the files to a new directory and then deleted the files in the old directory. `mv` doesn't make a copy of the files.

The syntax of the `mv` command is identical to the `cp` command:

```
mv from-filename to-filename
```

The command `mv fred1 fred1.old` copies the file fred1 to a backup file named fred1.old and deletes the old fred1 file, whereas the command `mv ~fred1.old /backup/jack` moves the fred1.old file from the home directory to the /backup/jack directory.

Deleting Files with *rm* To delete files under Linux, you use the `rm` command. The `rm` command is dangerous because as soon as a file is deleted, you can never recover it. For safety reasons, you should use the following form of the `rm` command:

```
rm -i filename
```

The `-i` parameter tells the command to query, or inquire, the user to see if that's the file they really want to remove. For example, the command `rm fred1` removes the file named fred1, whereas the command `rm -i fred1` deletes the fred1 file after asking whether the user really wants to remove this file.

CAUTION

As soon as you delete a file under Linux, that file is gone. You can't undelete a file or directory under Linux like you can with DOS. If you delete a file, your only hope is a backup copy.

Displaying File Contents with *more* The `more` command displays a screen of a text file. You can look through a text file without invoking an editor, printing the file, or trying to pause the terminal as it displays the file. To display the contents of your `emacs` configuration file, for example, you can type the following:

```
more .emacs
```

NOTE If you try to pass a binary data file to `more`, you could have some unpleasant effects—for example, your terminal can lock up. If your terminal does lock up, try pressing <Ctrl-q> or <Ctrl-s>.

A disadvantage with `more` is that you can't back up to see a screen of information once it passes. But the command discussed in the following section overcomes that problem.

Using *less*—a Better *more* `less` displays information a screen at a time on your terminal. The program's name is a play on words for the program it's meant to replace—`more`. Like `more`, `less` can display a screen of information in a text file, but unlike `more`, `less` allows you to page back and forth within the file. You can use the following command to browse through the readme file located in the info directory:

```
less /info/readme
```

Clearing the Screen with *clear* Sometimes after filling your terminal screen with information, you want a blank screen while you sit and contemplate your next action. Under DOS, you can use the `cls` command, but under Linux, you must use the `clear` command.

Dealing with DOS Files Under Linux

During installation, you were given the chance to make any DOS partitions you had available visible to Linux. These partitions were then placed in a directory you specified during configuration—for example, /dosc.

▶ **See** "Repartitioning the Drive," **p. 85**

If you want to copy these files to a floppy, using the `cp` command may cause problems because UNIX and Linux treat text files a little differently than DOS, especially when dealing with carriage returns and line-feeds. To overcome this problem, a group of programs were developed to help deal with MS-DOS files under a UNIX environment. These are the `m-` commands, which include such commands as `mcopy` and `mdir`. `mcopy` works just like the DOS `COPY` command, and `mdir` provides a directory listing. As you may notice, they resemble their DOS counterparts, except that they begin with the letter m, hence the name "`m-` commands." The `m-` commands are part of the `mtools` package, which is a collection of public-domain programs that allows UNIX to interact with DOS files much more easily.

These commands also make copying files to floppy disks much easier because you can use the DOS designation, like A, rather than the Linux designation /dev/fd0. For more information on the `m-` commands, enter

```
man mtools
```

Table 5.4 provides a brief listing of the various `m-` commands.

Table 5.4 The *m-* Commands

Command	Description
`mattrib`	Displays the file attributes for the specified file(s)
`mcd`	Changes directory to the specified path
`mcopy`	Copies the files specified to the new path

Command	Description
`mdel`	Deletes the specified files
`mdir`	Provides a directory listing
`mformat`	Formats a floppy
`mlabel`	Labels the DOS file system
`mmd`	Makes a directory
`mrd`	Removes a directory (must be empty, just as in DOS)
`mren`	Renames an existing DOS file
`mtype`	Displays the text contents of a DOS file

NOTE Although you can see a DOS file with Linux and even do some editing on text files in DOS partitions that Linux can see, you can't execute DOS or Windows programs under Linux. However, projects are under way across the Internet to supply such emulation for Linux. Although the prospects look very good for such emulators in the future, at this time DOS and Windows emulation isn't fully available. You'll have a brief introduction to both items later in this chapter.

Shutting Down Linux

When you're finished using a DOS machine, you can typically just turn off the power and walk away. You could also do the same under Windows, although there's a great possibility for file damage. Under Linux, there are even more chances for damaging your system, both to hardware and file systems, by simply turning off the power. You must shut down Linux in an orderly fashion, or you might corrupt the operating system to the point where it can't boot the next time you try.

Linux keeps a lot of information about itself and files in memory, in areas called ***buffers***, before writing the information to disk. This process helps improve system performance and control access to the hardware—something a multitasking operating systems needs to maintain so that one user doesn't try to use a hardware device that another user is using. If you turn off the power, this information is lost and you can corrupt your file system.

▶ **See** "Shutting Down Linux," **p. 218**

Because Linux is a multiuser and multitasking operating system, it must make sure that every user stops processing gracefully and save any work in progress before shutting the system down, to prevent data loss and file damage. This also gives each user logged in to the system time to log out. To shut down Linux in an orderly fashion, you must use the `shutdown` command. The `shutdown` command syntax is

```
shutdown [-r] time-to-shutdown [message]
```

The optional `-r` flag indicates that Linux should immediately reboot after it shuts down. This is useful if you want to quit Linux and boot to another operating system.

`time-to-shutdown` indicates when the system should shut down. The time is specified on a 24-hour clock, so you can tell the machine to shut down at 11 p.m. by entering

```
shutdown 23:00
```

The *`message`* parameter is a message sent to each user logged in to the system. This message is displayed on their terminals. You can use this message to tell users why you're shutting down the system. For example, if you needed to do weekly backups, you can use the following message to make sure that everyone logs out of the system:

```
[root@web /root]# shutdown -r 23:00 Shutting down at 11:00pm for system
maintenance
```

Remember, don't simply turn off the computer or press the reset button to exit Linux.

CAUTION

On some systems, Linux traps the <Ctrl-Alt-Del> reboot keystroke and executes an orderly shutdown as though the user had typed the `shutdown` command. However, on some systems Linux can't detect this keystroke combination and reboots immediately.

If you do accidentally turn off your system and damage the file structure, you can use the `fsck` command to try and repair the file system.

▶ **See** "Using the *fsck* Command," **p. 275**

Running Linux Programs

When you're familiar with moving around Linux and executing basic commands, you can try several applications installed when you set up the system. These applications cover a broad range of utilities, from a calculator to full-featured C and C++ compilers. Some of these programs cost hundreds of dollars; thanks to the GNU philosophy, however, many are readily available, and the only monetary outlay is the cost of getting the program from the Internet.

Luckily, many programs for Linux are also available on local bulletin boards, which you can reach via the telecommunications program included with the Slackware and Red Hat distributions of Linux. Also, many CD-ROM vendors supply CD-ROMs with hundreds of UNIX programs in source code. You can retrieve these programs from the CD-ROM and, by using the `gcc` and `g++` compilers distributed with Linux, get those programs up and running on your PC—even if you've never compiled a program before.

Finally, these programs are text-based and don't require the X Windows system to operate; thus, they may not have flashy graphics, but they work with most Linux installations.

Using the *workbone* CD Player

workbone is installed with the Slackware distribution. workbone is a text-based CD player written by Thomas McWilliams. If you have a CD-ROM capable of playing audio CDs, you should give it a try.

McWilliams wrote the program for his own enjoyment by hacking an X Windows-based program. Because he did this for his own enjoyment, workbone may not work correctly with every CD-ROM drive.

The program uses the numeric keypad to control the CD, so make sure that you have the <Num Lock> key engaged. Table 5.5 lists the various controls used.

Table 5.5 *workbone* Commands

Key	Description
0	Exits workbone and leaves music playing
DEL	Displays the help screen
1	Goes backward 15 seconds
2	Aborts workbone and stops music
3	Goes forward 15 seconds
4	Goes to the Previous selection
5	Restarts the current selection
6	Goes to the next selection
7	Stops
8	Pauses/resumes
9	Plays

As workbone plays, the display updates the time and current selection. If you want to continue working while your CD plays, you have two choices:

- You can exit workbone and leave the music playing (key 0).
- If you want to keep the display up and running, you can simply switch to another virtual terminal via the <Alt> key and log in to another account. When you want to check on the display, you can switch back to the proper virtual terminal and check on the status of the CD.

You can also stop the CD with the 0 key and then later simply re-execute the program to see what tracks are playing. For more information, check out the man page by typing **man workbone**.

Using the *sc* Spreadsheet Calculator

Do spiffy computers sell software, or does software sell computers? This is an age-old question whose answer tends to side with the concept that the proper application can sell thousands of computers. When the program called VisiCalc entered the market, PC use in business exploded. Why? Because for years, business people had played what-if games with their businesses on pieces of paper called ledgers, or spreadsheets. VisiCalc was an electronic version of the paper spreadsheet; it revolutionized how business did its forecasting and planning. Today, the successors of VisiCalc, such as Microsoft Excel and Lotus 1-2-3, still carry on the legacy started by VisiCalc. In the world of Linux, sc carries on that same legacy.

sc is a spreadsheet calculator containing rows and columns of cells. Each cell can contain a numeric value, a label string, or an expression or formula that evaluates to a numeric value or label string. These label strings can also be based on other cells to form complex relationships across a multiple collection of information.

If you've worked with other spreadsheet programs, you should have no problem getting up to speed on using sc. If you do need help, you can run a tutorial program to help you learn by entering

```
sc /usr/lib/sc/tutorial.sc
```

This tutorial provides an excellent introduction to using sc. If you need a quick reference card, you can print one by entering

```
scqref ¦ lpr
```

The solid bar is referred to as a *pipe* because you're piping, or passing on, the results of one command, scqref, to another command, lpr.

NOTE Check out Chapter 21, "Printing," if you have any problems printing with Linux. The biggest problem you may face, besides the possibility of not being able to print at all, is a bad case of the *jaggies*, which are the stair-step effects caused by how UNIX/Linux treats carriage returns and line feeds versus how MS-DOS treats them when printing text files containing these characters.

For online help about sc, simply type **man sc**.

Using the *bc* Calculator

bc is a command-line calculator for those quick-and-dirty calculations. bc is actually a sophisticated programming language that allows you to evaluate arithmetic expressions interactively.

When executed, bc responds with a short copyright notice and then leaves you at the command prompt, a blank line. You can then enter simple addition and subtraction functions. You can also perform division and multiplication—however, the version of bc distributed with Linux truncates the result of division and multiplication operations. (This is one of the hazards to be aware of when dealing with GNU software.) bc is great for simple calculations, as long as you're aware of the possible problems with its division and multiplication operations.

Another great feature is bc's capability of storing values from one operation to the next with a simple syntax, *variable-name = expression*. The following example calculates the value of 125 * 5 and stores the result in the var1 variable. To see what the results of the calculation are, you can type the name of the variable and bc prints the value on the next line, as shown in the example. Next, the example sets the variable var2 to the contents of var1 divided by 5.

```
var1 = 125 * 5
var1
625
var2 = var1 / 5
var2
```

Using the *minicom* Telecommunications Package

Let's hope, after having read the chapters in Part V, "Network Administration," that you can get your Linux system up and running on the Internet, the global Information Superhighway so much in the news today. Until then, you can still connect with the rest of the world if you have a modem and a telecommunications package. Linux supplies the package, called minicom, so all you have to do is supply the modem connected to one of your serial ports.

minicom, like a lot of Linux software, was written by a single person with help from many people on the Internet. The main author of minicom is Miquel van Smoorenburg. minicom is a very robust application that rivals many other commercial applications. With it you can connect to various bulletin-board services, maintain a list of numbers to dial, and download and upload files as soon as you connect. Help for most of minicom's functionality is available on the man page.

The first item to remember is that minicom uses the key sequence <Ctrl-Shift-a> to access the various functions, such as auto-dial and file downloading. To get help at anytime while in minicom, simply press <Ctrl-a><z> to display a brief command summary screen. Table 5.6 lists a few of those commands.

Table 5.6 *minicom* Command Summary

Key	Description
D	Dialing directory
S	Sends files
P	Communication parameters
L	Toggles on or off capturing the session to a file
F	Sends a BREAK to the other terminal
T	Sets terminal emulation between vt100, Minix, or ANSI
W	Toggles line wraps on or off

continues

Table 5.6 Continued

Key	Description
G	Runs a minicom script file
R	Receives a file
A	Adds a line-feed character to the end of lines
H	Hangs up the phone line
M	Initializes the modem
K	Runs the Kermit protocol
E	Toggles on or off local echo
C	Clears the local screen
O	Allows you to configure minicom
J	Jumps to a new command shell
X	Quits and resets the modem
I	Cursor key mode
Z	Displays the help screen
B	Scrolls back through the terminal window

While in the help window, you can simply press the appropriate letter to execute the command. From the minicom program, however, you must preface the appropriate letters with <Ctrl-a>.

minicom has four file-transfer protocols: zmodem, ymodem, xmodem, and kermit. If possible, you should try to use zmodem because of its superior error-recovery capabilities. If zmodem isn't available on the other system you're dialing, you should try each of the protocols in the order given. This isn't to say kermit is a bad protocol (it's not)—it's just slower than most of the others. The upside to using kermit is that more than likely almost any system you log in to supports kermit.

