Red Hat Linux 7.2

The Official Red Hat Linux Reference Guide

Contents

Red Hat Linux 7.2

viii

Introduction

Welcome to the *Official Red Hat Linux Reference Guide*.

The *Official Red Hat Linux Reference Guide* contains useful information about your Red Hat Linux system. From fundamental concepts, such as the structure of the Red Hat Linux filesystem, to the finer points of system security and authentication control, we hope you will find this book to be a valuable resource.

This guide is for you if you want to learn a bit more about how your Red Hat Linux system works. Topics that you will explore include the following:

- *Booting Red Hat Linux* — Information about runlevels, `rc.d` directories, and how to start your favorite applications at boot time.

- *The /proc filesystem* — Provides you with the kernel's view of the system.

- *The GRUB boot loader* — A behind the scenes look at GRUB and how it boots your system.

- *System and network security* — Find out the most common methods used by attackers to compromise your system and how to prevent security problems.

- *Apache directives* — A look at the default Apache configuration and the options available.

Before reading this guide, you should be familiar with the contents of the *Official Red Hat Linux Installation Guide* concerning installation issues, the *Official Red Hat Linux Getting Started Guide* for basic Linux concepts and the *Official Red Hat Linux Customization Guide* for general customization instructions. The *Official Red Hat Linux Reference Guide* contains information about advanced topics that may not affect every user, depending upon how they use their Red Hat Linux system.

HTML and PDF versions of all Official Red Hat Linux manuals are available online at http://www.redhat.com/support/manuals.

Finding Appropriate Documentation

You need documentation that is appropriate to your level of Linux expertise. Otherwise, you might feel overwhelmed or not find the necessary information to answer your questions. The *Official Red Hat Linux Reference Guide* deals with the more technical aspects and options of your Red Hat Linux system. This section will help you decide whether to look in this manual for the information you need or consider other Red Hat Linux manuals, including online sources, in your search.

Three different categories of people use Red Hat Linux, and each of these categories require different sets of documentation and informatative sources. To help you figure out where you should start, determine your own experience level:

New to Linux

This type of user has never used any Linux (or Linux-like) operating system before or has had only limited exposure to Linux. They may or may not have experience using other operating systems (such as Windows). Is this you? If so, skip ahead to *Documentation For First-Time Linux Users*.

Some Linux Experience

This type of user has installed and successfully used Linux (but not Red Hat Linux) before or may have equivalent experience with other Linux-like operating systems. Does this describe you? If so, turn to *For the More Experienced.*

Experienced User

This type of user has installed and successfully used Red Hat Linux before. If this describes you, turn to *Documentation for Linux Gurus.*

Documentation For First-Time Linux Users

For someone new to Linux, the amount of information available on any particular subject, such as printing, starting up the system or partitioning your hard drive, can be overwhelming. It helps to initially step back and gain a decent base of information centered around how Linux works before tackling these kinds of advanced issues.

Your first goal should be to obtain some useful documentation. This cannot be stressed enough. Without documentation, you will only become frustrated at your inability to get your Red Hat Linux system working the way you want.

You should acquire the following types of Linux documentation:

*   *A brief history of Linux* — Many aspects of Linux are the way they are because of historical precedent. The Linux culture is also based on past events, needs or requirements. A basic understanding of the history of Linux will help you figure out how to solve many potential problems before you actually see them.

*   *An explanation of how Linux works* — While delving into the most arcane aspects of the Linux kernel is not necessary, it is a good idea to know something about how Linux is put together. This is particularly important if you have been working with other operating systems, as some of the assumptions you currently hold about how computers work may not transfer from that operating system to Linux.

*   *An introductory command overview (with examples)* — This is probably the most important thing to look for in Linux documentation. The underlying design philosophy for Linux is that it's better to use many small commands connected together in different ways than it is to have a few large (and complex) commands that do the whole job themselves. Without examples that illustrate this

approach to doing things, you may find yourself intimidated by the sheer number of commands available on your Red Hat Linux system.

Keep in mind that you do not have to memorize all of the available Linux commands. Different techniques exist to help you find the specific command you need to accomplish a task. You need only know the general way in which Linux functions, what you need to accomplish, and how to access the tool that will give you the exact instructions you need to execute the command.

The *Official Red Hat Linux Installation Guide* is a excellent reference for helping you get your Red Hat Linux system successfully installed and initially configured. The *Official Red Hat Linux Getting Started Guide* covers the history of Linux, basic system commands, GNOME, KDE, RPM, and many other fundamental concepts. You should start with these two books and use them to build the base of your Red Hat Linux knowledge. Before long, more complicated concepts will begin to make sense because you already grasp the general ideas.

