

Chapter 3: Installation Issues

Installation chapters are rarely anyone's favorite. I find myself skipping over installation chapters simply because they mirror the installation instructions that came packaged with the software I'm using. This chapter aims to be something different.

Most Linux installation chapters discuss the steps necessary to install a particular distribution of Linux. These chapters also end when the final "install" button is clicked. In this chapter, however, you'll learn the important steps to take during the installation procedure to ensure that your operating system is secure:

- Differences in installation procedures and security on various Linux distributions
- Partitions and security
- Choosing network services at installation
- Boot loaders

About Various Linux Distributions, Security, and Installation

More than 110 Linux distributions exist, and more will undoubtedly appear and disappear over time. These distributions all share some common characteristics: the same kernel releases, the same basic applications, and, with few exceptions, the same core source code.

This might persuade you that all Linux distributions are identical. Not true. Subtle differences do exist:

- Different Linux distributions have different installation tools, and their functionality might vary. Some installation tools automatically specify which network servers activate on boot, and some don't. Others ask you.
- Some installation tools drill down into individual packages so that you can choose precisely what software is installed. Other installation tools offer less incisive scope, such as asking you which *sets* of software you'd like to install rather than which individual applications.

If you're new to Linux, these variables can affect your system's security. Frankly, you might end up with innumerable software packages and servers installed that you know nothing about.

This is a major problem facing Linux newcomers, and the publishing field hasn't helped. Although there are countless Linux primer books, few of them contain comprehensive lists of installable software. This leaves newbies in an odd position. Faced with choosing individual applications or installing the entire distribution, most will choose the latter.

Note - Older distributions, such as early SlackWare, worked differently. The installation tool, based on shell scripts with a dialog front end, paused at every application and utility, forcing you to choose whether or not to install it. Each dialog displayed the application's description per its Linux Software Map entry. This allowed you to ascertain each program's

purpose and whether or not you needed it. For system administrators who have an understanding of Unix, this is fine. For others, it made installing Linux tedious and confusing.

Is it really so important that you understand precisely what you're installing? Yes, and here's why: Linux markedly differs from other operating systems in that no single entity controls development and testing. When you venture beyond Linux's kernel (the system's heart), Linux is composed of several thousand different tools, modules, libraries, and so forth.

Many of these components are derived from third-party, academic, freelance, and commercial developers all around the world. Each developer is responsible for their application's quality control, and hence your mileage might greatly vary. To understand why, please examine [Figure 3.1](#).

Figure 3.1

Various types of Linux software.

[Figure 3.1](#) shows various types of Linux software and an admittedly generalized critique of quality control at each level. Here's what it shows:

- The Linux kernel and must-have tools have been rigorously tested for common programming errors that could potentially threaten system security. The folks doing this testing have a lot of experience and are familiar with Linux source and development history, particularly from a security standpoint.
- *Semi-commercial tools* are tools that would be commercial on any other platform. Recently, there's been a huge influx of such tools as large corporate vendors move into Linux territory. These tools might have excellent security, but many probably don't. Porting complex commercial applications to Linux, a relatively new and unfamiliar operating system, is an error-prone enterprise. Furthermore, some vendors view Linux ports as policy decisions (testing the water) and allocate less time and effort to analyzing their port's security status, unless the application is specifically related to security.
- Finally, beyond core Linux code and semi-commercial contributions lie freelance, beta, and other tools. This category already makes up a substantial portion of Linux and is growing rapidly. Testing here varies. Many new Linux tools are the result of the well-intentioned, enthusiastic efforts of budding programmers. Some have long Unix experience and are well aware of security issues. Others might be just starting out.

As you move farther from Linux's basic core, you reap increasingly disparate results—with the notable exception of security tools. Some Linux security tools have reached levels of excellence equaled only in high-performance, commercial security applications.

If you're using Linux for personal use, you can install the entire distribution without worry. Just employ good security practices, back up often, and be prepared to learn through trial and error.

However, if you're using Linux for enterprise or mission-critical tasks, and therefore cannot tolerate error, take a different approach:

- Before employing Linux in your enterprise environment, learn a bit about software packages, what they do, how long they've been around, and whether you actually need them. For this, I recommend visiting the Linux Software Map at <http://www.boutell.com/lsm/>. The LSM is searchable, which is nice because there are currently about 3,000 entries.
- If your Linux distribution includes proprietary tools, investigate their utility and security track record. See Appendix D, "Sources for More Information," for more information about each distribution (bug lists, revision tracking sites, bulletins, vendor advisories, and so on).

Beyond these steps, try adhering to this cardinal rule: *Less is more*. Try installing only what you need.

This can be difficult, especially if you've just discovered Linux. Linux offers a wide range of applications and multiple subsets within each application type. Thus, in addition to the dozen text editors available on your distribution's CD-ROM, there are probably 25 more Linux text editors available. That's a lot of choices.

In particular, be extremely careful when you're choosing networked applications (anything that relies on a daemon). If a networked application has flaws, it can expose your system to remote attack. No other operating system offers as many networked applications as Linux. Indeed, Linux developers have gone hog-wild, networking everything from CD players to scribble pads. If it can be networked at all, Linux surely has networked it.

In short, before you install Linux in an enterprise environment, take the time to read about it. It's worth the effort, and you'll find your research interesting and enlightening. Linux is an operating system that's rich with possibilities and that supports truly amazing applications. For example, do you need DNA-sequencing tools or a means to view molecular structures? No problem. Go to <http://SAL.KachinaTech.COM/index.shtml>.

Finally, I should point out that even given all this, when Linux is properly installed and maintained, it offers *excellent* security. You simply need a Linux security overview, which is what this book is for, after all. Let's get started.

All Distributions Are Not Created Equal...

If you haven't chosen a distribution yet, now is the time to do so—but be aware that not all Linux distributions are the same or stress the same features. This can be difficult for first-time users to understand. After all, Linux is Linux, isn't it? Yes and no. As I've already mentioned, the installation procedures vary greatly among the different Linux distributions. Additionally, the feature sets vary—some versions are focused on the user experience, whereas others are aimed at creating a brick wall in terms of security. Unfortunately, many Linux distributions try to be everything to everyone and come up short.