The next area you should be aware of is that minicom takes advantage of some commands that gives it access to the same type of power controlled by the superuser; thus, anyone running minicom has access to certain features of Linux that you may not want them to have.

▶ **See** "Handling File Security," **p. 242**

Playing Games

If you installed the y package, you have access to myriad games. Most are text-based, so you don't need the X Windows system up and running to enjoy a few minutes of fun. To get an idea of the variety of games, check out the /usr/games directory. By listing the files, you can see the available games. If you don't know what a game is or does, you can try to get help on the game with the `man` command. Of course, if you're adventurous, you can simply start the game and explore. Have fun!

Tetris

Tetris originated in the former Soviet Union. In the game, various shapes drop from the sky and pile up at the bottom of the screen. The object of the game is to eradicate those shapes building up and keep the game area from filling. You accomplish the elimination by completely filling a row across the playing field. When you connect one wall of the playing field with the other so that there are no gaps, that row disappears and all the shapes above it fall down to take up the vacated row. The catch to this strategy is that the shapes fall in a variety of patterns. To fill a row, you must decide how to orient a shape and then where to place it before it touches another block. When a shape touches another block, it remains at that position.

This game has been ported to most platforms, so if you've played the game on other systems, you should have no problem playing Tetris under Linux.

This version of the game is meant to be played only from the terminal, so don't expect fancy graphics. Also, the biggest pain is that on other systems you can position and orient the falling shapes with the keyboard arrow keys—not so with this version of Tetris. You must use the keys listed in Table 5.7 for positioning and orienting the various blocks.

Table 5.7 Tetris Command Keys

Command	Key
Move left	<,>
Move right	</>
Rotate	<.>
Drop	Space bar
Pause	<s>
Quit	<q>
Refresh the screen	<Ctrl-l>

Dungeon

Dungeon is a text adventure based on the ancient Adventure text games, but instead of caves, you deal with dungeons. You interact with this text-based world in search of treasures and adventure; if you've played other text adventures, this one is very similar. If you've used only glittery graphics, sit back and use the brain. You interact with the game by issuing commands and requests in the form of verbs and nouns. For instance, at the beginning of the game it tells you that you're in an open field west of a big, white house with a boarded front door. There's a small mailbox here. At the prompt, you can issue the following command to read whatever is in the box:

```
There is a small mailbox here.
> open box
Opening the mailbox reveals:
a leaflet.
> read leaflet
```

The game then provides a brief overview of the game and the talented programmers who built it. The last line of the information in the leaflet tells you to get assistance by entering the command `help` or `info`.

Trek

Trek is a text-based game based on the popular TV series *Star Trek*. Your goal is to survive the bloody battles with the Klingons and rid your star sector from their scourge. When you start the game by typing **`trek`**, you're asked a series of questions to set up the game:

- You're asked for the length of the game you want to play.
- You can restart a saved game from a logfile. To do so, you specify a logfile on the command line. This file name then becomes the name of the game saved.
- You're asked what skill level you want to play.
- You can enter a password so that others can't claim your glor y. No, really—you need a password so that no one but you can blow up your ship.

At any point—during setup or while playing—you can type a question mark to get help on the possible answers and actions available to you. Table 5.8 lists some of the possible actions.

Table 5.8 Trek Commands

Command	Description
`abandon`	Quits Trek
`damages`	Lists the damages your starship has sustained
`impulse`	Goes to impulse power
`ram`	Ramming speed
`srscan`	Short-range scan
`undock`	Leaves starbase
`capture`	Captures the Klingons
`destruct`	Self-destructs
`lrscan`	Long-range scan

Command	Description
`dump`	Who knows?
`visual`	Looks at the Klingons' position
`cloak`	Cloaks the ship
`dock`	Enters starbase
`move`	Plots and follows course
`rest`	Rests for a while
`terminate`	Quits
`Warp`	Engages warp engines
`computer`	Finds out some information
`help`	Calls a starbase for help
`phasers`	Fires phasers
`shields`	Shields up
`torpedo`	Fires torpedoes

The game begins by telling you how many Klingons are in your sector and how many starbases are here and their location. Docking at a starbase can replenish and repair your ship. Unfortunately, the game doesn't tell you where the nasty Klingon warships are located. Make sure that you pay attention to energy use; otherwise, you're in for some bad surprises.

Although this text-based game has no glitzy graphics, you can get a short-range scan with the `srscan` command, which displays your sector and all known objects at their respective coordinates. `srscan` also provides you with valuable information on the condition of your ship. All coordinates refer to a Cartesian matrix that you can maintain on paper—better yet, graph paper—so you don't have to remember from one `srscan` command to the other.

Running DOS Programs Under Linux

After you have enough of running various Linux applications, you occasionally might want to run some of your DOS or Windows programs. Although not yet a complete reality, work is progressing to allow you to do just that, by emulating the various operating systems under Linux. DOSEMU is a program that lets programs based on MS-DOS (and variants such as PC-DOS) run under Linux. DOSEMU stands for DOS EMUlator.

Also, a project is under way to allow users access to Windows programs under Linux. This project, called Wine, and is discussed later in "Running MS Windows Programs under Linux."

Installing DOSEMU

You can find the current version of DOSEMU on the accompanying Slackware CD-ROM under the name /contrib/dosemu_0.000 and contrib/dosemu_0.060. This file archive and its files need to be placed in the /usr/src directory and then unzipped and untarred by using the following commands:

```
[root@web src]# gzip -d dosemu_5.tgz
[root@web src]# tar -xvf dosemu_5.tar
```

Next, you must build the various files by using the following commands:

```
[root@web src]#make config
[root@web src]#make depend
[root@web src]#make most
```

These commands should install the DOSEMU files in the /var/lib/dosemu directory. You must be logged in as root and have at least 10M of virtual memory available for the build.

NOTE You must have installed package d, the program development package. You need the various compilers and tools within this package to build the DOS emulator.

Configuring DOSEMU

After the emulator is built, you must configure the system. To start, make a bootable DOS disk and copy the following DOS files on the disk: command.com, fdisk.exe, and sys.com.

Next, copy the following DOSEMU files from the dosemu subdirectory onto the floppy: emufs.sys, ems.sys, cdrom.sys, and exitemu.com. You can use the m- commands mentioned earlier in "Dealing with DOS Files Under Linux" to copy the files from the Linux partitions to the floppy drive.

TIP If you have trouble finding the Linux files, you can use the `find` command to locate the necessary files—for example,

```
find -name emufs.sys -print
```

This command will display the location of the file on your system, providing that it exists.

DOSEMU requires a configuration file, dosemu.conf, to operate correctly. You must customize this file for your system. You can find an example file in the examples directory on your system with the name config.dist. Listing 5.1 shows config.dist. Remarks are indicated with a pound symbol (#), and most options take the form of *parameter value*. If a parameter has more than one value, the values are placed within braces ({}).

Listing 5.1 A Sample dosemu.conf File

```
# Linux dosemu 0.51 configuration file.
# Updated to include QuickStart documentation 5/10/94 by Mark Rejhon
# James MacLean, jmaclean@fox.nstn.ns.ca, 12/31/93
# Robert Sanders, gt8134b@prism.gatech.edu, 5/16/93
#
# NOTICE:
#  - Although QuickStart information is included in this file, you
#    should refer to the documentation in the "doc" subdirectory of the
#    DOSEMU distribution, wherever possible.
#  - This configuration file is designed to be used as a base to make
#    it easier for you to set up DOSEMU for your specific system.
#  - Configuration options between lace brackets { } can be split onto
#    multiple lines.
#  - Comments start with # or ; in column 1. (beginning of a line)
#  - Send Email to the jmaclean address above if you find any errors.

#************************ DEBUG ****************************************
#
# QuickStart:
#  This section is of interest mainly to programmers. This is useful if
#  you are having problems with DOSEMU and you want to enclose debug info
#  when you make bug reports to a member of the DOSEMU development team.
#  Simply set desired flags to "on" or "off", then redirect stderr of
#  DOSEMU to a file using "dos 2>debug" to record the debug information
#  if desired. Skip this section if you're only starting to set up.
#
debug { config off   disk  off   warning off   hardware off
port  off   read  off   general off   IPC   off
video  off   write  off   xms   off   ems   off
serial off   keyb  off   dpmi  off
printer off   mouse  off
}

#************************ MISCELLANEOUS ********************************
#
#  Want startup DOSEMU banner messages? Of course :-)
dosbanner on
#
#  timint is necessary for many programs to work.
timint on

#************************** KEYBOARD ***********************************
#
# QuickStart:
#  With the "layout" keyword, you can specify your country's keyboard
#  layout. The following layouts are implemented:
#    finnish       us       dvorak    sf
#    finnish_latin1  uk       sg       sf_latin1
#    gr         dk       sg_latin1  es
#    gr_latin1     dk_latin1  fr       es_latin1
#    be         no       fr_latin1
#  The us-layout is selected by default if the "layout" keyword is omitted.
#
#  The keyword "keybint" allows more accurate keyboard interrupts,
#  It is a bit unstable, but makes keyboard work better when set to "on".
#
#  The keyword "rawkeyboard" allows for accurate keyboard emulation for
```

continues

Listing 5.1 Continued

```
#  DOS programs, and is only activated when DOSEMU starts up at the
#  console. It only becomes a problem when DOSEMU prematurely exits
#  with a "Segmentation Fault" fatal error, because the keyboard would
#  have not been reset properly. In that case, you would have to reboot
#  your Linux system remotely, or using the RESET button. In reality,
#  this should never happen. But if it does, please do report to the
#  dosemu development team, of the problem and detailed circumstances,
#  we're trying our best! If you don't need near complete keyboard
#  emulation (needed by major software package), set it to "off".
#
keyboard { layout us keybint on rawkeyboard on }
# keyboard { layout gr-latin1 keybint on rawkeyboard on }
#
#  If DOSEMU speed is unimportant, and CPU time is very valuable to you,
#  you may want to set HogThreshold to a non-zero value. This means
#  the number of keypress requests in a row before CPU time is given
#  away from DOSEMU. A good value to use could be 10000.
#  A zero disables CPU hogging detection via keyboard requests.
#
HogThreshold 0

#**************************** SERIAL ************************************
#
# QuickStart:
#  You can specify up to 4 simultaneous serial ports here.
#  If more than one ports have the same IRQ, only one of those ports
#  can be used at the same time. Also, you can specify the com port,
#  base address, irq, and device path! The defaults are:
#    COM1 default is base 0x03F8, irq 4, and device /dev/cua0
#    COM2 default is base 0x02F8, irq 3, and device /dev/cua1
#    COM3 default is base 0x03E8, irq 4, and device /dev/cua2
#    COM4 default is base 0x02E8, irq 3, and device /dev/cua3
#  If the "com" keyword is omitted, the next unused COM port is assigned.
#  Also, remember, these are only how you want the ports to be emulated
#  in DOSEMU. That means what is COM3 on IRQ 5 in real DOS, can become
#  COM1 on IRQ 4 in DOSEMU!
#
#  Also, as an example of defaults, these two lines are functionally equal:
#  serial { com 1 mouse }
#  serial { com 1 mouse base 0x03F8 irq 4 device /dev/cua0 }
#
#  If you want to use a serial mouse with DOSEMU, the "mouse" keyword
#  should be specified in only one of the serial lines. (For PS/2
#  mice, it is not necessary, and device path is in mouse line instead.)
#
#  Uncomment/modify any of the following if you want to support a modem
#  (or any other serial device).
#serial { com 1 device /dev/modem }
#serial { com 2 device /dev/modem }
#serial { com 3 device /dev/modem }
#serial { com 4 device /dev/modem }
#serial { com 3 base 0x03E8 irq 5 device /dev/cua2 }
#
```

```
#  If you have a non-PS/2 mouse, uncomment/modify one of the following.
#serial { mouse com 1 device /dev/mouse }
#serial { mouse com 2 device /dev/mouse }
#
#  What type is your mouse? Uncomment one of the following.
#  Use the 'internaldriver' option with ps2 and busmouse options.
#mouse { microsoft }
#mouse { logitech }
#mouse { mmseries }
#mouse { mouseman }
#mouse { hitachi }
#mouse { mousesystems }
#mouse { busmouse }
#mouse { ps2 device /dev/mouse internaldriver }
#  The following line won't run for now, but I hope it will sometime
#mouse { mousesystems device /dev/mouse internaldriver cleardtr }

#************************ NETWORKING SUPPORT ******************************
#
#  Turn the following option 'on' if you require IPX/SPX emulation.
#  Therefore, there is no need to load IPX.COM within the DOS session.
#  The following option does not emulate LSL.COM, IPXODI.COM, etc.
#  NOTE: MUST HAVE IPX PROTOCOL ENABLED IN KERNEL !!
ipxsupport off
#
#  Enable Novell 8137->raw 802.3 translation hack in new packet driver.
#pktdriver novell_hack

#************************ VIDEO *******************************************
#
# !!WARNING!!: A LOT OF THIS VIDEO CODE IS ALPHA! IF YOU ENABLE GRAPHICS
# ON AN INCOMPATIBLE ADAPTOR, YOU COULD GET A BLANK SCREEN OR MESSY SCREEN
# EVEN AFTER EXITING DOSEMU. JUST REBOOT (BLINDLY) AND THEN MODIFY CONFIG.
#
# QuickStart:
#  Start with only text video using the following line, to get started.
#  then when DOSEMU is running, you can set up a better video configura-
#  tion.
#
# video { vga console }     # Use this line, if you are using VGA
# video { cga console }     # Use this line, if you are using CGA
# video { ega console }     # Use this line, if you are using EGA
# video { mda console }     # Use this line, if you are using MDA
#
# Even more basic, like on an xterm or over serial, use one of the
# following :
#
#  For Xterm
# video { vga chunks 25 }
#  For serial at 2400 baud
# video { vga chunks 200 }
#
# QuickStart Notes for Graphics:
#  - If your VGA-Bios resides at E000-EFFF, turn off video BIOS shadow
```