Beyond reading Red Hat Linux manuals, several other excellent documentation resources are available for little or no cost:

Introduction to Linux Websites

•   http://www.redhat.com — On the Red Hat website, you will find links to the Linux Documentation Project (LDP), online versions of the Red Hat Linux manuals, FAQs (Frequently Asked Questions), a database which can help you find a Linux Users Group near you, technical information in the Red Hat Support Knowledge Base, and more.

•   http://www.linuxheadquarters.com — The Linux Headquarters website features easy to follow, step-by-step guides for a variety of Linux tasks.

Introduction to Linux Newsgroups

You can participate in newsgroups by watching the discussions of others attempting to solve problems, or by actively asking or answering questions. Experienced Linux users are known to be extremely helpful when trying to assist new users with various Linux issues — especially if you are posing your questions in the right venue. If you do not have access to a news reader application, you can access this information via the web at http://www.deja.com. Dozens of Linux-related newsgroups exist, including the following:

•   linux.help — A great place to get help from fellow Linux users.

•   linux.redhat — This newsgroup primarily covers Red Hat Linux-specific issues.

•   linux.redhat.install — Pose installation questions to this newsgroup or search it to see how others solved similar problems.

•   linux.redhat.misc — Questions or requests for help that do not really fit into traditional categories go here.

- linux.redhat.rpm — A good place to go if you are having trouble using RPM to accomplish particular objectives.

Beginning Linux Books

- *Red Hat Linux for Dummies, 2nd Edition* by Jon "maddog" Hall; IDG

- *Special Edition Using Red Hat Linux* by Alan Simpson, John Ray and Neal Jamison; Que

- *Running Linux* by Matt Welsh and Lar Kaufman; O'Reilly & Associates

- *Red Hat Linux 7 Unleashed* by William Ball and David Pitts; Sams

The books suggested here are excellent primary sources of information for basic knowledge about a Red Hat Linux system. For more in-depth information concerning the various topics discussed throughout this book, many of the chapters list specific book titles, usually in an *Additional Resources* area.

For the More Experienced

If you have used other Linux distributions, you probably already have a basic grasp of the most frequently used commands. You may have installed your own Linux system, and maybe you have even downloaded and built software you found on the Internet. After installing Linux, however, configuration issues can be very confusing.

The *Official Red Hat Linux Customization Guide* is designed to help explain the various ways your Red Hat Linux system can be configured to meet specific objectives. Use this manual to learn about specific configuration options and how to put them into effect.

When you are installing software that is not covered in the *Official Red Hat Linux Customization Guide*, it is often helpful to see what other people in similar circumstances have done. HOWTO documents from the Linux Documentation Project, available at http://www.redhat.com/mirrors/LDP/HOWTO/HOWTO-INDEX/howtos.html, document particular aspects of Linux, from low-level kernel esoteric changes to using Linux for amateur radio station work.

Documentation for Linux Gurus

If you are a long-time Red Hat Linux user, you probably already know that one of the best ways to understand a particular program is to read its source code and/or configuration files. A major advantage of Red Hat Linux is the availability of the source code for anyone to read.

Obviously, not everyone is a C programmer, so the source code may not be helpful for you. However, if you have the knowledge and skills necessary to read it, the source code holds all of the answers.

Document Conventions

When you read this manual, you will see that certain words are represented in different fonts, type-faces, sizes and weights. This highlighting is systematic; different words are represented in the same style to indicate their inclusion in a specific category. The types of words that are represented this way include the following:

`command`

> Linux commands (and other operating system commands, when used) are represented this way. This style should indicate to you that you can type in the word or phrase on the command line and press [Enter] to invoke a command. Sometimes a command contains words that would be displayed in a different style on their own (e.g., filenames). In these cases, they are considered to be part of the command, so the entire phrase will be displayed as a command. For example:
>
> Use the `cat testfile` command to view the contents of a file, named `testfile`, in the current working directory.

`filename`

> Filenames, directory names, paths and RPM package names are represented this way. This style should indicate that a particular file or directory exists by that name on your Red Hat Linux system. Examples:
>
> The `.bashrc` file in your home directory contains bash shell definitions and aliases for your own use.
>
> The `/etc/fstab` file contains information about different system devices and filesystems.
>
> The `/usr/share/doc` directory contains documentation for various programs.
>
> Install the `webalizer` RPM if you want to use a Web server log file analysis program.

application

> This style should indicate to you that the program named is an end-user application (as opposed to system software). For example:
>
> Use Netscape Navigator to browse the Web.