The following is a short look at some of the current distributions and what sets each one apart from the pack:

Stampede Linux—Available for Intel and Alpha processors, Stampede provides a hardware-optimized port of Linux. This is not a good beginner distribution, but would work nicely for a network administrator or seasoned Unix professional. <http://www.stampede.org/>

Phat Linux—The Phat distribution is an excellent starting place for users who have been working with

Microsoft Windows and are unwilling to give up their Windows installation completely. Phat installs on an existing Windows partition and offers a full complete KDE-based Linux desktop environment. Installation is painless and extremely quick. <http://www.phatlinux.org/>

SuSE—Available for Alpha, PowerPC, Intel, and Sparc platforms, SuSE offers a simple installation process, large collection of included applications, and power features for the advanced user. One of the big SuSE advantages is out of the box support for a journaling file system. This can be used to create a very stable and fault-tolerant desktop or server. <http://www.suse.org/>

Yellow Dog—The Yellow Dog distribution is for PowerPC computers and is mainly intended to provide a secure and optimized Linux distribution for the Macintosh G3 and G4 series as well as IBM RS/6000 machines. If you're a Mac user looking for a simple transition from Mac OS, you're better suited running the standard LinuxPPC distribution. <http://www.yellowdoglinux.com/>

OpenLinux—OpenLinux originally described a single Linux distribution. Today it describes a family of distributions from Caldera. If you know what your Linux application will be, Caldera is the place to go. From ASP solutions to a secure desktop environment, Caldera offers distributions targeted to different applications, all with easy installation and excellent support. <http://www.calderasystems.com/>

Linux Mandrake—Based on the Red Hat distribution, Linux Mandrake is a Pentium-optimized distribution with graphical administration add-ons that make installation, updates, and file management a breeze. Although the Mandrake distribution is relatively new, it is quickly becoming a favorite of many users. In fact, PC Data ranked Mandrake as the number one selling Linux distribution in December 2000. <http://www.linux-mandrake.com/>

Red Hat—Red Hat is the powerhouse of Linux distributions. It has led the Linux charge into the workplace and, in many respects, is single-handedly responsible for making Linux a player in the enterprise workplace. Sporting a remarkably simple installer with auto-partitioning, RAID support, and desktop or server installations, it can create both secure desktop systems and powerful servers. Unfortunately, the introduction of Red Hat 7.0 alienated many longtime users with a restructured file system and other significant changes. If you're a first-time user, however, you'll be amazed at the polish given to the Red Hat distribution. Red Hat is available for Intel, Sparc, and Alpha systems. <http://www.redhat.com/>

Debian—Debian Linux is a popular distribution amongst advanced Linux/Unix users and system administrators. The installation process is not nearly as seamless as other distributions, but, at the same time, the quality of the included software and stability of the system as a whole are much greater. Debian does *not* bill itself as a Linux distribution, per se. Instead, it is a package of software and utilities that happens to run on the Linux kernel. Efforts are underway to port Debian to other kernels (BSD, and so on) as well. <http://www.debian.org/>

Slackware—The Slackware Linux distribution was the first popular distribution created. It enjoyed great success in the early and mid-1990s, but after a few years it started to lag behind the powerhouses such as Red Hat and SuSE. Recently, Slackware has been reborn and is now up-to-date with the latest applications and services. Although not as friendly as other distributions, Slackware has been described as "a Linux user's Linux" and offers hardcore users the tools and utilities they'll need to create an Internet server or desktop platform. <http://www.slackware.com/>

I've used most of the distributions in this list and have found them to be well constructed and useful. Your best bet, if you're undecided, is to try out a few distributions and see what suits you best. After you

decide on a route, stick with it. Switching between distributions can lead to confusion, as well as decreased efficiency in maintain your systems.

Partitions and Security

During installation, Linux will prompt you to partition your hard drive. This section will examine how your partitioning approach can affect your security.

What Are Partitions, Exactly?

Partitions are areas on your hard drive that are reserved for file systems. Let's look at their relationship to your hard drive at large.

Hard drives are composed of one or more layers called *platters*. Older SCSI drives, in particular, often house multiple platters. Please see [Figure 3.2](#).

Figure 3.2

Hard drives can have one platter or several.

Each platter's surface vaguely resembles the surface of a vinyl record. Please see [Figure 3.3](#).

As depicted in [Figure 3.3](#), platters are covered by groove-like structures, circles that get increasingly smaller as they get closer to the center. The spaces between these circles are *tracks*. Tracks are divided into smaller units called *sectors*, which contain even smaller units that record data bits.

Figure 3.3

Your hard drive's tracks, sectors, and data.

The total number of tracks that occupy the same region on all platters form a *cylinder*. Please see [Figure 3.4](#).

Figure 3.4

All tracks occupying an identical area form a cylinder.

Partitions are composed of a user-specified range of contiguous cylinders. With DOS and Windows 3.11 (and even Windows 95's early release), users needed only one partition. This occupied virtually the entire disk and contained system files, user files, and swap files. Please see [Figure 3.5](#).

Figure 3.5

The DOS partition occupies almost the entire disk.

Note - As hard drives larger than 2 gigabytes have become more affordable, this has changed. DOS/Windows and the first release of Windows 95 could only handle 2GB or less. Hence, to accommodate a large disk, you had to format it in 2GB partition increments, in which your first partition would be Drive C:, your second partition would be Drive D:, and so on. Later releases of Windows impose no such restriction.

In Linux, it's more common to have multiple partitions, primarily to maintain strict control over where data ends up. Normally, when you use only a single partition (as you would with DOS), your operating system writes data arbitrarily wherever it finds suitable space, and so do users. Eventually, your data becomes spread out, unmanageable, and disorganized.

In contrast, things are a bit more orderly when you create multiple partitions. For example, you can separate swap files from your live file system. Each partition exclusively owns a specific disk area. Figure 3.6 depicts a fairly common partitioning scenario.

Another common scenario is when you install two or more operating systems on the same disk drive but on different partitions, and they can coexist problem-free.

Linux supports a wide range of partition types. Table 3.1 lists a few of the more interesting ones.

Figure 3.6

Here, the disk has two swap partitions and one native file Linux partition.

Table 3.1 Various Partition Types Supported by Linux

Number	Partition Type
2	XENIX root, an antiquated, Unix-based operating system for PCs that is rarely used today. It has a long history. Originally based on Unix version 7, later incorporating features from BSD 4.1, and finally conforming to SYS V, XENIX has been marketed by many companies, including Microsoft and the Santa Cruz Operation (SCO).
7	The High Performance File System or HPFS, a fault-tolerant system that incorporates advanced caching, long filenames, and support for traditionally incompatible file structures. It is the basis for the OS/2 system. Learn more about HPFS at http://www.cs.wisc.edu/~bolo/shipyard/hpfs.html .
8	AIX (IBM Unix).
40	Venix 80286, a System V-compatible version of Unix from VentureCom.
63	GNU HURD, which hails from the Free Software Foundation and will eventually be a replacement for the Unix kernel. To learn more about HURD, go to http://www.gnu.org/software/hurd/hurd.html .
64	Novell NetWare.
81	Minix.
82	Linux swap partition.
83	Linux native partition.

93

Amoeba, a distributed operating system that runs on SPARCstations (Sun4c and Sun4m), as well as the 386/486, 68030, Sun 3/50, and Sun 3/60. Amoeba is used to pool the power of multiple workstations into one powerful block of computing power. Learn more about Amoeba at <http://www.cs.vu.nl/pub/amoeba/>.

Linux supports more partitions than those listed here. For a complete list, go to <http://mm.iit.uni-miskolc.hu/Data/texts/Linux/SAG/node35.html>. Also, for a complete list of all PC partition types (including those Linux does not support) go to http://www.win.tue.nl/math/dw/personalpages/aeb/linux/partitions/partition_types-1.html.

Many folks install both DOS/Windows and Linux on the same hard drive, on separate partitions. This offers them latitude and flexibility. They can learn Linux while still relying on Windows, and enjoy at least one-way compatibility. Please see [Figure 3.7](#).

Figure 3.7

Linux and DOS/Windows can coexist, but only Linux offers compatibility.