continues

Listing 5.1 Continued

```
#   for this address range and add the statement vbios_seg 0xe000
#   to the correct vios-statement, see the example below.
#  - Set "allowvideoportaccess on" earlier in this configuration file
#   if DOSEMU won't boot properly, such as hanging with a blank screen,
#   beeping, or the video card bootup message.
#  - Video BIOS shadowing (in your CMOS setup) at C000-CFFF must be dis-
#   abled.
#
#   *> CAUTION <*: TURN OFF VIDEO BIOS SHADOWING BEFORE ENABLING GRAPHICS!
#
#  It may be necessary to set this to "on" if DOSEMU can't boot up properly
#  on your system when it's set "off" and when graphics are enabled.
#  Note: May interfere with serial ports when using certain video boards.
allowvideoportaccess on
#
#  Any 100% compatible standard VGA card _MAY_ work with this:
#video { vga console graphics }
#
#  If your VGA-BIOS is at segment E000, this may work for you:
#video { vga console graphics vbios_seg 0xe000 }
#
#  Trident SVGA with 1 megabyte on board
#video { vga console graphics chipset trident memsize 1024 }
#
#  Diamond SVGA
#video { vga console graphics chipset diamond }
#
#  ET4000 SVGA card with 1 megabyte on board:
#video { vga console graphics chipset et4000 memsize 1024 }
#
#  S3-based SVGA video card with 1 megabyte on board:
#video { vga console graphics chipset s3 memsize 1024 }

#************************** MISCELLANEOUS *********************************
#
# QuickStart:
#  For "mathco", set this to "on" to enable the coprocessor during DOSEMU.
#  This really only has an effect on kernels prior to 1.0.3.
#  For "cpu", set this to the CPU you want recognized during DOSEMU.
#  For "bootA"/"bootC", set this to the bootup drive you want to use.
#  It is strongly recommended you start with "bootA" to get DOSEMU
#  going, and during configuration of DOSEMU to recognize hard disks.
#
mathco on      # Math coprocessor valid values: on off
cpu 80386      # CPU emulation valid values: 80286 80386 80486
bootA          # Startup drive valid values: bootA bootC

#************************ MEMORY *****************************************
#
# QuickStart:
#  These are memory parameters, stated in number of kilobytes.
#  If you get lots of disk swapping while DOSEMU runs, you should
#  reduce these values. Also, DPMI is still somewhat unstable,
#  (as of early April 1994) so be careful with DPMI parameters.
#
xms 1024       # XMS size in K, or "off"
ems 1024       # EMS size in K, or "off"
dpmi off       # DPMI size in K, or "off". Be careful with DPMI!

#********************** PORT ACCESS *************************************
#
# !!WARNING!!: GIVING ACCESS TO PORTS IS BOTH A SECURITY CONCERN AND
# SOME PORTS ARE DANGEROUS TO USE. PLEASE SKIP THIS SECTION, AND
# DON'T FIDDLE WITH THIS SECTION UNLESS YOU KNOW WHAT YOU'RE DOING.
#
# ports { 0x388 0x389 } # for SimEarth
# ports { 0x21e 0x22e 0x23e 0x24e 0x25e 0x26e 0x27e 0x28e 0x29e } # for
# jill

#******************* SPEAKER ********************************************
#
# These keywoards are allowable on the "speaker" line:
#  native   Enable DOSEMU direct access to the speaker ports.
#  emulated  Enable simple beeps at the terminal.
#  off     Disable speaker emulation.
#
speaker native     # or "off" or "emulated"

#******************* HARD DISKS *****************************************
#
# !!WARNING!!: DAMAGE MIGHT RESULT TO YOUR HARD DISK (LINUX AND/OR DOS)
# IF YOU FIDDLE WITH THIS SECTION WITHOUT KNOWING WHAT YOU'RE DOING!
#
# QuickStart:
#  The best way to get started is to start with a boot floppy, and set
#  "bootA" above in the configuration. Keep using the boot floppy
#  while you are setting this hard disk configuration up for DOSEMU,
#  and testing by using DIR C: or something like that.
#  If you want DOSEMU to be able to access a DOS partition, the
#  safer type of access is "partition" access, because "wholedisk"
#  access gives DOSEMU write access to a whole physical disk,
#  including any vulnerable Linux partitions on that drive!
#
#  !!! IMPORTANT !!!
#  You must not have LILO installed on the partition for dosemu to boot
#  off.
#  As of 04/26/94, doublespace and stacker 3.1 will work with wholedisk
#  or partition only access. Stacker 4.0 has been reported to work with
#  wholedisk access. If you want to use disk compression using partition
#  access, you will need to use the "mkpartition" command included with
#  dosemu to create a partition table datafile for dosemu.
#
#  Please read the documentation in the "doc" subdirectory for info
#  on how to set up access to real hard disk.
#
#  "image" specifies a hard disk image file.
#  "partition" specifies partition access, with device and partition
```

continues

Listing 5.1 Continued

```
#   number.
#  "wholedisk" specifies full access to entire hard drive.
#  "readonly" for read only access. A good idea to set up with.
#
#disk { image "/var/lib/dosemu/hdimage" }   # use diskimage file.
#disk { partition "/dev/hda1" 1 readonly }  # 1st partition on 1st IDE.
#disk { partition "/dev/sda2" 1 readonly }  # 1st partition on 2nd SCSI.
#disk { wholedisk "/dev/hda" }              # Entire disk drive unit

#******************* DOSEMU BOOT*****************************************
#
#  Use the following option to boot from the specified file, and then
#  once booted, have bootoff execute in autoexec.bat. Thanks Ted :-).
#  Notice it follows a typical floppy spec. To create this file use
#  dd if=/dev/fd0 of=/var/lib/dosemu/bdisk bs=16k
#
#bootdisk { heads 2 sectors 18 tracks 80 threeinch file /var/lib/dosemu/#bdisk }
#
#  Specify extensions for the CONFIG and AUTOEXEC files. If the below
#  are uncommented, the extensions become CONFIG.EMU and AUTOEXEC.EMU.
#  NOTE: this feature may affect file naming even after boot time.
#  If you use MSDOS 6+, you may want to use a CONFIG.SYS menu instead.
#
#EmuSys EMU
#EmuBat EMU

#******************* FLOPPY DISKS ***************************************
#
# QuickStart:
#  This part is fairly easy. Make sure that the first (/dev/fd0) and
#  second (/dev/fd1) floppy drives are of the correct size, "threeinch"
#  and/or "fiveinch". A floppy disk image can be used instead, however.
#
#  FOR SAFETY, UNMOUNT ALL FLOPPY DRIVES FROM YOUR FILESYSTEM BEFORE
#  STARTING UP DOSEMU! DAMAGE TO THE FLOPPY MAY RESULT OTHERWISE!
#
floppy { device /dev/fd0 threeinch }
floppy { device /dev/fd1 fiveinch }
#floppy { heads 2 sectors 18 tracks 80
#     threeinch file /var/lib/dosemu/diskimage }
#
#  If floppy disk speed is very important, uncomment the following
#  line. However, this makes the floppy drive a bit unstable. This
#  is best used if the floppies are write-protected.
#
#FastFloppy on

#******************* PRINTERS *******************************************
#
# QuickStart:
#  Printer is emulated by piping printer data to a file or via a unix
#  command such as "lpr". Don't bother fiddling with this configuration
#  until you've got DOSEMU up and running already.
#
```

```
#printer { options "%s" command "lpr" timeout 20 }
#printer { options "-p %s" command "lpr" timeout 10 }  # pr format it
#printer { file "lpt3" }
```

You must then use a text editor to change the settings from the example configuration file to match your system. Such items as processor type and video cards must match.

NOTE You can also boot DOSEMU from a hard drive partition, instead of from a floppy. To access a hard drive, simply configure a drive/partition in the dosemu.conf file.

Running DOSEMU

To run DOSEMU, simply type **dos** at any Linux prompt. To exit, use the `exitemu` command from the prompt. Table 5.9 provides a listing of command-line options you can pass to DOSEMU. You also can use `-?` to get a complete, up-to-date listing of command-line parameters.

Table 5.9 DOSEMU Command-Line Parameters

Parameter	Description
-A	Boot from the A drive
-C	Boot from the hard drive
-c	Optimize video performance from virtual terminals
-D	Set debug options
-e	Specify the amount of EMS memory
-F#	Number (#) of floppies to use from dosemu.conf
-f	Flip the definition of the A and B floppy drives
-H#	Number (#) of hard disks to use from dosemu.conf
-k	Use the raw keyboard console defined in the `rawkeyboard` parameter of dosemu.conf
-P	Copy the debug information to a file
-t	Deliver the time interrupt 9
-V	Activate VGA emulation
-x	Specify the amount of XMS memory
-?	Display summary help for each command
-2	Emulate a 286
-3	Emulate a 386
-4	Emulate a 486

From the DOS prompt supplied by DOSEMU, you can run most DOS programs except those that require DPMI (DOS Protected Mode Interface) support. Simply type the name of the program and—providing that DOSEMU can find the program in your path—DOSEMU will load and run the program.

Table 5.10 shows some of the programs known to operate under Linux, but more are added every day (check the file EMUsuccess.txt, in the directory where DOSEMU was installed, for an up-to-date listing). Table 5.11 lists some of the programs that don't work with Linux.