[key]

> A key on the keyboard is shown in this style. For example:
>
> To use [Tab] completion, type in a character and then press the [Tab] key. Your terminal will display the list of files in the directory that start with that letter.

[key]-[combination]

> A combination of keystrokes is represented in this way. For example:

The [Ctrl]-[Alt]-[Backspace] key combination will restart the X Window System.

text found on a GUI interface

A title, word or phrase found on a GUI interface screen or window will be shown in this style. When you see text shown in this style, it is being used to identify a particular GUI screen or an element on a GUI screen (e.g., text associated with a checkbox or field). Examples:

On the GNOME **Control Center** screen, you can customize your GNOME window manager.

Select the **Require Password** checkbox if you would like your screensaver to require a password before stopping.

top level of a menu on a GUI screen or window

When you see a word in this style, it indicates that the word is the top level of a pulldown menu. If you click on the word on the GUI screen, the rest of the menu should appear. For example:

Under **Settings** on a GNOME terminal, you will see the following menu items: **Preferences**, **Reset Terminal**, **Reset and Clear**, and **Color selector**.

If you need to type in a sequence of commands from a GUI menu, they will be shown like the following example:

Click on **Programs**=>**Applications**=>**Emacs** to start the Emacs text editor.

button on a GUI screen or window

This style indicates that the text will be found on a clickable button on a GUI screen. For example:

Click on the **Back** button to return to the Web page you last viewed.

computer output

When you see text in this style, it indicates text displayed by the computer on the command line. You will see responses to commands you typed in, error messages and interactive prompts for your input during scripts or programs shown this way. For example:

Use the `ls` to display the contents of a directory:

```
$ ls
Desktop              axhome          logs        paulwesterberg.gif
Mail                 backupfiles     mail        reports
```

The output returned in response to the command (in this case, the contents of the directory) is shown in this style.

prompt

A prompt, which is a computer's way of signifying that it is ready for you to input something, will be shown in this style. Examples:

```
$
```

```
#
```

```
[stephen@maturin stephen]$
```

```
leopard login:
```

user input

> Text that the user has to type, either on the command line, or into a text box on a GUI screen, is displayed in this style. In the following example, **text** is displayed in this style:
>
> To boot your system into the text based installation program, you will need to type in the **text** command at the `boot:` prompt.
>
> Another example, with the word **root** displayed as something the user needs to type in:
>
> If you need to log in as root when you first log into your system, and you are using the graphical login screen, at the `Login` prompt, type **root**. At the `Password` prompt, type in the root password.

glossary entry

> A word that appears in the glossary will be shown in the body of the document in this style. For example:
>
> The lpd **daemon** handles printing requests.
>
> In this case, the style of the word **daemon** should indicate to you that a definition of the term is available in the glossary.

Additionally, we use several different strategies to draw your attention to certain pieces of information. In order of how critical the information is to your system, these items will be marked as a note, a caution or a warning. For example:

Note

Remember that Linux is case sensitive. In other words, a rose is not a ROSE is not a rOsE.

CAUTION

Do not do routine tasks as root — use a regular user account unless you need
to use the root account to administer your system.

WARNING

**If you choose not to partition manually, a server installation will remove
all existing partitions on all installed hard drives. Do not choose this
installation class unless you are sure you have no data you need to save.**

Using the Mouse

Red Hat Linux is designed to use a three-button mouse. If you have a two-button mouse, you should
have selected three-button emulation during the installation process. If you're using three-button em-
ulation, pressing both mouse buttons at the same time equates to pressing the missing third (middle)
button.

In this document, if you are instructed to click with the mouse on something, that means click the left
mouse button. If you need to use the middle or right mouse button, that will be explicitly stated. (This
will be reversed if you've configured your mouse to be used by a left handed person.)

The phrase "drag and drop" may be familiar to you. If you're instructed to drag and drop an item on
your GUI desktop, click on something and hold the mouse button down. While continuing to hold
down the mouse button, drag the item by moving the mouse to a new location. When you've reached
the desired location, release the mouse button to drop the item.

Copying and Pasting Text With X

Copying and pasting text is easy using your mouse and the X Window System. To copy text, simply
click and drag your mouse over the text to highlight it. To paste the text somewhere, click the middle
mouse button in the spot where the text should be placed.