Although DOS and Windows cannot access the Linux partition, Linux can access the DOS partition, thereby allowing you to copy files back and forth across file systems.

Note - During installation, Linux asks you to specify additional or foreign file systems that you'd like to access. Linux mounts those file systems in the directory of your choice. A typical configuration would be to mount the DOS file system from Linux in `/dos`.

Linux newcomers often use the configurations depicted in [Figures 3.6](#) and [3.7](#) because they're easy to implement. Many new Linux users are satisfied if they can just complete the installation with no problems, so they're apt to avoid more complicated partitioning schemes. Moreover, few installation routines highlight the relationship between partitioning and security, and give no hint that such configurations are risky.

In fact, the scenarios depicted in [Figures 3.6](#) and [3.7](#) expose your system to attack and hinder your ability to exercise effective system administration.

If you'd like to automatically manage your partitions, I suggest that you use a distribution such as Red Hat 7.x. During installation, Red Hat gives you the option of automatically partitioning your drives. The result is shown here:

```
[root@bcdinc jray]# fdisk /dev/hda

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 784 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot   Start     End  Blocks  Id System
```

```
/dev/hda1 *      1      3      24066  83 Linux
/dev/hda2        4      784    6273382+  5 Extended
/dev/hda5        4      338    2690856  83 Linux
/dev/hda6       339      673    2690856  83 Linux
/dev/hda7       674      706    265041  83 Linux
/dev/hda8       707      739    265041  83 Linux
/dev/hda9       740      772    265041  82 Linux swap
```

Command (m for help):

This is a rather complex partitioning scheme that sets up separate boot, user, and swap partitions. These partitions are then automatically mounted as well:

```
[root@bcdinc jray]# mount
/dev/hda8 on / type ext2 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/hda1 on /boot type ext2 (rw)
/dev/hda6 on /home type ext2 (rw)
/dev/hda5 on /usr type ext2 (rw)
/dev/hda7 on /var type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
```

Again, if you're a first-time user, Red Hat's automatic partition system makes installation as easy as Windows or Mac OS. If you've decided to use another distribution or partition the drive manually, there are several rules you should follow.

Lumping Linux into a Single Partition

First, you should never put root and user file systems on the same Linux partition. If you do so, you increase the chance that attackers can exploit SUID programs to access restricted areas.

Note - SUID files are special in that they always execute with owner privileges, no matter who runs them. For example, if root owns a SUID program, that program will execute with root privileges and have considerable power to access, alter, and overwrite files that might otherwise be unreachable. If an attacker can exploit weaknesses in SUID programs, he can threaten the system at large. (Learn more about SUID programs in Chapter 4, "Basic Linux System Administration.")

Additionally, lumping Linux into a single native partition makes your life as a system administrator difficult. For example, it might hinder your ability to incisively update or back up individual packages or file systems. And when the full Linux system occupies one partition, even limited file corruption can cause systemic problems (meaning that one corrupted directory hierarchy can affect others). Disk optimization (which is something you rarely even have to consider under Linux) is another problem under a single partition system. As new software is installed, old software is removed, kernels are updated, and so on, fragmentation will increase. Although there are tools to optimize Linux disks, they are unreliable and a pain to use. Most frequently the only real maintenance that can be performed on a single-partition system is to reinstall the operating system.

To avoid these problems, create a separate partition for each major file system. [Figure 3.8](#) depicts one possible configuration.

Figure 3.8

All major file systems are on separate partitions.

This enhances security and makes backups and recovery manageable. You can specify different backup schedules for different partitions, system files are separated from data files, and so on. This approach also allows you to exercise more stringent control over each file system and how it is mounted.

Note - The term *mount* refers to how Linux makes different file systems available to you. When Linux mounts a local or foreign file system, it attaches the system to a local device and/or directory. This gives you an access point. For example, to grant you access to your CD-ROM, Linux associates the CD-ROM drive with the device `/dev/cdrom` (usually), and you must specify a directory as the mount point (typically, `/mnt/cdrom` or `/cdrom`). From that point on, your CD-ROM's top-level directory is accessible in `/cdrom` and its subdirectories are available beneath it (`/cdrom/docs`, `/cdrom/install`, `/cdrom/source`, and so on).

At system startup, Linux mounts all available file systems per the specifications set forth in `/etc/fstab`. You can use `/etc/fstab` to incisively control how users and the system access your partitions. Let's quickly cover `/etc/fstab` now.

`/etc/fstab`

`/etc/fstab` is the plain text file in which you specify file system mount options. Each line addresses one file system. For example, the following entry specifies mount options for an MS-DOS file system mountable in `/dos`:

```
/dev/hda4 /dos msdos defaults 1 1
```

The line consists of six fields:

- The file system specification—Here you specify either the block device or file system to be mounted—in this case, partition 4 on the first drive. This is what Linux will mount.
- The file system file location—This is the mount point—in this case, it's `/dos`, a common naming for a DOS file system mount point, as discussed earlier.
- The file system type—In this field, you describe the file system's type: Minix, extended, DOS, HPFS, iso9660/CDROM, Network File System (NFS), or swap.
- The file system mount options—Here you specify the level of access that users and the system will have on this mounted file system. Here's where security comes in. Your choices are as follows:



defaults	Everything (quota, read-write, and suid).
noquota	No quotas, generally.
nosuid	No SUID access.
quota	Quotas are active.
ro	Read-only.
rw	Read-write.
suid	SUID access is okay.

- File system dump parameters—This is a numerical value to flag file systems that need to be dumped (backed up).
- File system check sequence number—Here you specify the file system's priority for integrity checks performed by `fsck`. (`fsck` is a file system integrity checker that examines file systems at boot by default.)

Where should you force a `nosuid` mount? Anywhere that local or remote users might be up to no good. For example, suppose that you anticipate providing anonymous FTP services (not a great idea). If so, consider creating a separate partition for this and have Linux mount it `nosuid`. This still allows data to be written but addresses the `SUID` problem.

Other Advantages of Multiple Partitions

So, multiple partitions offer you at least four advantages:

- Easy backup and upgrade management
- Faster booting (in some cases)
- The ability to control how each file system is mounted
- Protection against renegade `SUID` programs

There are other advantages. One is that the multipartition approach prevents accidental denial of service and shields your root file system from overflow. For example, `/var` stores logging information. If you have just a single partition containing root, `/usr`, `/var`, and `/tmp`, logs in `/var` can potentially flood your entire file system (and users can too).

Sizing Out Partitions

As noted, new users sometimes shy away from multiple partitions (beyond swap and root). That's because creating multiple partitions forces you to make some hard choices. For example, just how large should each partition be? Unfortunately, there's no definite answer to this question except when you're dealing with swap and root partitions. Swap partitions are typically twice the size of real memory available (recent decreases in RAM pricing make this unnecessary), and root should have 64MB minimum (although I allocate 100MB).

In respect to other file systems, you'll make your choices depending on different factors. One factor is what you intend to do with your Linux box. On a multiuser system, you'll want to give your users at least 20MB each (and probably more). Hence, for 10 users, you'll need a `/home` partition of at least 210MB.