Table 5.10 Programs Known to Run with DOSEMU

		Success Story
Name	**Function**	**Posted by**
1st Wordplus	GEM word processor	**jan@janhh.hanse.de**
4desc	4dos desc editor	**piola@di.unito.it**
4DOS 4.2	Command interp.	**rideau@clipper.ens.fr**
4dos 5.0c	Command interp.	**J1MCPHER@VAXC.STEVENS-TECH.EDU**
ack3d	3-D engine	**martin5@trgcorp.solucorp.qc.ca**
ACU-COBOL	Compiler	**fjh@munta.cs.mu.OZ.AU**
Alite 1.10		**ph99jh42@uwrf.edu**
AmTax 93 & 94	Tax software	**root@bobspc.canisius.edu**
ansi.sys	Screen/keyboard driver (display functions)	**ag173@cleveland.Freenet.Edu**
arj v2.41a	[Un]archiver	**tanner@winternet.mpls.mn.us**
As Easy As 5.01	Spreadsheet	**ph99jh42@uwrf.edu**
Autoroute Plus	Route planner	**hsw1@papa.attmail.com**
Axum	Sci. graphics	**miguel@pinon.ccu.uniovi.es**
battle chess	Chess game	**jvdbergh@wins.uia.ac.be**
Binkley 2.50eebd	Fidomailer	**stub@linux.rz.tu-clausthal.de**
Blake Stone_	Game	**owaddell@cs.indiana.edu**
bnu 1.70	Fossil (Fido)	**stub@linux.rz.tu-clausthal.de**

Name	Function	Success Story Posted by
Borland C++ 2.0	86/286 C/C++ IDE	rideau@clipper.ens.fr
Boston Business EDT+		keegstra@csdr2.fsfc. nasa.gov
Cardbox Plus	Database	hsw1@papa.attmail.com
Castle Wolfenstein	3-D game	gt8134b@prism.gatech.EDU
Checkit diagnostics		
clipper 5.1	dBASE compiler	jvdbergh@wins.uia.ac.be
COMPRESS	Compressed fs	rideau@clipper.ens.fr
CCM (Crosstalk)	Modem program	
cshow 8.61	Picture viewer	jvdbergh@wins.uia.ac.be
cview	Picture viewer	lotov@avarice.ugcs. caltech.edu
d86/a86		
DataPerfect 2.1	Database	fbennett@uk.ac.ulcc.clus1
Dbase 4		corey@amiganet.xnet.com
Derive 1.2	Math package	miguel@pinon.ccu. uniovi.es
Disk Freedom 4.6	Disk utility	
diet 1.45f	File compression	stub@linux.rz.tu-clausthal.de
dosnix 2.0	UNIX utilities	miguel@pinon.ccu. uniovi.es
Dosshell task	Swapper	jmaclean@fox.nstn.ns.ca
dtmm	Molecular models	miguel@pinon.ccu. uniovi.es
Dune 2	Game	COLIN@fs1.in.umist.ac.uk
dviscr	EMTEX dvi preview	ub9x@rz.uni-karlsruhe.de
Easytrax	Layout editor	maehler@wrcd1.urz. uni-wuppertal.de
Elvis	`vi` clone	miguel@pinon.ccu. uniovi.es

continues

Table 5.10 Continued

Name	Function	Success Story Posted by
Epic Pinball	Game	krismon@quack.kfu.com
ETen 3.1	Chinese terminal	tyuan!root@mp.cs.niu.edu
Eureka 1.0	Math package	miguel@pinon.ccu. uniovi.es
Falcon 3.0	Fighter simulator	rapatel@rockypc. rutgers.edu
FastLST 1.03	FidoNdlstcompiler	stub@linux.rz.tu-clausthal.de
FormGen II		root@bobspc.canisius.edu
freemacs 1.6d	Editor	ph99jh42@uwrf.edu
Frontier (Elite II)	Game	COLIN@fs1.in.umist.ac.uk
FW3		Sebastian.Bunka@ vu-wien.ac.at
MS Flight Simulator 5	Game (runs *slow*!)	newcombe@aa.csc. peachnet.edu
Foxpro 2.0	Database	
Framework 4		corey@amiganet.xnet.com
Freelance Graphics 2.1	Graph/drawing application	jwest@jwest.ecen. okstate.edu
GEM/3	GUI	jan@janhh.hanse.de
GEM Draw	GEM drawing app	jan@janhh.hanse.de
GEM Paint	GEM painting app	jan@janhh.hanse.de
gmouse	Mouse driver	tk@pssparc2.oc.com
God of Thunder	Game	ensor@cs.utk.edu
Gravity	Simulation package	miguel@pinon.ccu. uniovi.es
GWS for DOS	Graphic file conv	bchow@bchow.slip
Gzip 1.1.2	File compression	miguel@pinon.ccu. uniovi.es
Harpoon	Game	wielinga@physics.uq.oz.au

Name	Function	Success Story Posted by
Harvard Graphics 3.0	Graph/drawing package	miguel@pinon.ccu.uniovi.es
Hero's Quest I	Game	lam836@cs.cuhk.hk
Hijaak 2.0	Graphic file conv	bchow@bchow.slip
hocus pocus	Apogee game	kooper@dutiws.TWI.TUDelft.NL
Image Alchemy Pro (-v doesn't work)	Graphic file conv	J1MCPHER@VAXC.STEVENS-TECH.EDU
Incredible Machine	Game (slow)	sdh@po.cwru.edu
Key Spreadsheet Plus	Spreadsheet (on non-doublespaced disks)	jwest@jwest.ecen.okstate.edu
Lemmings		sdh@po.cwru.edu
less 1.7.7	More than more	miguel@pinon.ccu.uniovi.es
LHA	File compression	
Lotus Manuscript	Word processor	miguel@pinon.ccu.uniovi.es
Managing Your Money	Financial	newcombe@aa.csc.peachnet.edu
Manifest	(dies during memory timings)	hsw1@papa.attmail.com
Mathcad 2.01	Math package	root@bobspc.canisius.edu
MathCad 2.06	Math package	miguel@pinon.ccu.uniovi.es
mcafee 9.23 v112	Virus scanner	jvdbergh@wins.uia.ac.be
Microemacs	Editor	hjstein@MATH.HUJI.AC.IL
MicroLink Yaht 2.1		root@bobspc.canisius.edu
Microsoft C 6.0	Compiler	ronnie@epact.se
Microsoft Assembler 5.0	Assembler	ronnie@epact.se
Microsoft Library 2.0		root@bobspc.canisius.edu

continues

Table 5.10 Continued

Name	Function	Success Story Posted by
Microsoft Make	make	ronnie@epact.se
MicrosoftMouse Drv 8.2	Mouse driver	hswl@papa.attmail.com
MoneyCounts 7.0	Accounting package	raeburn@cygnus.com
mscmouse	Mouse driver	tk@pssparc2.oc.com
nnansi.com	ANSI driver	mdrejhon@undergrad.math.uwaterloo.ca
Netzplan	GEM project mgr	jan@janhh.hanse.de
NHL Hockey	Game	krismon@quack.kfu.com
NJStar 2.1	Chinese word proc	aab2@cornell.edu
Norton Utils 4.5	Disk utils	rideau@clipper.ens.fr
Norton Utils 7.0	Disk utils	rideau@clipper.ens.fr
PAF	Geneology package	geek+@CMU.EDU
Paradox	Database	hp@vmars.tuwien.ac.at
PC Paintbrush IV	Paint program	bchow@bchow.slip
PCtools 4.20	Disk utils	rideau@clipper.ens.fr
pcwdemo		vinod@cse.iitb.ernet.in
PC-Write 3.0	Word processor	
pcxlab 1.03	PCX viewer	miguel@pinon.ccu.uniovi.es
peachtree complete 6.0	Accounting	stjeanp@math.enmu.edu
Pinball Dreams	Game	ronnie@lysator.liu.se
PKzip/unzip	File compression	
pklite 1.15	File compression	stub@linux.rz.tu-clausthal.de
Pong Kombat	Game	ensor@cs.utk.edu
PrintShop	Greeting card pkg	geek+@CMU.EDU
Procomm Plus 2.0	Communication	newcombe@aa.csc.peachnet.edu

Name	Function	Success Story Posted by
Procomm 2.4.3	Communication	hsw1@papa.attmail.com
Pspice 5.0	Circuit sim.	root@bobspc.canisius.edu
Q&A	Word proc/database	newcombe@aa.csc. peachnet.edu
Qbasic/edit (from DOS 5.0)	Interpreter	
Qedit	Editor	
QuickC	Compiler	martin@trcsun3.eas. asu.edu
Quicken 4.0 for DOS	Accounting pkg	juphoff@nrao.edu
Quicken 6.0 for DOS	Accounting pkg	
Quicken 7.0 for DOS	Accounting pkg	juphoff@astro.phys. vt.edu
Railroad Tycoon		juphoff@astro.phys. vt.edu
Red Baron	Game	wielinga@physics. uq.oz.au
RM/COBOL	compiler	fjh@munta.cs.mu.OZ.AU
Rpro 1.6		root@bobspc.canisius.edu
scan109	Antivirus	miguel@pinon.ccu. uniovi.es
scan112	Antivirus	piola@di.unito.it
Scorch	Tank game	geek+@CMU.EDU
Shez94	Arcer-Shell	stub@linux.rz.tu-clausthal.de
sled	Editor	piola@di.unito.it
Space Quest IV	Game	lam836@cs.cuhk.hk
Spell Casting 301		mancini@phantom.com
SPSS/PC+4.0	Statistical pkg	jr@petz.han.de
Squish 1.01	Fido Scan/Tosser	stub@linux.rz.tu-clausthal.de

continues

Table 5.10 Continued

		Success Story
Name	**Function**	**Posted by**
Stacker 3.1	Compressed fs	mdrejhon@undergrad math.uwaterloo.ca
Stacker 4.00	Compressed fs	J1MCPHER@VAXC. STEVENS-TECH.EDU
StatPhys	Simulation pkg	miguel@pinon.ccu. uniovi.es
STSORBIT	Orbit simulation	troch@gandalf.rutgers.edu
Stunts	Game?	gt8134b@prism.gatech.EDU
Superstor	Compressed fs	rideau@clipper.ens.fr
TAG 2.02	Polish word proc	rzm@oso.chalmers.se
TASM 2.51	MACRO assembler	rideau@clipper.ens.fr
Telix	Modem program	jou@nematic.ep. nctu.edu.tw
THelp from BC++2.0	Popup help	rideau@clipper.ens.fr
TimED/beta	Fido MSGeditor	stub@linux.rz.tu-clausthal.de
TLINK 4.0	LINKER	rideau@clipper.ens.fr
Topspeed Modula-2	Compiler	mayersn@hermes. informatik.uni-stuttgart.de
Turbo Debugger 2.51	Realmode debugger	rideau@clipper.ens.fr
Turbo Pascal 5.5	Compiler	
Turbo Pascal 6.0	Compiler	t2262dj@cd1.lrz-muenchen.de
Turbo Pascal 7.0	Compiler	mdrejhon@undergrad. math.uwaterloo.ca
Turb-opoly 1.43		root@bobspc.canisius.edu
Ultima 6	Game	msphil@birds.wm.edu
Vpic 6.1		root@bobspc.canisius.edu
warlords II	Game	buckel@cip. informatik.uni-wuerzburg.de
Warrior of Destiny	Game	msphil@birds.wm.edu
WITWI Carmen	Game	tillemaj@cae.wisc.edu

Name	Function	Success Story Posted by
Sandiego		
Windows 3.0	Windows (real mode)	cjw1@ukc.ac.uk
Wolf3d	Game	owaddell@cs.indiana.edu
WordPerfect 5.1	Word processor	sdh@po.cwru.edu
WordPerfect 6.0	Word processor (needs >1M RAM)	lujian@texmd.minmet.mcgill.ca
Xtpro 1.1	Disk util	root@bobspc.canisius.edu
XWing	Game (very slow)	ronnie@lysator.liu.se
Zarkov 2.6	Chess	a-acero@uchicago.edu
zoo	File compression	

Table 5.11 Programs Known *Not* to Run with DOSEMU

Name	Function	Posted By
4D-box	Boxing game	jvdbergh@wins.uia.ac.be
Apple][emulator	Emulator	ph99jh42@uwrf.edu
Borland C++ 3.1 IDE	Compiler	juphoff@uppieland.async.vt.edu
brief	Editor	bchow@bchow.slip
Chuck Yeager Aircombat	Flight simulator	jvdbergh@wins.uia.ac.be
CIVILIZATION	Game	miguel@pinon.ccu.uniovi.es
DesqView 2.51 (Alt key doesn't work)		hsw1@papa.attmail.com
doom	Game	rideau@clipper.ens.fr
dpms from Stacker 4.0		J1MCPHER@VAXC.STEVENS-TECH.EDU
dxma0mod.sys	Token-ring driver	adjihc4@cti.ecp.fr
dxmc0mod.sys	Token-ring driver	adjihc4@cti.ecp.fr
ELDB	Economics database	hjstein@math.huji.ac.il

continues

Table 5.11 Continued

Name	Function	Posted By
FIPS 0.2.2	Disk util (hdimage FAT problem)	
Howitzer	Tank game	**geek+@CMU.EDU**
Lahey Fortran	Fortran compiler	**hjstein@math.huji.ac.il**
Maple V2	Math package	**ralf@ark.btbg.sub.de**
MSDOS 5/6 QBASIC/EDIT	Editor	**bchow@bchow.slip**
NORTON UTILITIES 7.0	Disk utils	**bchow@bchow.slip**
Quattro Pro 4.0	Spreadsheet	**jwest@jwest.ecen.okstate.edu**
Raptor	Game	**ensor@cs.utk.edu**
Silent Service II	Submarine game	**jvdbergh@wins.uia.ac.be**
thunderByte scan	Virus scanner	**jvdbergh@wins.uia.ac.be**
Ventura Publisher 3.0	Desktop pub	**niemann@swt.ruhr-uni-bochum.de**
wildunix	Wild cards	**miguel@pinon.ccu.uniovi.es**
Windows 3.1		**juphoff@uppieland.async.vt.edu**

Running programs under DOSEMU has several problems, mostly because the computer is emulating DOS and the underlying machine instead of actually running DOS. Emulation slows down the system. The slowdown can become annoying, especially when you're also running other Linux programs in other virtual terminals. Video updates are also rather slow under DOSEMU.