More to Come

The *Official Red Hat Linux Reference Guide* is part of Red Hat's commitment to provide useful and
timely support to Red Hat Linux users. Future editions will feature expanded information on changes

to system structure and organization, new and powerful security tools, and other resources to help you extend the power of your Red Hat Linux system — and your ability to use it.

That's where you can help.

We Need Feedback!

If you find an error in the *Official Red Hat Linux Reference Guide*, or if you have thought of a way to make this manual better, we'd love to hear from you! Please submit a report in Bugzilla (http://bugzilla.redhat.com/bugzilla) against the component *rhl-rg*.

Be sure to mention the manual's identifier:

```
rhl-rg(EN)-7.2-Print-RHI (2001-08-30T12:53-0400)
```

If you mention the manual's identifier, we will know exactly which version of the guide you have.

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Sign Up for Support

If you have an official edition of Red Hat Linux 7.2, please remember to sign up for the benefits you are entitled to as a Red Hat customer.

You will be entitled to any or all of the following benefits, depending upon the Official Red Hat Linux product you purchased:

- Official Red Hat support — Get help with your installation questions from Red Hat, Inc.'s support team.
- Red Hat Network — Easily update your packages and receive security notices that are customized for your system. Go to http://rhn.redhat.com for more details.
- *Under the Brim: The Official Red Hat E-Newsletter* — Every month, get the latest news and product information directly from Red Hat.

To sign up, go to http://www.redhat.com/apps/activate/. You will find your Product ID on a black, red, and white card in your Official Red Hat Linux box.

To read more about technical support for Official Red Hat Linux, refer to the *Getting Technical Support* Appendix in the *Official Red Hat Linux Installation Guide*.

Good luck, and thank you for choosing Red Hat Linux!

The Red Hat Documentation Team

Part I   System Reference

1   Filesystem Structure

1.1 Why Share a Common Structure?

An operating system's filesystem structure is its most basic level of organization. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent upon the way it stores its files on a primary storage device (normally a hard disk drive). It is crucial for a variety of reasons that users, as well as programs at the time of installation and beyond, be able to refer to a common guideline to know where to read and write their binary, configuration, log, and other necessary files.

A filesystem can be seen in terms of two different logical categories of files:

*   Shareable vs. unshareable files

*   Variable vs. static files

Shareable files are those that can be accessed by various hosts; **unshareable** files are not available to any other hosts. **Variable** files can change at any time without system administrator intervention (whether active or passive); **static** files, such as documentation and binaries, do not change without an action from the system administrator or an agent that the system administrator has placed in motion to accomplish that task.

The reason for looking at files in this way has to do with the type of permissions given to the directory that holds them. The way in which the operating system and its users need to utilize the files determines the directory where those files should be placed, whether the directory is mounted read-only or read-write, and the level of access allowed on each file. The top level of this organization is crucial, as the access to the underlying directories can be restricted or security problems may manifest themselves if the top level is left disorganized or without a widely-utilized structure.

However, simply having a structure does not mean very much unless it is a standard. Competing structures can actually cause more problems than they fix. Because of this, Red Hat has chosen the the most widely-used filesystem structure and extended it only slightly to accommodate special files used within Red Hat Linux.

1.2 Overview of Filesystem Hierarchy Standard (FHS)

Red Hat is committed to the **Filesystem Hierarchy Standard** (**FHS**), a collaborative document that defines the names and locations of many files and directories. We will continue to track and follow the standard to keep Red Hat Linux FHS-compliant.

The current FHS document is the authoritative reference to any FHS-compliant filesystem, but the standard leaves many areas undefined or extensible. In this section, we provide an overview of the standard and a description of the parts of the filesystem not covered by the standard.

The complete standard is available at:

 http://www.pathname.com/fhs

Compliance with the standard means many things, but the two most important are compatibility with other compliant systems and the ability to mount the /usr partition as read-only (because it contains common executables and is not meant to be changed by users). Since /usr can be mounted read-only, /usr can be mounted from the CD-ROM or from another machine via read-only NFS.

1.2.1 FHS Organization

The directories and files noted here are a small subset of those specified by the FHS document. Check the latest FHS document for the most complete information.

The `/dev` Directory

The /dev directory contains filesystem entries which represent devices that are attached to the system. These files are essential for the system to function properly.

The `/etc` Directory

The /etc directory is reserved for configuration files that are local to your machine. No binaries are to be put in /etc. Any binaries that were formerly put in /etc should now go into /sbin or possibly /bin.