Some of these values are interdependent. For example, if you're accommodating many users and providing mail and news services, your `/var` and `/home` partitions will need to be substantial. Unless, of course, users use third-party mail and news solutions. In that case, their messages will be stored in their `/home/user` directory; for example, `/home/user/.netscape/`.

If you run a firewall, you'll need a large log directory hierarchy (`/var`), and this should have its own partition. In fact, you might be forced to put this partition on a separate disk drive. That way you'll avoid losing valuable audit information if the primary file system is corrupted.

However, in most cases your largest partitions will house the `/usr` and `/home` directories.

Note - Some Linux distributions are moving towards storing more dynamic data in the `/var` directory than they did previously. Red Hat 7.x, for example, assumes the Apache root to be `/var/http://www`. Take this into consideration when partitioning.

Let's look at a conservative example. Here's a `df` report from a 1.6GB IDE hard drive with a 128MB swap partition that isn't visible from the `df` query:

```
Filesystem      1024-blocks  Used Available Capacity Mounted on
/dev/hda2       66365   17160  45778    27% /
/dev/hda5       373695   1549  352845    0% /home
/dev/hda6       703417  344725  322356    52% /usr
/dev/hda7       127816   21235  99981    18% /var
/dev/hda8       123919     22  117498    0% /tmp
```

Here's the `fstab` information immediately after installation:

```
/dev/hda2 / ext2 defaults 0 1
/proc /proc proc defaults 0 0
/dev/hda1 none swap defaults 0 0
/dev/hda5 /home ext2 defaults 0 2
/dev/hda6 /usr ext2 defaults 0 2
/dev/hda7 /var ext2 defaults 0 2
```

```
/dev/hda8 /tmp ext2 defaults 0 2
#
/dev/fd0 /mnt/floppy ext2 defaults,noauto 0 0
#
/dev/hdb /mnt/cdrom iso9660 ro,noauto 0 0
```

Note partitions 5, 6, 7, and 8. These are logical partitions. You're allowed only four primary partitions in the Intel world, or three primary partitions, one extended partition, and multiple logical partitions. To create additional partitions, first establish an extended partition and then slice this into logical partitions using either `fdisk` or, if you have Red Hat, Disk Druid.

Caution - Some distributions offer user-friendly installation routines that automatically suggest disk layout (much like Sun's Solaris does). These routines are convenient, but think carefully before accepting such a partitioning scheme. Automatic partitioning does not take into account the way that the system will be used. Instead, it creates a generalized partition table that doesn't necessarily work well with Web or file servers. For beginners, however, automatic disk layout is a great way to create a solid file system foundation with very little effort.

Although you've probably used `fdisk` already, some folks who purchased this book might not have installed Linux yet. For their benefit, I'll briefly address `fdisk` here. If your Linux distribution doesn't use `fdisk`, keep reading. `Cfdisk` and Disk Druid are both discussed later in the chapter.

fdisk

`fdisk` is a partition manipulator for Linux. During your installation, Linux will move you from a semi-graphical environment to a command-line interface so that you can partition your disks. At that point, you'll almost certainly be dealing with `fdisk`.

`fdisk`'s initial prompt will look much like this:

```
Using /dev/hda as default device!
```

```
The number of cylinders for this disk is set to 1579.
This is larger than 1024, and may cause problems with:
1) software that runs at boot time (e.g., LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
```

```
Command (m for help):
```

Before continuing, if you're using `fdisk` for the first or even the fifth time, review the list of valid commands. That way, you can familiarize yourself with each one and reduce the chance of error. To view the complete command set, type `m` and press Enter. In response, `fdisk` will print a help menu:

```
Command action
  a toggle a bootable flag
  b edit bsd disklabel
  c toggle the dos compatibility flag
```

```

d delete a partition
l list known partition types
m print this menu
n add a new partition
o create a new empty DOS partition table
p print the partition table
q quit without saving changes
s create a new empty Sun disklabel
t change a partition's system id
u change display/entry units
v verify the partition table
w write table to disk and exit
x extra functionality (experts only)

```

Also, examine the current partition table before you make any changes. That way, you can verify whether any partitions already exist. To do so, type **p** and press Enter. If you're working with an unpartitioned disk, `fdisk` will print a blank table:

```

Disk /dev/hda: 32 heads, 63 sectors, 1579 cylinders
Units = cylinders of 2016 * 512 bytes

```

```

Device Boot Start End Blocks Id System

```

```

Command (m for help):

```

Now you're ready to begin creating your partitions.

From here on, I'll stick with the values from the preceding partitioning example. You'll need to adjust partition sizes according to your own needs. This is merely a walkthrough that demonstrates how to create an extended partition and logical partitions within it. Few Linux how-to books address this issue. (Most such books focus on Red Hat installations. Red Hat includes Disk Druid, a semi-graphical tool that simplifies the process for you. However, you might be installing another distribution, one with command-line `fdisk`. If so, this next section will illustrate the steps required when you're creating such partitions by hand.)

Creating the Swap and Root Partitions

First, you'll need to create your swap and root partitions. In this example, I'll assume that you're installing to a new hard drive, with no other existing file systems previously installed.

To create a new partition, type **n** and press Enter. In response, `fdisk` will ask you what style of partition you want. Type **p** and press Enter for primary:

```

Command Action

```

```

e extended
p primary partition (1-4)
p

```

`fdisk` will then ask you to number the new partition. This is your first primary partition and will house your swap file, so choose 1:

```

Partition Number (1-4): 1

```

Next, `fdisk` will ask you to specify where the partition starts. This is your first partition and you want to write it from the first cylinder onward, so choose 1:

```
First cylinder: (1-1579) 1
```

Finally, to complete the cycle, `fdisk` will ask you to size the partition. Swap file size is a matter of personal preference. In past years, Linux tutorials prescribed a ratio approach: "If you have 8MB of RAM, you'll need a 16MB swap file, minimum." With the cost of 128MB RAM falling well below \$100, it is rarely necessary to rely on swap space.

As per the preceding example, choose 128MB (based on 64MB of physical RAM):

```
Last cylinder or +size or +sizeM or +sizek (1-1579): +128M
```

After you create each partition, reexamine the `fdisk` partition table. This way, if you make typographical errors, you can catch them before writing changes to disk. Here's what the updated table will look like after you create the first partition:

```
Command (m for help): p
```

```
Disk /dev/hda: 32 heads, 63 sectors, 1579 cylinders
Units = cylinders of 2016 * 512 bytes
```

```
   Device Boot   Start    End  Blocks  Id System
/dev/hda1          1    130  131008+  83 Linux native
```

Note that the partition is type 83 (Linux native). You need to change this. This partition is a swap partition, and you must manually designate it as such. To do so, type `t` and press Enter:

```
Command (m for help): t
```

In response, `fdisk` will prompt you for the partition number. Choose 1:

```
Partition number (1-4):1
```

Finally, `fdisk` will ask which partition type you want. Choose 82 to convert the partition to a Linux swap:

```
Hex Code (L to list): 82
```