Many DOS programs hog the CPU, because they believe they're the only program running. This prevents other Linux programs from receiving access to the CPU. To alleviate this problem, Thomas G. McWilliams wrote a program called `garrot` to release access of the processor back to Linux from DOS-hogging programs. You can find `garrot` on the **sunsite.unc.edu** FTP site in the /pub/linux/alpha/dosemu directory.

Running Windows Programs Under Linux

DOSEMU can't run Microsoft Windows programs, so the Linux community has embarked on creating a program that will allow Linux users to run such programs. This Windows emulator is called Wine. Wine isn't a standard acronym; it can stand as WINdows Emulator or, because Wine can be built as a static library instead of an emulator, Wine Is Not a Windows Emulator. Both acronyms are from the Windows FAQ.

The Windows FAQ is required reading if you want to experiment with Wine, because Wine isn't as far along in development as DOSEMU. Thus, it's very experimental and error-prone. Also, not many Windows programs are supported. In fact, to use Wine, you must have Windows installed on a partition accessible to Linux, because Wine still relies on many parts of Windows to work. Wine also requires the X system to be installed and operational.

To experiment with Wine, you'll need the following:

- A Linux kernel, version 99.13 or above
- Source code for Wine, because it's available only in source code format
- The d package installed for the compiler tools to build the source code
- At least 8M of RAM and at least a 12M swap drive
- At least 10M of disk space
- X Windows installed and configured
- A pointing device such as a mouse
- Microsoft Windows installed on a partition accessible to Linux

Part I
Ch 5

Because Wine is under heavy development, new versions are released almost weekly. The newest source code is located at **sunsite.unc.edu** (and other major FTP sites) in the /pub/Linux/ALPHA/wine/development directory. The file is named after the date of its release—for example, wine-961201.tar.tgz.

▷ **See** "Using FTP for Remote File Transfer," **p. 540**

Because Wine is changing so fast and is so unstable, it's not included on the accompanying CD-ROMs. If you want to experiment with Wine, feel free to download the newest files and read over the FAQs and HOWTOs. These documents are located on the CD-ROM in the /docs directory and provide the information needed to compile, install, configure, and use Wine.

Installing Wine is very similar to installing DOSEMU, with the exception that you can place the source tar file anywhere. Use the `tar` command to unarchive the file in the directory—for example,

```
[root@web wine]# gzip -d 950606.tar.gz
[root@web wine]# tar -xvf 950606.tar
```

Building Wine is a little more involved than building DOSEMU—in fact, it's more like building a new kernel. You must answer several questions to configure the build process. The Wine HOWTO explains the full process in detail.

Next, you must answer several questions to configure Wine with runtime parameters. These configuration parameters are stored in a file named /usr/local/etc/wine.conf. Although you can edit this file by hand, it's best to use the supplied configure program to do so.

After you configure the compilation files and the runtime parameter file, you can build Wine with the simple command `make`. This process takes several minutes. To use Wine, you invoke the emulator and provide the path name to a Windows executable file—for example,

```
[tackett@web ~]
$wine /dosc/windows/winmine.exe
```

The programs now supported by Wine are calc.exe, clock.exe, notepad.exe, and winmine.exe. This list is continuously expanding, so check the FAQ and HOWTO for current programs supported by the Windows emulator.

NOTE MS-DOS and Microsoft Windows aren't the only operating systems emulated under Linux. There are also emulators for the old Apple II, CPM, and the newer Macintosh operating systems. You can generally find these emulators on FTP sites in the \pub\Linux\system\emulators directory. ■

From Here...

This chapter has just lightly touched on getting started with Linux and the various application programs available. For more information, see the following chapters:

- Chapter 7, "Using X Windows," deals with the graphical user interface provided with Linux, XFree86.
- Chapter 13, "Upgrading and Installing Software," explains how to install new software from CD-ROM or the Internet.
- Chapter 19, "Using the *vi* Editor," and Chapter 20, "Using the *emacs* Editor," discuss two of the more popular text editors for Linux.

PART III

Managing the File System

CHAPTER 14

Managing File Systems

File systems form the basis for all data on a Linux system. Linux programs, libraries, system files, and user files all reside on file systems. Proper management of file systems is critical because all your data and programs exist on top of file systems.

Many of the steps outlined in this chapter are performed automatically when you install Linux. However, you should learn to manage your file systems so that you can create, manage, and maintain your Linux system. Understanding file system management is critical to successful systems administration. Your file system must work properly for your Linux system to work at all.

Understanding file systems

File systems are directories created on hard drives, floppies, or CD-ROMs. File systems can also be available over the network.

Mounting and unmounting file systems

To access a file system, you must make its existence know to Linux by mounting the file system to a mount point.

Working with the Network File System (NFS)

NFS allows Linux to access file systems on remote machines as though they were on the local hard drive.

Maintaining file systems

Over time, Linux file systems can become stale or corrupted, and thus need to be maintained.

Creating and formatting file systems

You must create and format file systems as new hard drives are added.

Using swap files and partitions

To take advantage of virtual memory, Linux needs *swap space*—disk space the system uses as memory when physical RAM is exhausted.

Understanding File Systems

Under Linux, the file space that's visible to users is based on a tree structure, with the root at the top. The various directories and files in this space branch downward from the root. The top directory, /, is known as the *root directory.* Figure 14.1 gives a graphical example of a tree structure.

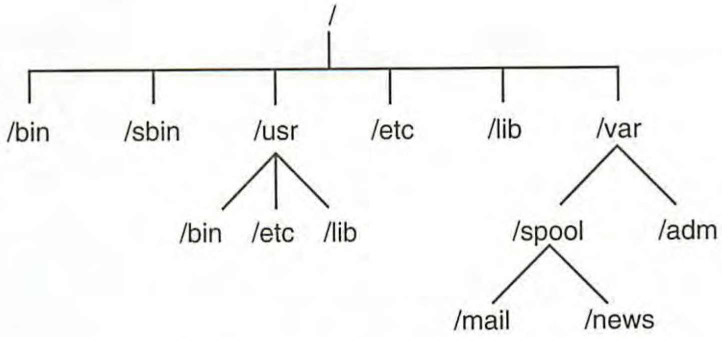


FIG. 14.1
Picture the Linux file system as an upside-down tree, with the root at the top and the branches and leaves spreading downward.

To users, this directory tree looks like a seamless entity—they just see directories and files. In reality, many of the directories in the file tree are physically located as different partitions on a disk, on different disks, or even on different computers. When one of these disk partitions is attached to the file tree at a directory known as a *mount point,* the mount point and all directories below it are referred to as a *file system.*

The Linux operating system is made up of several directories and many different files. Depending on how you selected your installation, these directories may be different file systems. Typically, most of the operating system resides on two file systems: the root file system, known as /, and a file system mounted under /usr (pronounced as "user").

If you change directories to the root directory with the `cd /` command and ask for a directory listing, you see several directories. These make up the contents of the root file system and provide the mount points for other file systems as well.

The /bin directory contains executable programs, known as *binaries.* (In fact, the directory named /bin is short for *binary.*) These programs are essential system files. Many Linux commands, such as `ls`, are actually programs found in this directory.

The /sbin directory is also used to store system binary files. Most files in this directory are used for system administration purposes.

The /etc directory is very important, containing many of the Linux system configuration files. Essentially, these files give your Linux system its "personality." The password file, passwd, is found here, as is the list of file systems to mount at startup, fstab. Also, this directory contains the startup scripts for Linux, the list of hosts with IP addresses that you want permanently recorded, and many other types of configuration information.

The shared libraries that programs use when they run are stored in the /lib directory. By using shared libraries, many programs can reuse the same code, and these libraries can be stored in a common place, thus reducing the size of your programs at run time.

The /dev directory contains special files known as *device files*, which are used to access all the different types of hardware on your system. For example, the /dev/mouse file is for reading input from the mouse. By organizing access to hardware devices in this way, Linux effectively makes the interface to a hardware device look like any other piece of software. This means that you, in many cases, can use the same syntax that you use with software to perform operations on computer hardware devices. For example, to create a tape archive of your home directory on a floppy drive, you can use the following command:

```
tar -cdf /dev/fd0 ~tackett
```

Many of the devices in the /dev directory are in logical groups. Table 14.1 lists some of the most commonly used devices in the /dev directory.

Table 14.1 Some of the Most Commonly Used Devices in the /dev Directory

Device File	Description
/dev/console	The *system console*, which is the computer monitor physically connected to your Linux system.
/dev/hd	The device driver interface to IDE hard drives. The /dev/hda1 device refers to the first partition on hard drive hda. The device /dev/hda refers to the entire hard disk hda.
/dev/sd	The device driver interface for SCSI disks. The same conventions for SCSI disks and partitions apply as they do to the IDE /dev/hd devices.
/dev/fd	Device drivers that provide support for floppy drives. /dev/fd0 is the first floppy drive and /dev/fd1 is the second floppy drive.
/dev/st	The device driver for SCSI tape drives.
/dev/tty	Device drivers that provide different consoles for user input. The name comes from when terminals known as *teletypes* were physically hooked to a UNIX system. Under Linux, these files provide support for the virtual consoles that can be accessed by pressing <Alt-F1> through <Alt-F6>. These virtual consoles provide separate simultaneous local login sessions.
/dev/pty	Device drivers that provide support for pseudo-terminals, which are used for remote login sessions such as login sessions using Telnet.
/dev/ttyS	The serial interface ports on your computer. /dev/ttyS0 corresponds to COM1 under MS-DOS. If you have a serial mouse, /dev/mouse is a symbolic link to the appropriate ttyS device that your mouse is connected to.
/dev/cua	Special call-out devices used with modems.

continues

Table 14.1 Continued

Device File	Description
/dev/null	A very special device—essentially a black hole. All data written to /dev/null is lost forever. This can be very useful if you want to run a command and throw away the standard output or the standard error. Also, if /dev/null is used as an input file, a file of zero length is created.

The /proc directory is actually a virtual file system. It's used to read process information from memory.

The /tmp directory is used to store temporary files that programs create when running. If you have a program that creates a lot of large temporary files, you may want to mount the /tmp directory as a separate file system rather than just have it as a directory on the root file system. If /tmp is left as a directory on the root file system and has lots of large files written to it, the root file system can fill up.

The /home directory is the base directory for user home directories. It's common to mount this as a separate file system so that users can have plenty of room for their files. In fact, if you have many users on your system, you may need to separate /home into several file systems. To do so, you could create subdirectories such as /home/staff and /home/admin for staff members and administrators, respectively. Mount each of these as different file systems and then create the users' home directories under them.

The /var directory holds files that tend to change in size over time. Typically, various system log files are located below this directory. The /var/spool directory and its subdirectories are used to hold data that's of a transitory nature, such as mail and news that's recently received from or queued for transmission to another site.

TIP You can create other mount points under the / directory if you want. You might want to create a mount point named /cdrom if you routinely mount CD-ROMs on your system.

The /usr directory and its subdirectories are very important to the operation of your Linux system. It contains several directories with some of the most important programs on your system. Typically, subdirectories of /usr contain the large software packages that you install. Table 14.2 discusses some of the /usr subdirectories. The /usr directory is almost always mounted as a separate file system.

Table 14.2 Important Subdirectories in the /usr File System

Subdirectory	Description
/usr/bin	This directory holds many of the executable programs found on your Linux system.
/usr/etc	This directory contains many miscellaneous system configuration files.

Subdirectory	Description
/usr/include	Here and in the subdirectories of /usr/include is where you find all the include files for the C compiler. These header files define constants and functions and are critical for C programming.
/usr/g++-include	This directory contains the include files for the C++ compiler.
/usr/lib	This directory contains various libraries for programs to use during linking.
/usr/man	This directory contains the various manual pages for programs on your Linux system. Below /usr/man are several directories that correspond to the different sections of the man pages.
/usr/src	This directory contains directories that hold the source code for different programs on your system. If you get a package that you want to install, /usr/src/***packagename*** is a good place to put the source before you install it.
/usr/local	This directory is designed for local customizations to your system. In general, much of your local software is installed in this directory's subdirectories. The format of this directory varies on almost every UNIX system you look at. One way to set it up is to have a /usr/local/bin for binaries, a /usr/local/etc for configuration files, a /usr/local/lib for libraries, and a /usr/local/src for source code. The entire /usr/local directory tree can be mounted as a separate file system if you need a lot of room for it.

Mounting and Unmounting File Systems

By now, you should have a good feel for what a file system is. So how do you set up a directory as a separate file system?

To mount a file system in the Linux directory tree, you must have a physical disk partition, CD-ROM, or floppy disk that you want to mount. You also must make sure that the directory to which you want to attach the file system, known as the ***mount point***, actually exists.