The X11 and skel directories should be subdirectories of /etc:

```
/etc
  |- X11
  |- skel
```

The X11 directory is for X11 configuration files such as XF86Config. The skel directory is for "skeleton" user files, which are used to populate a home directory when a user is first created.

The `/lib` Directory

The /lib directory should contain only those libraries that are needed to execute the binaries in /bin and /sbin. These shared library images are particularly important for booting the system and executing commands within the root filesystem.

The `/mnt` Directory

The /mnt directory refers to temporarily mounted filesystems, such as CD-ROMs and floppy disks.

The `/opt` Directory

The `/opt` directory provides an area for usually large, static application software packages to be stored.

For packages that wish to avoid putting their files throughout the filesystem, `/opt` provides a logical and predictable organizational system under that package's directory. This gives the system administrator an easy way to determine the role of each file within a particular package.

For example, if `sample` is the name of a particular software package located within `/opt`, then all of its files could be placed within directories inside `/opt/sample`, such as `/opt/sample/bin` for binaries and `/opt/sample/man` for manual pages.

Large packages that encompass many different sub-packages, each of which accomplish a particular task, also go within `/opt`, giving that large package a standardized way to organize itself. In this way, our `sample` package may have different tools that each go in their own sub-directories, such as `/opt/sample/tool1` and `/opt/sample/tool2`, each of which can have their own `bin`, `man`, and other similar directories.

The `/proc` Directory

The `/proc` directory contains special "files" that either extract information from or send information to the kernel.

Due to the great variety of data available within `/proc` and the many ways this directory can be used to communicate with the kernel, an entire chapter has been devoted to the subject. For more information, please see Chapter 4, *The `/proc` Filesystem*.

The `/sbin` Directory

The `/sbin` directory is for executables used only by the root user. The executables in `/sbin` are only used to boot and mount `/usr` and perform system recovery operations. The FHS says:

> "`/sbin` typically contains files essential for booting the system in addition to the binaries in `/bin`. Anything executed after `/usr` is known to be mounted (when there are no problems) should be placed in `/usr/sbin`. Local-only system administration binaries should be placed into `/usr/local/sbin`."

At a minimum, the following programs should be in `/sbin`:

```
arp, clock,
getty, halt,
init, fdisk,
fsck.*, grub,
ifconfig, lilo,
mkfs.*, mkswap,
```

```
reboot, route,
shutdown, swapoff,
swapon, update
```

The `/usr` Directory

The `/usr` directory is for files that can be shared across a whole site. The `/usr` directory usually has its own partition, and it should be mountable read-only. The following directories should be subdirectories of `/usr`:

```
/usr
    |- bin
    |- dict
    |- etc
    |- games
    |- include
    |- kerberos
    |- lib
    |- libexec
    |- local
    |- sbin
    |- share
    |- src
    |- X11R6
```

The `bin` directory contains executables, `doc` contains non-FHS compliant documentation pages, `etc` contains system-wide configuration files, `games` is for games, `include` contains C header files, `kerberos` contains binaries and much more for Kerberos, and `lib` contains object files and libraries that are not designed to be directly utilized by users or shell scripts. The `libexec` directory contains small helper programs called by other programs, `sbin` is for system administration binaries (those that do not belong in `/sbin`), `share` contains files that are not architecture-specific, `src` is for source code, and `X11R6` is for the X Window System (XFree86 on Red Hat Linux).

The `/usr/local` Directory

The FHS says:

> "The `/usr/local` hierarchy is for use by the system administrator when installing software locally. It needs to be safe from being overwritten when the system software is updated. It may be used for programs and data that are shareable amongst a group of hosts, but not found in `/usr`."

The `/usr/local` directory is similar in structure to the `/usr` directory. It has the following subdirectories, which are similar in purpose to those in the `/usr` directory:

```
/usr/local
```

```
                    |- bin
                    |- doc
                    |- etc
                    |- games
                    |- include
                    |- lib
                    |- libexec
                    |- sbin
                    |- share
                    |- src
```

The `/var` Directory

Since the FHS requires that you be able to mount `/usr` read-only, any programs that write log files or need `spool` or `lock` directories should probably write them to the `/var` directory. The FHS states `/var` is for:

> "…variable data files. This includes spool directories and files, administrative and logging data, and transient and temporary files."