When you reexamine the partition table, `fdisk` will reflect the changes:

```
Command (m for help): p
```

```
Disk /dev/hda: 32 heads, 63 sectors, 1579 cylinders
Units = cylinders of 2016 * 512 bytes
```

```
   Device Boot   Start    End  Blocks  Id System
/dev/hda1          1    130  131008+  82 Linux swap
```

Next, create the root partition. Here again, size is a matter of personal preference. You should allocate at least 32MB to root, although I've seen people make this partition as large as 100MB. In any case, the

procedure is precisely the same. You start by creating a new partition. Type **n** and press Enter. Then `fdisk` will ask what style of partition you'd like. Again, type **p** and press Enter for primary:

Command Action

```
e extended
p primary partition (1-4)
p
```

Then `fdisk` will ask you to number the new partition. This will be your second primary partition, so choose **2**:

Partition Number (1-4): **2**

In response, `fdisk` will ask you to specify where the partition starts:

First cylinder: (131-1579)

Note that the first valid starting cylinder is now 131. That's because your swap partition occupies cylinders 1 through 130. Therefore, you'll start your root partition at cylinder 131:

First cylinder: (1-1560) **131**

And finally, `fdisk` will ask you to size the partition. For this example, allocate 64MB:

Last cylinder or +size or +sizeM or +sizek (131-1579):**+64M**

The results show a Linux (type 82) swap partition and a root (type 83) partition:

Command (m for help): **p**

```
Disk /dev/hda: 32 heads, 63 sectors, 1579 cylinders
Units = cylinders of 2016 * 512 bytes
```

```
   Device Boot   Start    End  Blocks  Id System
/dev/hda1            1   130  131008+  82 Linux swap
/dev/hda2           131   198   68544   83 Linux native
```

Creating the Extended Partition

The next step is to create an extended partition that will occupy the remaining disk space. To create an extended partition, type **n** and press Enter (new), and then choose **e** for extended:

Command Action

```
e extended
p primary partition (1-4)
e
```

Here, `fdisk` will ask you to specify the extended partition's first cylinder. In this case, the first available cylinder is 199, so choose that:

First cylinder: (199-1579):**199**

Finally, `fdisk` will ask you to specify the extended partition's last cylinder. In general, you should go with the very last cylinder. That way, the extended partition occupies the remaining disk space. However, you choose to leave some space at the end of the disk, so specify cylinder 1560:

```
Last cylinder or +size or +sizeM or +sizek (199-1579): 1560
```

Here are the results:

```
Command (m for help): p
```

```
Disk /dev/hda: 32 heads, 63 sectors, 1579 cylinders
Units = cylinders of 2016 * 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1		1	130	131008+	82	Linux swap
/dev/hda2		131	198	68544	83	Linux native
/dev/hda3		199	1560	1372896	5	Extended

The table now shows one Linux swap, one Linux native, and one Linux extended partition. Your remaining task is to allocate several logical partitions.

Creating Logical Partitions Within the Extended Partition

Now that `fdisk` is aware of an extended partition, the `fdisk` menu will change. To create your first logical partition (for `/home`), type `n` and press Enter. In response, `fdisk` offers a new menu. Here, choose `l` for logical:

```
Command Action
```

```
l  logical (5 or over)
p  primary partition (1-4)
l
```

Then `fdisk` will ask you to specify the new logical partition's first cylinder. Note that the first available cylinder is 199, which is the same first available cylinder that you specified for the extended partition. That's because your logical partitions will lie on top of the extended partition. So, choose 199:

```
First cylinder: (199-1579):199
```

Finally, `fdisk` will ask you to specify this logical partition's last cylinder. To give `/home` 370MB, choose 581:

```
Last cylinder or +size or +sizeM or +sizek (199-1579): 581
```

Here are the results so far:

```
Command (m for help): p
```

```
Disk /dev/hda: 32 heads, 63 sectors, 1579 cylinders
Units = cylinders of 2016 * 512 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1		1	130	131008+	82	Linux swap

```

/dev/hda2      131   198   68544   83 Linux native
/dev/hda3      199   1560  1372896  5 Extended
/dev/hda5      199   581   386032+ 83 Linux native

```

You add the remaining partitions, /usr, /var, and /tmp, in the same fashion. Here's the sequence for /usr:

Command Action

```

l  logical (5 or over)
p  primary partition (1-4)
l

```

```

First cylinder: (582-1579):582
Last cylinder or +size or +sizeM or +sizek (581-1579): 1302

```

Here's the sequence for /var:

Command Action

```

l  logical (5 or over)
p  primary partition (1-4)
l

```

```

First cylinder: (1303-1579):1303
Last cylinder or +size or +sizeM or +sizek (1303-1579): 1433

```

And finally, the sequence for /tmp:

Command Action

```

l  logical (5 or over)
p  primary partition (1-4)
l

```

```

First cylinder: (1433-1579):1303
Last cylinder or +size or +sizeM or +sizek (1433-1579): 1560

```

When you view the final results, fdisk will reflect the following changes:

Command (m for help): **p**

```

Disk /dev/hda: 32 heads, 63 sectors, 1579 cylinders
Units = cylinders of 2016 * 512 bytes

```

```

   Device Boot   Start    End  Blocks  Id System
/dev/hda1            1    130  131008+  82 Linux swap
/dev/hda2           131    198   68544   83 Linux native
/dev/hda3           199   1560  1372896   5 Extended
/dev/hda5           199    581   386032+  83 Linux native
/dev/hda6            582   1302  726736+  83 Linux native
/dev/hda7          1303   1433  132016+  83 Linux native
/dev/hda8          1434   1560  127984+  83 Linux native

```

After you've achieved and verified your desired results, choose w. This will exit fdisk and permanently commit these changes to disk. Linux will then return you to the main installation program.

Other Partitioning Tools

Not every Linux installation program directs you to `fdisk` for partitioning. Instead, you might work with `cdisk` or Disk Druid. These tools are much easier to use.

`cdisk`

`cdisk` is a Curses-based partition manipulator for Linux.

Note - *Curses* is a development package for creating menu-based programs on Unix terminals. Curses applications vaguely resemble old DOS programs, in that you can navigate menu choices by using arrow keys. Traditional Curses applications have a black background and white foreground. Menu choices appear in white until highlighted with a white bar, at which point the highlighted text turns black. Learn more about Curses programming at <http://dickey.his.com/ncurses/ncurses-intro.html>.

`cdisk` presents a comfortable and easy-to-navigate interface. Please see [Figure 3.9](#).

Figure 3.9

Partitions viewed in `cdisk`'s Curses environment.

For the most part, you'll have no trouble navigating `cdisk` using arrow keys—the program provides ample help along the way. However, I've provided a summary of important `cdisk` keystrokes and their functions in Table 3.2. This is in the event that on your first installation, you're forced to use `cdisk` but have little or no accompanying documentation—a common problem.

Table 3.2 Keystroke Commands in `cdisk`

Key	Function
?	Get help.
b	Set (or unset) the highlighted partition as bootable.
d	Delete the highlighted partition.
g	-Enter an expert mode where you can alter the disk's listed geometry. Warning: Use this function with caution. This is much like specifying your own disk drive settings (heads, cylinders, blocks) in your BIOS. Chances are that <code>cdisk</code> 's auto-detected values are correct. If you specify erroneous values, your Linux system may not boot.
h	Get help.
n	Create a new partition.
p	Obtain and print the current partition table information.
q	Quit <code>cdisk</code> .

t	Change the file system type (much like <code>t</code> works in <code>fdisk</code>).
W	Write changes to disk. (You must issue the <code>w</code> command in uppercase.)