Mounting a file system doesn't create the mount point directory. The mount point must exist before you try to mount the file system. Suppose that you want to mount the CD-ROM in drive /dev/sr0 under the mount point /mnt. A directory named /mnt must exist, or the mount fails. After you mount the file system under that directory, all the files and subdirectories on the file system appear under the /mnt directory. Otherwise, the /mnt directory is empty.

TIP

Use `df .` if you need to know on which file system the current directory is located. The command's output shows the file system as well as the free space available.

Part III Ch 14

Mounting File Systems Interactively

As you may have guessed by now, Linux uses the `mount` command to mount a file system. The syntax of the `mount` command is

```
mount device mountpoint
```

`device` is the physical device that you want to mount; *`mountpoint`* is the point in the file system tree where you want it to appear.

NOTE The `mount` command can be used only by superusers. This is to help ensure system security. Several software packages are available that allow users to mount specific file systems, especially floppy disks.

`mount` accepts several command-line arguments in addition to the two mentioned above (see Table 14.3). If a needed command isn't given, `mount` attempts to figure it out from the /etc/fstab file.

Table 14.3 Command-Line Arguments for the *mount* Command

Argument	Description
`-f`	Causes everything to be done except for the actual mount system call. This "fakes" mounting the file system.
`-v`	Verbose mode; provides additional information about what `mount` is trying to do.
`-w`	Mounts the file system with read and write permissions.
`-r`	Mounts the file system with read-only permission.
`-n`	Mounts without writing an entry in the /etc/mtab file.
`-t` *`type`*	Specifies the type of the file system being mounted. Valid types are `minux`, `ext`, `ext2`, `xiafs`, `msdos`, `hpfs`, `proc`, `nfs`, `umsdos`, `sysv`, and `iso9660` (the default).
`-a`	Causes `mount` to try to mount all file systems in /etc/fstab.
`-o` *`list_of_options`*	When followed by a comma-separated list of options, causes `mount` to apply the options specified to the file system being mounted. Many options are available here; for a complete list, refer to the `mount` man page.

NOTE Several forms of the `mount` command are very common. `mount /dev/hdb3 /mnt` mounts the hard-disk partition /dev/hdb3 under the directory /mnt. `mount -r -t iso9660/dev/sr0 /mnt` mounts the SCSI CD-ROM drive /dev/sr0, which is read-only and of the ISO 9660 file format under the directory /mnt. `mount -vat nfs` mounts all the NFS file systems listed in the /etc/fstab file.

TIP If a file system doesn't mount correctly, use the command `mount -vf` *`device mountpoint`* to see what `mount` is doing. This gives a verbose listing and tells `mount` to do everything except mount the file system. This way, you can fake out the `mount` command and get a lot of information about what it's trying to do.

Mounting File Systems at Boot Time

Under most circumstances, the file systems that your Linux system uses won't change frequently. For this reason, you can easily specify a list of file systems that Linux mounts when it boots and that it unmounts when it shuts down. These file systems are listed in a special configuration file named /etc/fstab, for *file system table.*

The /etc/fstab file lists the file systems to be mounted, one file system per line. The fields in each line are separated by spaces or tabs. Table 14.4 lists the different fields in the /etc/fstab file.

Table 14.4 Fields in the /etc/fstab File

Field	Description
File system specifier	Specifies the block special device or the remote file system to be mounted.
Mount point	Specifies the mount point for the file system. For special file systems such as swap files, use the word `none`, which makes swap files active but not visible within the file tree.
Type	Gives the file system type of the specified file system. Now, the following types of file systems are supported: • `minix`, a local file system supporting file names of 14 or 30 characters • `ext`, a local file system with longer file names and larger inodes (this file system has been replaced by the `ext2` file system and should no longer be used) • `ext2`, a local file system with longer file names, larger inodes, and other features • `xiafs`, a local file system • `msdos`, a local file system for MS-DOS partitions • `hpfs`, a local file system for OS/2 High Performance File System partitions • `iso9660`, a local file system used for CD-ROM drives • `nfs`, a file system for mounting partitions from remote systems

continues

Table 14.4 Continued

Field	Description
	• swap, a disk partition or special file used for swapping • umsdos, a UMSDOS file system • sysv, a System V file system
Mount Options	A comma-separated list of mount options for the file system. At a minimum, it must contain the type of mount for the file system. See the `mount` man page for more information on mount options.
Dump Frequency	Specifies how often the file system should be backed up by the `dump` command. If this field isn't present, `dump` assumes that the file system doesn't need to be backed up.
Pass Number	Specifies in what order the file systems should be checked by the `fsck` command when the system is booted. The root file system should have a value of 1. All other file systems should have a value of 2. If a value isn't specified, the file system won't be checked for consistency at boot time.

TIP It's recommended that you mount your file systems at boot time via the /etc/fstab file instead of by using the `mount` command. Remember, only superusers can use `mount`.

The following is a sample fstab file:

```
# device        directory      type     options
/dev/hda1       /              ext2     defaults
/dev/hda2       /usr           ext2     defaults
/dev/hda3       none           swap     sw
/dev/sda1       /dosc          msdos    defaults
/proc           /proc          proc     none
```

In this sample file, you can see several different file systems. First, notice that comments in the file are prefixed by a # character. In this fstab file, two normal Linux file systems are mounted—the disk partitions /dev/hda1 and /dev/hda2. These are listed as being of type `ext2` and are mounted under the root directory, / and /usr respectively.

The entry defaults listed under the `options` field indicates that this file system should be mounted by using a common set of default options. Specifically, the file system is mounted read/write enabled, it's to be interpreted as a block special device, all file I/O should be done asynchronously, the execution of binaries is permitted, the file system can be mounted with the `mount -a` command, the set UID (user ID) and set GID (group ID) bits on files are interpreted on this file system, and ordinary users aren't allowed to mount this file system. As you can see, it's a lot easier just to type **`defaults`** for the option instead.

▶ **See** "Creating the Swap Partition," **pp. 60 and 97**

The partition /dev/hda3 is a swap partition that's used for kernel virtual-memory swap space. Its mount point is specified as `none` because you don't want it to appear in the file system tree. It still has to be in the /etc/fstab file, so the system knows where it's physically located. Swap partitions are also mounted with the option `sw`.

The /proc file system is a virtual file system that points to the process information space in memory. As you can see, it doesn't have a corresponding physical partition to mount.

TIP For full information on all options available in the /etc/fstab file, refer to the man page for fstab.

MS-DOS file systems can also be mounted automatically. The partition /dev/sda1 is the first partition on the SCSI hard drive sda. It's mounted as an MS-DOS partition by specifying `msdos` as the type and by giving /dosc as its mount point. You can place the mount point for the MS-DOS file system anywhere—there's no requirement that it be under the root directory.

Unmounting File Systems

Now that you know all sorts of stuff about mounting file systems, it's time to look at how to unmount. You use the `umount` command to unmount file systems. You would want to unmount a file system for several reasons: so that you can check/repair a file system with `fsck`, unmount NFS-mounted file systems in case of network problems, or unmount a file system on a floppy drive.

NOTE This command is `umount`, not "unmount." Make sure that you type it correctly.

There are three basic forms of the `umount` command:

```
umount device ¦ mountpoint
```

```
umount -a
```

```
umount -t fstype
```

`device` is the name of the physical device to unmount; *`mountpoint`* is the mount point directory name (specify only one or the other). The `umount` command has only two command-line parameters: `-a` unmounts all file systems, and `-t` *`fstype`* acts only on file systems of the type specified.

Part III Ch 14

CAUTION

The `umount` command doesn't unmount a file system that's in use. For example, if you have some file system mounted under /mnt and you try

```
cd /mnt
```

continues

continued

umount /mnt

you get an error telling you that the file system is busy. You have to change to a different directory in another file system to unmount the file system mounted under /mnt.

Understanding the Network File System

The Network File System (NFS) is a system that allows you to mount file systems from a different computer over a TCP/IP network. Under NFS, a file system on a remote computer is mounted locally and looks just like a local file system to users. The illusion of being mounted locally has numerous uses. For example, you can have one machine on your network with a lot of disk space acting as a file server. This computer has all the home directories of all your users on its local disks. By mounting these disks via NFS on all your other computers, your users can access their home directories from any computer.

NFS has three essential components:

- The computers with the file systems that you want to NFS mount must be able to communicate with each other via a TCP/IP network.
- The computer with the file system that you're interested in as a local file system must make that file system available to be mounted. This computer is known as the *server*, and the process of making the file system available is known as *exporting the file system*.
- The computer that wants to mount the exported file system, known as the *client*, must mount the file system as an NFS file system via the /etc/fstab file at boot time or interactively via the `mount` command.

The following sections discuss exporting the file system and mounting it locally.

Exporting an NFS File System

For clients to mount an NFS file system, this file system must be made available by the server. Before the file system can be made available, you must ensure that it's mounted on the server. If the file system is always going to an NFS exported file system, you should make sure that you have it listed in the /etc/fstab file on the server so that it automatically mounts when the server boots.

When you have the file system mounted locally, you can make it available via NFS. This is a two-step process. First, you must make sure that the NFS daemons `rpc.mountd` and `rpc.nfsd` are running on your server. These daemons are usually started from the startup /etc/rc.d/init.d/nfs script. Usually, all that's needed is to make sure that the following lines are in your script:

```
daemon rpc.mountd
daemon rpc.nfsd
```

NOTE As RPC-based programs, the `rpc.mountd` and `rpc.nfsd` daemons aren't managed by the `inetd` daemon but are started up at boot time, registering themselves with the `portmap` daemon. You must be sure to start them only after `rpc.portmap` is running.

Second, you must enter the NFS file system in a configuration file named /etc/exports. This file contains information about what file systems can be exported, what computers are allowed to access them, and what type and level of access is permitted.

Understanding the /etc/exports File

The /etc/exports file is used by the `mountd` and `nfsd` daemons to determine what file systems are to be exported and what restrictions are placed on them. File systems are listed in /etc/exports, one per line. The format of each line is the name of the mount point for a local file system, followed by a list of computers that are allowed to mount this file system. A comma-separated list of mount options in parentheses may follow each name in the list. Table 14.5 lists the mount options available in the /etc/exports file.

Table 14.5 Mount Options Available in the /etc/exports File

Option	Description
`insecure`	Permits non-authenticated access from this machine.
`secure`	Requires secure RPC authentication from this machine.
`root_squash`	Maps any requests from root, UID 0 on the client, to the UID NOBODY_UID on the server.
`no_root_squash`	Doesn't map any requests from UID 0 (default behavior).
`ro`	Mounts the file system as read-only (default behavior).
`rw`	Mounts the file system as read-write.
`link_relative`	Converts absolute symbolic links (where the link contents start with a slash) into relative links by prefixing the link with the necessary number of `../` characters to get from the directory containing the link to the root on the server.
`link_absolute`	Leaves all symbolic links as they are (normal behavior for Sun NFS servers). This is the default behavior for Linux.
`map_daemon`	Maps local and remote names and numeric IDs by using an `lname/uid map` daemon on the client where the NFS request originated. Used to map between the client and server UID spaces.
`all-squash`	Maps all uids and gids to the anonymous user. This option is useful for NFS-exported public directories, such as those housing FTP and news.

continues

Table 14.5 Continued

Option	Description
no-all-squash	The opposite of the all-squash option. This is the default option for Linux.
squash-uids	Specifies a list of uids subject to anonymous mappings. A valid list of ids looks like this: squash_uids=0-15,20,25-50
squash-gids	Specifies a list of gids subject to anonymous mappings. A valid list of ids looks like this: squash_gids=0-15,20,25-50
anonuid	Sets the uid for the anonymous account. This option is useful for PC/NFS clients.
anongid	Sets the gid for the anonymous account. This option is useful for PC/NFS clients.
noaccess	Used to exclude certain subdirectories from a client. Makes everything below the directory inaccessible to the client.

Here is a sample /etc/exports file:

```
/home               bill.tristar.com(rw) fred.tristar.com(rw)
➥george.tristar.com(rw)
/usr/local/bin          *.tristar.com(ro)
/projects           develop.tristar.com(rw) bill.tristar.com(ro)
/pub                (ro,insecure,root_quash)
```

In this example, the server exports four different file systems. /home is mounted with read/write access on three different computers: bill, fred, and george. This indicates the directory probably holds user home directories because of the directories' names. The /usr/local/bin file system is exported as read-only with access allowed for every computer in the tristar.com domain.

The /projects file system is exported with read/write access for the computer develop.tristar.com but with read-only access for bill.tristar.com.

For the /pub file system, there's no list of hosts that are allowed access. This means that any host is allowed to mount this file system. It has been exported as read-only, non-authenticated access allowed, and the server remaps any request from root on a remote machine that accesses this file system.

Mounting NFS File Systems

Mounting an NFS file system is similar to mounting any other type of file system. You can mount NFS file systems from the /etc/fstab file at boot time or interactively via the mount command.