Disk Druid

Disk Druid, common to Red Hat installation as a `fdisk` alternative, is even easier to use. The is entirely graphical. Please see [Figure 3.10](#).

Figure 3.10

Disk Druid's opening screen.

To add your partitions, highlight the Add button and press `Enter`. In response, Disk Druid displays a dialog box with all the options you'll ever need. Please see [Figure 3.11](#).

Figure 3.11

Disk Druid's partition editing screen.

Summary of Partitions and Security

Because partitioning has a strong bearing on your system security, you should carefully weigh your options before installation. Making your final decisions will never be easy.

Balancing disk load is probably the most challenging aspect of partitioning, particularly with smaller disks. By creating multiple partitions, you limit each file system's ability to grow. In certain instances, of course, that's exactly what you want. However, it's irritating to later discover that you failed to allocate adequate disk space.

One thing that can help is to know each major file system's purpose. Here they are, in short order:

- `/`—Houses relatively few files (mostly startup scripts).
- `/usr`—Houses most of your software.
- `/home`—Houses your user directories.
- `/opt`—This is for third-party add-on software (Netscape, StarOffice, and so on).
- `/var`—Houses garden-variety administrative logs, mail, and news.

Disk balancing also gets easier if you develop policies for a consistent application set. For example, perhaps you limit third-party software to Netscape Communicator, StarOffice, and Corel WordPerfect. This eliminates the need for a large `/var` partition and gives you a ballpark figure on how large `/opt` has to be.

Of course, there's no law mandating that you create a dozen partitions. The partition parameters in the preceding examples are for demonstration purposes only. You can get along nicely with just three partitions, especially if only a few trusted users have access to your Linux system. Only you can

accurately assess how many partitions you'll need and which file systems to segregate.

Here are some closing tips:

- You might prefer fewer partitions, or you might want to prioritize file systems that must or should be segregated. If so, the important file systems to house on separate partitions are root (/), /var, and /tmp from a security viewpoint, or root (/), /var, and /usr from an administrative viewpoint. At bare minimum, I *strongly* advise housing root on its own partition.
- If you allocate partitions to non-Linux operating systems, carefully consider how you want Linux to mount them. For example, suppose that you have a small Windows partition at the beginning of the disk. If you use this partition almost exclusively when in Windows, consider having Linux mount it read-only or not at all. That way, you protect it from either accidental or intentional damage.
- If you're running a firewall, sniffer, or other network-monitoring device, funnel logs to their own partition (preferably on another disk).
- Exercise care when setting partition mount options. Sometimes, restrictive policies can lead to administrative headaches. For example, suppose that you decide to lump contributed binaries into /usr/local and have Linux mount /usr/local read-only. Later, this might hamper your ability to perform upgrades without first redefining the mount option.

Finally, here are some resources for more information on partitioning:

- *Debian Linux Installation & Getting Started* by Boris D. Beletsky (borik@isracom.co.il). The author takes you through each step of installation, with special focus on disk partitioning. Find it at <http://www.debian.org/releases/stable/#new-inst>.
- *Installing Red Hat Linux*. The Red Hat installation manual details the partition utilities available under Red Hat, including fdisk and Disk Druid. http://www.redhat.com/support/docs/installing_linux.html.
- *Linux Installation and Getting Started* by Matt Welsh. Although slanted heavily toward SlackWare, this document is superb, stepping through every aspect of installation and partitioning in excruciating detail. Find it at <http://durak.org/sean/pubs/ligs-slackware/node1.html>.
- *The Linux Disk HOWTO* by Stein Gjoen (sgjoen@nyx.net). The author discusses drive geometry and structure, disk layout, partitioning, and so forth, in great detail. Find it at <http://www.ict.pwr.wroc.pl/doc/Linux-HOWTO/Disk-HOWTO.html>.
- *The Linux Partition HOWTO* by Kristan Koehntopp & Toni Harris (kris@koehntopp.de). The author discusses important issues about disk balancing, partition sizes, and so on. Find it at <http://ldp.linuxisgod.com/HOWTO/mini/Partition/>.

Choosing Network Services During Installation

As noted earlier, Linux supports many network services. Your job is determining which ones you need. Network services come in two basic flavors:

- Services that deliver information to clients for human consumption. For example, a Web server, which allows users to download documents and media.
- Services that deliver information to clients or hosts for network and operational purposes. For example, Dynamic Host Configuration Protocol, which automatically sets up clients' network configuration.

Network services that provide people with data or functionality are generally not essential. Instead, they are privileges and niceties that you afford your users, and you'd profit by viewing them that way. Indeed, because almost every service you run will complicate system administration and security, the fewer you allow the better. Here are some nonessential services that provide people with data or functionality:

- `bootpd`—A server that can implement the bootstrap protocol, which allows you to boot diskless clients from a server. During startup, a diskless client queries the server and discovers its IP address. It also loads any files specified by the server. (Typically, the server forwards a boot program.) Don't run `bootpd` if you don't need it.
- `fingerd`—The `finger` server, which gathers personal information on specified users, including their username, real name, shell, directory, and office telephone number (if available). On request, `fingerd` forwards this information to anyone using a `finger` client. Here's an example of what `fingerd` returns:

```
Login name: unowen           In real life: U. N. Owen
Directory: /home/unowen     Shell: /sbin/sh
On since Feb 3 18:13:14 on pts/15 from ppp-208-19-49-133.samshacker.net
Mail last read Wed Feb 3 18:01:12 1999
```

- This isn't essential by any means. In fact, it might expose your users and your Linux server to unwanted invasions of privacy. Disable `fingerd` unless you have a good reason not to. To do so, comment it out in `/etc/inet.d` by placing a `#` symbol at the beginning of the `finger` definition line). Users of `xinetd` can either remove the server file from their `/etc/xinetd.d` directory, or set `disable = yes` within the service file.
- `ftpd`—File Transfer Protocol (FTP), which provides standard file transfer over internetworks. Today, there's less reason to run an FTP server. The WWW has made it easy to distribute files using HTTP, which most users are more familiar with anyhow. If you *are* going to provide FTP services, see Chapter 11, "FTP Security."
- `httpd`—The Hypertext Transfer Protocol server. This is your Web server. Without a doubt, you'll want to provide at least limited Web services. Check Chapter 14, "Web Server Security," for ways to tighten access control and general Web security.
- `nfs`—Network File System, a system that allows you to transparently import files from or export file systems to remote hosts. These files appear and act as though they were installed on your local machine. NFS is useful in many situations. For example, if you're hosting Web servers for third parties (running a Web farm), you can run exports to a RAID server. That way, all user Web directories are actually stored on a single server, redundant and prepared for possible individual host failures. To users, who maintain their own Web pages, everything appears to be local when they telnet or FTP into their co-located box.

- NFS has many other uses, too. However, if you don't need it, don't install or enable it. NFS has some security issues, even though secure NFS systems do exist. Learn more in Chapter 15, "Secure Web Protocols."
- `nntpd`—Network News Transfer Protocol server. This is the Usenet news server. Today, most people get Usenet news from their ISP's feed, so there's little reason to run NNTP yourself.
- `rlogind`—The `rlogin` (remote login) server. `rlogin` is an `r` service that allows users to conduct remote terminal sessions, much like `telnet` does. A major difference between `rlogin` and `telnet` is that `rlogin` allows users to set up passwordless access on trusted hosts with trusted users. You probably don't want this.
- `rshd`—The remote shell (`rsh`) server. `rsh` allows users to execute commands on remote hosts running `rshd`. This is a member of the `r` services family (`rsh`, `rlogin`, and so on), which is a notorious security hazard. Carefully consider whether you need to provide such services.
- `talkd`—The `talk` server. `talk` is an interactive chatting system for Linux that splits each user's screen in half. The top half echoes the requesting party's keystrokes, and the bottom echoes the responding party's keystrokes. Is this essential? Hardly. However, if your system is in-house (not wired to the Net), you might want to keep `talk` for quick interdepartmental communication.
- `telnetd`—The `telnet` server. Although `telnet` can increase risk, it is indispensable for some administrative tasks, so you'll probably want it. Check Chapter 13, "Telnet and SSH Security," for ways to lock down `telnet` and keep it useful but safe.
- `sshd`—The secure shell server. The Secure Shell Daemon provides a replacement to `telnet` that offers full traffic encryption. This is a good thing to keep around if you want to perform remote administration.
- `tftp`—Trivial File Transfer Protocol (TFTP). TFTP is an antiquated means of transferring files. You probably don't need it.

These are just a few examples, Chapter 14 has a larger list with descriptions. A default installation could result in many more nonessential services cluttering up your system and eroding its security. For this reason, whenever possible, you should run a verbose installation and explicitly reject packages that you don't need.

Five Minutes to a More Secure System

After installing Linux, the first thing you should do is assess the services that are running on your computer. Linux starts services in one of two primary ways: by running them at boot time, or by a connection request occurring at the appropriate port.

`inetd` and `xinetd`

The first place to look for unnecessary services is in the `/etc/inetd.conf` file or `/etc/xinetd.d` directory. The `inetd` process is the Internet daemon. It listens on the appropriate port for a service request, and then starts the corresponding server to process the request. If you have a system that uses `inetd`, check your system for the presence of `/etc/inetd.conf`. A sample of the file contents are shown here:

```
#echo stream tcp nowait root internal
#echo dgram udp wait root internal
#discard stream tcp nowait root internal
#discard dgram udp wait root internal
#daytime stream tcp nowait root internal
#daytime dgram udp wait root internal
#chargen stream tcp nowait root internal
#chargen dgram udp wait root internal
#time stream tcp nowait root internal
#time dgram udp wait root internal
#
# These are standard services.
#
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
#
# Pop and imap mail services et al
#
#pop-2 stream tcp nowait root /usr/sbin/tcpd ipop2d
#pop-3 stream tcp nowait root /usr/sbin/tcpd ipop3d
#imap stream tcp nowait root /usr/sbin/tcpd imapd
```

Each line denotes a particular service that can be activated on your system. If a line in this file begins with a pound sign (#), it is disabled. Unless you are absolutely sure that you need a service, you can comment out any services that are currently enabled on your system. After making your changes, either reboot your system or reload the `inetd` configuration by finding the `inetd` process ID (`ps ax | grep inetd`), and then issuing the command `kill -1 <inetd PID>`.

If you've looked for the `/etc/inetd.conf` file and can't find one, you're probably running `xinetd`, an advanced replacement for the standard Internet daemon. Check the system for the presence of `/etc/xinetd.conf`. If it exists, open it.

```
service ntalk
{
    socket_type      = dgram
    wait            = yes
    user            = nobody
    group           = tty
    server          = /usr/sbin/in.ntalkd
}
```

Each of the `xinetd` services is configured in a block, as seen here. To disable a service, comment out the entire block, or add a line reading `"disable=yes"` to the block. Depending on your setup, your `/etc/xinetd.conf` file might include a line like this:

```
includedir /etc/xinetd.d
```

If you see such a line, `xinetd` will read the contents of that directory and look in each of the contained files for other service definitions. Typically a single service is defined for each file. To disable services on a system that uses this option, simply move the service files from the included directory (usually `/etc/xinetd.d`), or add the `"disable=yes"` line to the service file you want to incapacitate.

Runlevel Services

Runlevel services, rather than starting when requested, start when the system boots. These services are

usually defined by creating a symbolic link between service start and stop files in `/etc/rc.d/init.d` and the runlevel directories `/etc/rc.d/rc#.d`, where the # varies between 0 and 6. (Unless you've modified your system, runlevel 3 is the standard text boot mode, whereas runlevel 5 is the graphical X Window startup.)

To view the services included on your system, list the `/etc/rc.d/init.d` directory:

```
[jray@bcdinc rc.d]$ ls /etc/rc.d/init.d/
anacron  functions  keytable  network  rawdevices  single  ypbind
apmd     gpm        killall   nfs       rhnsd       snmpd   yppasswdd
arpwatch halt       kudzu    nfslock  rstatd     sshd    ypserv
atalk    httpd     linuxconf  pcmcia   rusersd    syslog
atd      identd   lpd       portmap  rwalld     webmin
cron     ipchains  named     pppoe    rwhod      xfs
dhcpd    kdcrotate netfs     random   sendmail   xinetd
```

To see which are running on your system, first determine the runlevel you're at. (Again, if you're in text mode, 3 is a good guess. If not, use 5.) Then list either the `rc3.d` (runlevel 3) or `rc5.d` (runlevel 5) directory:

```
[jray@bcdinc rc.d]$ ls /etc/rc.d/rc3.d
K00linuxconf K20rstatd  K60lpd    K87portmap  S40atd      S85gpm
K01kdcrotate K20rusersd K65identd K95kudzu    S50xinetd   S85httpd
K01pppoe     K20rwalld  K75netfs  K96pcmcia   S55named    S90cron
K03rhnsd    K20rwhod   K83ypbind S08ipchains S55sshd     S91atalk
K05keytable K34yppasswdd K84apmd  S10network  S56rawdevices S95anacron
K10xfs      K45arpwatch K84ypserv S12syslog   S65dhcpd    S99local
K20nfs      K50snmpd   K86nfslock S20random   S80sendmail S99webmin
```

Files that are prefaced with a K are files used to safely shut down processes, whereas files with the S prefix are used to start files. The number preceding the service name determines the order in which services are started or stopped. To completely remove a service from the runlevel, just delete the links from the runlevel directory.