CAUTION

You must be sure to separate the host name and file/system/path portions of the remote file system name with a colon, such as

```
mailserver:/var/spool/mail
```

when using the `mount` command or when making an entry in /etc/fstab. If you don't separate the host name from the directory, your system won't mount the remote directory correctly.

Mounting NFS File Systems via /etc/fstab When you specify an NFS file system in the /etc/fstab file, you identify the file system with the format

```
hostname:/file/system/path
```

hostname is the name of the server where the file system is located; */file/system/path* is the file system on the server.

The file-system type is specified as `nfs` in the mount options field of the file system entry. Table 14.6 lists the most commonly used `mount` options.

Table 14.6 Commonly Used Options for NFS Mounts

Option	Description
`rsize=n`	Specifies the datagram size in bytes used by the NFS clients on read requests. The default value is 1,024 bytes.
`wsize=n`	Specifies the datagram size in bytes used by the NFS clients on write requests. The default value is 1,024 bytes.
`timeo=n`	Sets the time, in tenths of a second, that the NFS client waits for a request to complete. The default value is 0.7 seconds.
`hard`	Mounts this file system by using a hard mount. This is the default behavior.
`soft`	Mounts this file system by using a soft mount.
`intr`	Allows signals to interrupt an NFS call. This is useful for aborting an operation when an NFS server doesn't respond.

Hard Mounts vs. Soft Mounts

Hard mounts and *soft mounts* determine how an NFS client behaves when an NFS server stops responding. NFS file systems are hard-mounted by default. With either type of mount, if a server stops responding, the client waits until the timeout value specified by the `timeo` option expires and then resends the request (this is known as a *minor timeout*). If the requests to the server continue to time out and the total timeout reaches 60 seconds, a *major timeout* occurs.

If a file system is hard mounted, the client prints a message to the console and starts the mount requests all over again by using a timeout value that's twice that of the previous cycle. This has the

continues

continued

potential to go on forever. The client keeps trying to remount the NFS file system from the server until it gets it. Period.

Soft mounts, on the other hand, just generate an I/O error to the calling process when a major timeout occurs. Linux then continues on its merry way.

Typically, important software packages and utilities that are mounted via NFS should be mounted with hard mounts. This is why hard mounts are the default. You don't want your system to start acting strange if the Ethernet gets unplugged for a moment; you want Linux to wait and continue when the network is back up. On the other hand, you might want to mount non-critical data, such as remote news spool partitions, as soft mounts so that if the remote host goes down, it won't hang your current login session.

A typical NFS file system entry in the /etc/fstab file might look like this:

```
mailserver:/var/spool/mail  /var/spool/mail   nfs timeo=20,intr
```

This entry mounts the /var/spool/mail file system located on the host mailserver at the local mount point /var/spool/mail. It specifies that the file system type is `nfs`. Also, it sets the timeout value to 2 seconds (20 tenths of a second) and makes operations on this file system interruptible.

Mounting NFS File Systems Interactively NFS file systems can be mounted interactively, just like any other type of file system. However, you should be aware that the NFS `mount` command isn't very pretty due to all the options that you can specify on the command line.

By using the previous example, the interactive `mount` command that you use to mount the /var/spool/mail file system becomes

```
# mount -t nfs -o timeo=20,intr mailserver:/var/spool/mail  /var/spool/mail
```

If you need to specify datagram sizes and timeouts, interactive `mount` commands can become very complex. It's highly recommended that you place these `mount` commands in your /etc/fstab file so that they can be mounted automatically at boot time.

Maintaining File Systems

As the systems administrator, you're responsible for maintaining the integrity of the file systems themselves. Typically, this means checking the file systems periodically for damaged or corrupted files. Linux automatically checks file systems at boot time if they have a value greater than 0 specified in the pass number field of the /etc/fstab file.

NOTE The ext2 file system commonly used under Linux has a special flag known as a *clean bit*. If the file system has been synchronized and unmounted cleanly, the clean bit is set on the file system. If the clean bit is set on a file system when Linux boots, it's not checked for integrity. ■

Using the *fsck* Command

It's a good idea to check your file systems occasionally for damaged or corrupt files. Under the Slackware distribution of Linux, you use the `fsck` (file system check) command to check your file systems. The `fsck` command is really a "front end" for a series of commands that are designed to check specific file systems. The syntax for the `fsck` command is

```
fsck [-A] [-V] [-t fs-type] [-a] [-l] [-r] [-s] filesys
```

However, the most basic form of the command is

```
fsck filesys
```

Table 14.7 describes the command-line options for the `fsck` command.

Table 14.7 Command-Line Arguments for *fsck*

Argument	Description
`-A`	Goes through the /etc/fstab file and tries to check all file systems in one pass. This option is typically used during the Linux boot sequence to check all normally mounted file systems. If you use `-A`, you can't use the *`filesys`* argument as well.
`-V`	Verbose mode. Prints additional information about what `fsck` is doing.
`-t` *`fs-type`*	Specifies the type of file system to be checked.
`filesys`	Specifies which file system is to be checked. This argument can be a block special device name, such as /dev/hda1, or a mount point, such as /usr.
`-a`	Automatically repairs any problems found in the file system without asking any questions. Use this option with caution.
`-l`	Lists all the file names in the file system.
`-r`	Asks for confirmations before repairing the file system.
`-s`	Lists the superblock before checking the file system.

The `fsck` command is actually a front-end program that calls the command to check the file system that matches the type you specify. To do so, Linux needs to know the file system type that it's checking. The easiest way to make sure that `fsck` calls the right command is to specify a file system type with the `-t` option to `fsck`. If you don't use the `-t` option, Linux tries to figure out the file system type by looking up the file system in /etc/fstab and by using the file type specified there. If `fsck` can't find the file type information in /etc/fstab, it assumes that you're using a Minix file system.

CAUTION

The fsck command assumes that the file system you're checking is a Minix file system if you don't tell it differently—either with the -t argument or by listing the type in /etc/fstab. Because your Linux file systems are probably of type ext2 and not Minix, you should be careful and make sure that fsck knows the correct type. This is especially important if you're checking a file system that isn't listed in the /etc/fstab file.

It's a good idea to unmount a file system before checking it. This ensures that none of the files on the file system are in use when they're being checked.

NOTE Remember, you can't unmount a file system if any of the files on it are busy. For example, if a user is now in a directory on a file system that you try to unmount, you get a message saying that the file system is busy. ■

Trying to check the root file system presents an additional problem. You can't directly unmount the root file system, because Linux must be able to access it in order to run. To check the root file system, you should boot from a maintenance floppy disk that has a root file system on it, and then run `fsck` on your real root file system from the floppy by specifying the special device name of your root file system. If `fsck` makes any changes to your file system, it's important that you reboot your system immediately. This allows Linux to reread important information about your file system and prevents your file system from further corruption.

CAUTION

Be sure to reboot your computer immediately after you run `fsck` if any changes were made to your file system, to prevent further corruption to your file system. Use the `shutdown -r` command or the `reboot` command to reboot.

Creating and Formatting File Systems

When you add a new hard disk to your computer or want to change the partition information on an old hard disk, you go through the steps of creating a file system from a raw disk. Assuming that you've added a new hard disk to your system, you must set the disk partition information and then create the actual file systems on the disk before Linux can use the disk. To change disk partition information, you use the `fdisk` command. After you partition the hard drive, you need to create the file systems by using the `mkfs` command.

Using *fdisk* to Create Disk Partitions

The `fdisk` command is used to create disk partitions and set the attributes that tell Linux what type of file system is on a particular partition. If you installed Linux from scratch on an

MS-DOS system, you had to run fdisk to change the disk partition information before you could install Linux.

CAUTION

Using fdisk on a disk can destroy all data on the disk. Because fdisk completely rewrites the file table on the disk, all your former files may be lost. Make sure that you have a complete, current backup of your disks before using fdisk.

You should always run the fdisk command on an unmounted file system. fdisk is an interactive, menu-driven program, not just a single command. To start fdisk, type

```
fdisk [drive]
```

drive is the physical disk drive that you want to work on. If you don't specify a disk, the disk /dev/hda is assumed. For example, to run fdisk on the second IDE hard drive in your system, enter

```
fdisk /dev/hdb
```

at the superuser command prompt. Because fdisk is a menu-driven program, several different commands are available when you're using fdisk, as summarized in Table 14.8.

Table 14.8 Commands Available from the *fdisk* Menu

Command	Description
a	Toggles the bootable flag on a partition
c	Toggles the DOS compatibility flag on a partition
d	Deletes a partition
l	Lists partition types known to fdisk
m	Displays a menu listing all available commands
n	Adds a new partition
p	Prints the partition table for the current disk
q	Quits without saving any changes
t	Changes the file system type for a partition
u	Changes display/entry units
v	Verifies the partition table
w	Writes the table to disk and exits

continues

Table 14.8 Continued

Command	Description
x	Lists additional functions for experts: ● b Moves the beginning location of data in a partition ● c Changes the number of cylinders ● d Prints the raw data in the partition table ● e Lists extended partitions on disk ● h Changes number of heads on disk ● r Returns to main menu ● s Changes number of sectors on disk

`fdisk` can set the file system type of a disk partition to any of several different types. Only use Linux `fdisk` to create partitions used under Linux. For MS-DOS or OS/2 partitions, you should use the `fdisk` tool that's native to that operating environment, and then use Linux's `fdisk` to tag the partitions as Linux native or Linux swap.

Table 14.9 lists the partitions supported by Linux `fdisk`. Each partition type has an associated hexadecimal code that identifies it. You must enter the appropriate code in `fdisk` when you want to set a partition type.

Table 14.9 Partition Codes and Types in Linux *fdisk*

Hex Code	Partition Type
0	Empty
1	DOS 12-bit FAT
2	XENIX root
3	XENIX usr
4	DOS 16-bit file system, less than 32M
5	Extended
6	DOS 16-bit file system supporting more than 32M
7	OS/2 High Performance File System (HPFS)
8	AIX
9	AIX bootable
a	OS/2 Boot Manager
40	Venix 80286

Hex Code	Partition Type
51	Novell?
52	Microport
63	GNU HURD
64	Novell NetWare
65	Novell NetWare
75	PC/IX
80	Old MINIX
81	Linux/MINIX
82	Linux swap, used for swap files under Linux
83	Linux native, common Linux file system type
93	Amoeba
94	Amoeba BBT
a5	BSD/386
b7	BSDI file system
b8	BSDI swap file system
c7	Syrinx
db	CP/M
e1	DOS access
e3	DOS R/O
f2	DOS secondary
ff	BBT

The following sections show how to use `fdisk`. Here is an example of how to use `fdisk` to set up the partitions on a hard disk for use by Linux. Assume that you want to configure the first IDE drive in your system for Linux. Make sure that you have a backup of your data. All data on your hard disk is destroyed in the process. The name of the first IDE hard disk is /dev/hda, which is the default device for Linux.

Part III Ch 14

Running *fdisk* You run `fdisk` with

```
# fdisk
```

`fdisk` responds with

```
Using /dev/hda as default device!
Command (m for help):
```

This tells you that `fdisk` is using disk /dev/hda as the device that you're working with. Because this is what you wanted, you're fine. You should always check to make sure that you're really on the disk that you think that you're on. Linux then displays the `fdisk` command prompt.

Displaying the Current Partition Table The first thing you want to do is display the current partition table. This is done with the `p` command:

```
Command (m for help): p
Disk /dev/hda: 14 heads, 17 sectors, 1024 cylinders
Units = cylinders of 238 * 512 bytes

Device      Boot     Begin     Start     End     Blocks   Id      System

Command (m for help):
```

This listing shows that the current disk, /dev/hda, has a geometry of 14 heads, 17 sectors, and 1,024 cylinders. The display units are in cylinders of 238 * 512 (121,856) bytes each. Because there are 1,024 cylinders and each cylinder is 121,856 bytes, you can deduce that the disk can hold 1,024×121,856 = 124,780,544 bytes, or about 120M. You can also see that /dev/hda has no partitions.

Creating a New Partition Assume that you want to create a 100M Linux file partition for user home directories and a 20M swap partition. Your next step is to use the `n` command to create a new partition:

```
Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1023):  1
Last cylinder or +size or +sizeM or +sizeK (1-1023):  +100M
```

Using the `n` command to create a new partition displays another menu. You must choose whether you want to create an extended partition or a primary partition. You typically want to create a primary partition unless you have more than four partitions on a disk. `fdisk` then asks you for the partition number that you want to create. Because this is the first partition on the disk, you answer **`1`**. You're then prompted for the first cylinder for the partition. This determines where on the disk the data area starts. Again, because this will be the first partition on the disk, you can start the partition at cylinder 1.

The next line asks you how large you want the partition to be. You have several options as to how to answer this question. `fdisk` accepts either a number, which it interprets as the size in cylinders, or the size in bytes, kilobytes, or megabytes. The size in bytes is specified as *`+bytes`*, where *`bytes`* is the size of the partition. Similarly, `+`*`size`*`K` and `+`*`size`*`M` set the partition size to size kilobytes or size megabytes, respectively. You know that you want a 100M partition, so the easiest answer to the prompt is **`+100M`**.

Rechecking the Partition Table Now you should check the partition table again to see what `fdisk` has done:

```
Command (m for help): p
Disk /dev/hda: 14 heads, 17 sectors, 1024 cylinders
Units = cylinders of 238 * 512 bytes

Device      Boot     Begin    Start     End     Blocks      Id     System
/dev/hda1                1        1     861     102400      81     Linux/MINIX
Command (m for help):
```

The partition table shows that you have 1 partition, /dev/hda1, that goes from cylinder 1 to cylinder 861 and uses 102,400 blocks. It's listed as being type 81, Linux/MINIX.

Creating the Swap Partition Now you need to create the 20M swap partition by using the remaining disk space. This is just like creating the first partition:

```
Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (862-1023):  862
Last cylinder or +size or +sizeM or +sizeK (862-1023): 1023
```

TIP It's usually better to go ahead and enter the size of the last partition in cylinders to make sure that you use all the disk space.

Here you specified partition number 2 for the second partition. When `fdisk` prompts for the first cylinder, notice that it gives a range of 862 to 1023. This is because the first partition takes up everything before cylinder 862. So enter **862** as the starting cylinder for the second partition. You want to use all the remaining space on the disk for the swap partition. You should have about 20M left, but if you specify the size in megabytes, the internal `fdisk` calculations could leave you with a couple of unused cylinders. So you enter **1023** for the last cylinder on the size prompt.

NOTE You might see an error similar to

```
        Warning: Linux cannot currently use the last xxx sectors of this
partition.
```

where *xxx* is some number. Such an error can be ignored. It's left over from the days when Linux couldn't access file systems larger than 64M. □

Making Sure the Sizes Are Correct At this point, you've created both partitions that you wanted to create. You should take a look at the partition table one more time to check that the sizes are correct:

```
Command (m for help): p
Disk /dev/hda: 14 heads, 17 sectors, 1024 cylinders
Units = cylinders of 238 * 512 bytes

Device        Boot   Begin     Start     End     Blocks      Id     System
/dev/hda1                1         1     861     102400      81     Linux/MINIX
```

```
/dev/hda2                862    862     1023     19159      81      Linux/MINIX
Command (m for help):
```

As you can see, /dev/hda1 uses cylinder 1 through cylinder 861 with a size of 102,400 blocks, which is approximately 100M. Partition /dev/hda2 goes from cylinder 862 to cylinder 1023 with a size of 19,156 blocks, or almost 20M.

Changing the Partition Type The next thing that you need to do is change the partition type for each partition. To change the partition type, use the `t` command at the `fdisk` command prompt. The most common choice for a standard Linux file system partition is to set it to partition type 83, Linux native. Swap partitions should be set to partition type 82, Linux swap.

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 83
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
```

When you use the `t` command, you're prompted for the partition number that you want to change. You're then prompted for the hex code for the file system ID that you want to set the partition to. Typically, Linux file systems are set to type 83 for normal file systems, and type 82 for swap partitions. You can type `l` at this point to see a list of file systems, if you want.

Finishing Up Now that you've created the partitions and labeled them, you should take one last look at the partition table before you exit just to make sure that everything is okay.

```
Command (m for help): p
Disk /dev/hda: 14 heads, 17 sectors, 1024 cylinders
Units = cylinders of 238 * 512 bytes

Device        Boot    Begin    Start    End     Blocks     Id     System
/dev/hda1                1       1      861     102400     83     Linux native
/dev/hda2              862     862     1023     19159      82     Linux swap
Command (m for help):
```

As you can see, the partitions are in the right place, they're the right size, and the file system types are set correctly. The last thing that you need to do is use the `w` command to write the partition table to disk and exit:

```
Command (m for help):  w

#
```

None of the changes that you make during an `fdisk` session take effect until you write them to disk with the `w` command. You can always quit with the `q` command and not save any changes. This said, you should still always have a backup of any disk that you want to modify with `fdisk`.

After you make changes to a disk with `fdisk`, you should reboot the system just to make sure that Linux has the updated partition information in the kernel.

Using *mkfs* to Build a File System

After you create a file system partition with `fdisk`, you must build a file system on it before you can use it for storing data. This is done with the `mkfs` command. Think of building a parking lot. If you think of `fdisk` as physically building the parking lot, `mkfs` is the part of the process that paints the lines so that the drivers know where to park.

Just like `fsck` is a "front-end" program for checking different types of file systems, `mkfs` actually calls different programs to create the file system, depending on what file system type you want to create. The syntax of the `mkfs` command is

```
mkfs  [-V] [-t fs-type] [fs-options] filesys [blocks]
```

where *filesys* is the device of the file system that you want to build, such as /dev/hda1.

CAUTION

The `mkfs` command also accepts the name of a mount point, such as /home, as the file system name. You should be extremely careful about using a mount point. If you run `mkfs` on a mounted "live" file system, you might very well corrupt all the data on that file system.

Table 14.10 lists the various command-line parameters that you can specify with `mkfs`.

Table 14.10 Command-Line Parameters for the *mkfs* Command

Option	Description
`-V`	Causes `mkfs` to produce verbose output, including all file system-specific commands that are executed. Specifying this option more than once inhibits execution of any file system-specific commands.
`-t` *fs-type*	Specifies the type of file system to be built. If the file system type isn't specified, `mkfs` tries to figure it out by searching for *filesys* in /etc/fstab and using the corresponding entry. If the type can't be deduced, a MINIX file system is created.
fs-options	Specifies file system-specific options that are to be passed to the actual file system-builder program. Although not guaranteed, the following options are supported by most file system builders: ● `-c` Checks the device for bad blocks before building the file system ● `-l` *file-name* Reads a list of the bad blocks on the disk from *file-name* ● `-v` Tells the actual file system builder program to produce verbose output

continues

Table 14.10 Continued

Option	Description
filesys	Specifies the device on which the file system resides. This parameter is required.
blocks	Specifies the number of blocks to be used for the file system.

Although `-t` *`fs-type`* is an optional argument, you should get in the habit of specifying the file system type. Just like `fsck`, `mkfs` tries to figure out the type of the file system from the /etc/fstab file. If it can't figure it out, it creates a MINIX file system by default. For a normal Linux file system, you probably want an `ext2` partition instead.

Using Swap Files and Partitions

Swap space on your Linux system is used for virtual memory. A complete discussion of all the issues involved with virtual memory is beyond the scope of this book. Any good general computer operating system text book discusses the issue in detail.

Linux supports two types of swap space: swap partition and swap files. A *swap partition* is a physical disk partition with its file system ID set to type 82, Linux swap, and is dedicated for use as a swap area. A *swap file* is a large file on a normal file system that's used for swap space.

You're better off using a swap partition instead of a swap file. All access to a swap file is performed through the normal Linux file system. The disk blocks that make up the swap file are probably not contiguous and, therefore, performance isn't as good as it is with a swap partition. I/O to swap partitions is performed directly to the device, and disk blocks on a swap partition are always contiguous. Also, by keeping the swap space off a normal file system, you reduce the risk of corrupting your regular file system if something bizarre happens to your swap file.

Creating a Swap Partition

To create a swap partition, you must have created a disk partition by using `fdisk` and tagged it as type 82, Linux swap. After you create the swap partition, you have two additional steps to follow to make the swap partition active.

First, you must prepare the partition in a manner similar to creating a file system. Instead of `mkfs`, the command for preparing the partition is `mkswap`. The syntax of the `mkswap` command is

```
mkswap [-c] device size-in-blocks
```

`device` is the name of the swap partition, such as /dev/hda2, and *`size-in-blocks`* is the size of the target file system in blocks. You can get the size in blocks by running `fdisk` and looking at the partition table. In the example earlier in the section "Making Sure the Sizes Are Correct," the size of /dev/hda2 was 19,159 blocks. Linux requires that swap partitions be between 9 and 65,537 blocks in size. The `-c` argument tells `mkswap` to check the file system for bad blocks when creating the swap space, which is a good idea.

Following the example in "Making Sure the Sizes Are Correct," the command for setting up a swap partition on /dev/hda2 is

```
mkswap -c /dev/hda2 19159
```

After you run `mkswap` to prepare the partition, you must make it active so that the Linux kernel can use it. The command to make the swap partition active is `swapon`. The syntax for the `swapon` command is

```
swapon filesys
```

`filesys` is the file system that you want to make available as swap space. Linux makes a call to `swapon -a` during boot, which mounts all available swap partitions listed in the /etc/fstab file.

NOTE Remember to put an entry for any swap partitions or swap files that you create into the /etc/fstab file so that Linux can automatically access them at boot time. □

Creating a Swap File

Swap files can be useful if you need to expand your swap space and can't allocate disk space to create a dedicated swap partition. Setting up a swap file is almost identical to creating a swap partition. The main difference is that you have to create the file before you can run `mkswap` and `swapon`.

To create a swap file, you use the `dd` command, which is used for copying large chunks of data. For a full description of this command, see the man page for `dd`. The main things that you have to know before creating the file are the name of the swap file you want to create and its size in blocks. A block under Linux is 1,024 bytes. For example, to create a 10M swap file named /swap, enter

```
# dd if=/dev/zero of=/swap bs=1024 count=10240
```

`of=/swap` specifies that the file to be created is named /swap, and `count=10240` sets the size of the output file to be 10,240 blocks, or 10M. You then use `mkswap` to prepare the file as a swap space:

```
# mkswap /swap 10240
```

Remember that you have to tell `mkswap` how big the file is. Before you run `swapon`, you need to make sure that the file is completely written to disk by using the `/etc/sync` command.

Now you're ready to make the swap file active. Like with the swap partition, you use the `swapon` command to make the file active; for example,

```
# swapon /swap
```

If you need to get rid of a swap file, you must make sure that it's not active. Use the `swapoff` command to deactivate the swap file, as in

```
# swapoff /swap
```

You can then safely delete the swap file.

From Here...

In this chapter, you've looked at many different aspects of the Linux file system, from a tour of the basic directory structure to mounting and unmounting file systems. You've explored accessing remote file systems with NFS and looked in detail at how to create file systems and prepare them for use. Finally, this chapter discussed the creation of swap partitions and swap files.

You can find more information about systems administration in the following chapters:

- Chapter 8, "Understanding Systems Administration," introduces you to common systems administration tasks.
- Chapter 10, "Managing User Accounts," describes how to set up and manage user accounts on your Linux system.
- Chapter 11, "Backing Up Data," discusses how to plan and implement plans for data backups.

Index

Special Edition

USING LINUX

Third Edition

The secrets to unleashing the full power of Linux—at your fingertips!

Que's *Special Edition Using Linux, Third Edition* is your all-in-one resource for installing, configuring, and working with this powerful operating system. From running Linux applications and backing up data to configuring a TCP/IP network , using email, and creating your own HTML page, you'll find all the information you need to master the power of Linux!

Jack Tackett Jr. is a freelance writer and computer consultant based in RTP, NC. He specializes in C/C++ and client/server applications for various high-tech firms. He is also a columnist for Newbridge Communication's Tech Talk newsletter. Currently he is a Linux consultant for Nortel's Information Network group in RTP.

David Gunter is an information technology consultant and computer author based in Cary, NC. In addition to software development, he has been involved with supporting and managing diverse systems and networks for over 10 years.

CD-ROMs include:

Red Hat™, Slackware, and Caldera®'s Open Linux Lite, three complete distributions of the 32-bit Linux operating system. Plus, the HTML version of Que's *Running a Perfect Web Site with Apache.*

◀ See inside Gatefold for more detailed information!

User Level

New Casual Accomplished Expert

Category: Operating System

Covers: Red Hat and Slackware, Caldera, Open Linux

- Learn how to install Red Hat Linux on various DEC Alpha platforms
- Install and configure the XFree86 windowing system, and get X Windows resource files
- Learn the Linux file and directory structure and how to move within it
- Maneuver through Linux and manage users with basic commands
- Learn how to get Apache, the world's most popular Linux Web server, up and running
- Make the most of your Internet connection by accessing Usenet news and email
- Explore the power of shells; understand, optimize and customize your Linux shell
- Receive tips for installing and using the X-based GUI included with Linux

100% Satisfaction Guaranteed

If you are not completely satisfied, you can return this product for a full refund. Call 1-800-858-7674

$59.99 USA / $84.95 CAN / £56.49 Net UK (inc of VAT)

QUE®

Visit us on the Internet at: http://www.quecorp.com

0 29236 11321 6

ISBN 0-7897-1132-X

95999

9 780789 711328