chkconfig

If you're more comfortable with a command-line tool, you can use `chkconfig` to show the what's in your current runlevel and to delete and add services. Additionally, `chkconfig` will also show the `xinetd` services enabled on your system. For example, to show the runlevel settings, type `chkconfig --list`:

```
[jray@bcdinc rc.d]$ chkconfig --list
anacron    0:off 1:off 2:on 3:on 4:on 5:on 6:off
httpd      0:off 1:off 2:off 3:on 4:on 5:on 6:off
apmd       0:off 1:off 2:on 3:off 4:on 5:on 6:off
syslog     0:off 1:off 2:on 3:on 4:on 5:on 6:off
cron       0:off 1:off 2:on 3:on 4:on 5:on 6:off
netfs      0:off 1:off 2:off 3:off 4:on 5:on 6:off
...
sendmail   0:off 1:off 2:on 3:on 4:on 5:on 6:off
snmpd     0:off 1:off 2:off 3:off 4:off 5:off 6:off
rhnsd     0:off 1:off 2:off 3:off 4:on 5:on 6:off
xinetd    0:off 1:off 2:off 3:on 4:on 5:on 6:off
ypbind     0:off 1:off 2:off 3:off 4:off 5:off 6:off
yppasswdd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
ypserv     0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

```

dhcpd      0:off 1:off 2:off 3:on 4:off 5:off 6:off
webmin     0:off 1:off 2:on 3:on 4:off 5:on 6:off
atalk      0:off 1:off 2:off 3:on 4:on 5:on 6:off
xinetd based services:
  imap:    on
  imaps:   off
  ipop2:   off
  ipop3:   on
  pop3s:   off

```

To delete a service from the runlevel or `xinetd`, use `chkconfig --del <service name>`. Similarly, you can add a service with `chkconfig --add <service name>`. This provides a clean interface to service management and is less prone to introducing errors into the system than editing the configuration files by hand.

Depending on your system, you might also have access to `ntsysv`, a Curses-based service editor. Users of KDE and GNOME also have runlevel editors built into their desktop systems. Personally, I find it easy to edit `xinetd/inetd` and runlevels manually.

If you plan to connect your machine to a network immediately upon installation, I *highly* recommend that you disable as many services as necessary before reading beyond this chapter. I've seen instances in which machines on open Internet connections have been hacked in less than an hour because they were not properly secured. Take your server's security seriously—it would be a shame if a hacker got to use your computer before you!

Boot Loaders

Boot loaders are small programs that manage the boot process. If you've worked with Windows NT, you've had some experience with a boot loader. At startup, NT's boot loader asks what operating system you'd like to boot to.

In Linux, the most commonly used boot-loading tool is LILO, the Linux Loader. During installation (typically at the very end), Linux will generate LILO values and ask you to verify them. At that time, you are given the opportunity to insert additional LILO boot options. For example, perhaps you have additional partitions and operating systems you'd like to add. This way, during system startup you can choose which operating system to use for that session.

LILO reads its options from `/etc/lilo.conf`, the LILO configuration file. `/etc/lilo.conf` provides an option for a boot password. Let's quickly cover that now.

`/etc/lilo.conf`: The LILO Configuration File

After installation, your `/etc/lilo.conf` will contain values for boot images, target drives, and the root partition. Here's the `/etc/lilo.conf` from the drive partitioned in the preceding example:

```

#
# general section
#
boot = /dev/hda
install = /boot/boot.b
message = /boot/message
prompt

```

```
# wait 20 seconds (200 10ths) for user to select the entry to load
timeout = 200

#
# default entry
#

image = /vmlinuz
  label = linux
  root = /dev/hda2
  read-only

#
# additional entries
#
```

Let's quickly familiarize you with `/etc/lilo.conf` and its contents. This way, when you edit it, you'll feel confident that you're making the right changes. Table 3.3 lists some commonly used options for `/etc/lilo.conf`.

Table 3.3 Commonly Used `/etc/lilo.conf` Options

Option	Purpose
<code>append=[hardware-params]</code>	Use this option to specify additional hardware parameters. For example, you might want to specify the amount of RAM you have or your hard drive's precise geometry, which might not necessarily be auto-detected.
<code>backup=[backup-file]</code>	Use this option to prompt LILO to copy the boot sector to a backup file.
<code>boot=[boot-]</code>	Use this option to specify the bootable partition. For example, in the sample <code>/etc/lilo.conf</code> , the boot device is <code>/dev/hda</code> (the first hard drive).
<code>delay=[time]</code>	Use this option to specify how long the boot loader should pause before booting, in tenths of a second. This is Linux's equivalent of Windows NT's <code>STARTUP/SHUTDOWN</code> pause setting. You can narrow this to nothing unless you intend to pass additional parameters at the <code>boot:</code> prompt.
<code>force-backup=[file]</code>	Use this option to back up the boot sector to a file and overwrite previous backups.
<code>install=[boot-sector]</code>	Use this option to install the specified file as the new boot sector. This is generally not required unless you want to specify a boot sector other than the default (<code>/boot/boot.b</code>).
<code>initrd=[ramdisk]</code>	Some Linux users need to have a ramdisk

<code>image]</code>	containing drivers for their hardware (such as SCSI cards); this entry will be created automatically upon installation. It is <i>very</i> important that it not be removed if it exists.
<code>message=[message-file]</code>	Use this option to specify a <code>message</code> file, which contains the text message that appears above the <code>boot:</code> prompt at boot time. Usually, this is a note from the vendor or a message demanding additional boot arguments. However, you can make this anything you like. (I've seen some pretty goofy ones.)
<code>password=[password]</code>	Use this option to set a boot password. We'll cover this in just a moment.
<code>restricted</code>	Use this option to specify that a password is required only when users attempt to pass additional boot arguments.
<code>timeout=[time]</code>	Use this option to specify how many tenths of a second the boot loader should wait before booting without keyboard input.
<code>verbose=[level]</code>	Use this option to control how verbose boot messages are. I recommend the maximum, which is 5.

Adding a Boot Password

To add a password to your `/etc/lilo.conf`, insert a line like this:

```
password=123456
```

This will prevent local users from booting Linux without a password. *Note that the password will not be encrypted.* Therefore, ensure that `/etc/lilo.conf` is owned by root and set to mode 600. If you don't, malicious users can later obtain your LILO password.

If you intend to automate reboots as part of some administrative procedure, you'll have to pass on the LILO `PASSWORD` option. If you do enable the `PASSWORD` option, Linux will arrest the reboot until an operator enters a password.

Summary of Boot Loaders

You might later decide not to use LILO. After all, it's not the only boot manager out there. Consult your boot loader documentation to see whether it also supports password protection. Every layer counts.

And finally, note that the `/etc/lilo.conf` `password` option does not prevent attackers from booting with a floppy. If your BIOS/PROM offers an option to disable floppy diskette boots, use it.

Another option is to install LILO to floppy. This way, attackers can't boot Linux from the hard drive

unless they have a boot disk and can guess your disk layout. If you take this approach, be sure to make several copies of your LILO boot disk, just in case your original gets corrupted.

Summary

Try to tailor your installation to meet your Linux server's essential needs, and discard the rest. There is no prescribed set of rules for this. Ascertaining those needs is an undertaking that demands skill, organization, and clear goals. Particularly when you're employing Linux in enterprise environments, you should outline how the server will be used, who will use it, and what data it will serve.

The next chapter departs from preliminary security measures (physical security, installation, and so on) in favor of old-fashioned system administration.

© Copyright Pearson Education. All rights reserved.