The following directories should be subdirectories of `/var`:

```
/var
   |- arpwatrch
   |- cache
   |- db
   |- ftp
   |- gdm
   |- kerberos
   |- lib
   |- local
   |- lock
   |- log
   |- named
   |- nis
   |- opt
   |- preserve
   |- run
   +- spool
        |- anacron
        |- at
        |- cron
        |- fax
        |- lpd
        |- mail
        |- mqueue
```

```
                |- news
                |- rwho
                |- samba
                |- slrnpull
                |- squid
                |- up2date
                |- uucp
                |- uucppublic
                |- vbox
                |- voice
        |- tmp
        |- www
        |- yp
```

System log files such as messages and lastlog go in /var/log. The /var/lib/rpm direc-
tory also contains the RPM system databases. Lock files go in /var/lock, usually in directories
particular for the program using the file. The /var/spool directory has subdirectories for various
systems that need to store data files.

1.2.2 `/usr/local` in Red Hat Linux

In Red Hat Linux, the intended use for /usr/local is slightly different from that specified by the
FHS. The FHS says that /usr/local should be where you store software that is to remain safe
from system software upgrades. Since system upgrades from Red Hat are done safely with the RPM
and Gnome-RPM, you do not need to protect files by putting them in /usr/local. Instead, we
recommend you use /usr/local for software that is local to your machine.

For instance, say you have mounted /usr via read-only NFS from a host named jake. If there
is a package or program you would like to install, but you are not allowed to write to jake, you
should install it under /usr/local. Later perhaps, if you have managed to convince the system
administrator of jake to install the program on /usr, you can uninstall it from /usr/local.

1.3 Special Red Hat Linux File Locations

In addition to the files pertaining to RPM that reside in /var/lib/rpm (see the RPM chapter in
the *Official Red Hat Linux Customization Guide* for more information on RPM), there are two other
special locations reserved for Red Hat Linux configuration and operation.

The configuration tools provided with Red Hat Linux install many scripts, bitmap, and text files in
/usr/lib/rhs. Since these files are generated by software on your system, you probably will not
want to edit any of them by hand.

The other special location (/etc/sysconfig) stores configuration information. Many scripts that
run at boot time use the files in this directory.

2 Users and Groups

The control of **users** and **groups** exists at the core of Red Hat Linux system administration.

Users can be either people (accounts tied to a physical user) or logical users (accounts that exist for applications so that they can perform specific tasks) . Both types of users have a **User ID** (which is usually unique) and **Group ID**.

Groups are logical expressions of organization. Groups form the foundation of tying users together and giving them permissions to read, write, or execute a given file.

Any file created is assigned a user and group when it is made. It is also assigned separate read, write, and execute permissions for the file's owner, the group assigned to the file, and any other users on that host. The user and group of a particular file, as well as the permissions on that file, can be changed by root or, to a lesser extent, by the creator of the file.

Proper management of users and groups, as well as assigning and revoking permissions, is one of the most important tasks of any system administrator.

2.1 Tools for User and Group Administration

Managing users and groups has traditionally been tedious, but Red Hat Linux provides a few tools and conventions to make users and groups easier to manage.

While you can use `useradd` to create a new user from the shell prompt, a popular way to manage users and groups is through redhat-config-users (see the *Official Red Hat Linux Customization Guide* for more information).

2.2 Standard Users

In Table 2–1, *Standard Users*, you will find the standard users set up by the installation process (as seen in the `/etc/passwd` file). The Group ID (GID) in this table is the *primary group* for the user. See Section 2.4, *User Private Groups* for details on how groups are used.

Table 2–1   Standard Users

User	UID	GID	Home Directory	Shell
root	0	0	/root	/bin/bash
bin	1	1	/bin	
daemon	2	2	/sbin	
adm	3	4	/var/adm	

User	UID	GID	Home Directory	Shell
lp	4	7	`/var/spool/lpd`	
sync	5	0	`/sbin`	`/bin/sync`
shutdown	6	0	`/sbin`	`/sbin/shutdown`
halt	7	0	`/sbin`	`/sbin/halt`
mail	8	12	`/var/spool/mail`	
news	9	13	`/var/spool/news`	
uucp	10	14	`/var/spool/uucp`	
operator	11	0	`/root`	
games	12	100	`/usr/games`	
gopher	13	30	`/usr/lib/gopher-` `data`	
ftp	14	50	`/var/ftp`	
nobody	99	99	`/`	
apache	48	48	`/var/www`	
named	25	25	`/var/named`	
piranha	60	60	`/etc/sysconfig/ha`	
amanda	33	6	`var/lib/amanda/`	
ident	98	98	`/`	`/sbin/nologin`
rpc	32	32	`/`	
wnn	49	49	`/var/lib/wnn`	
xfs	43	43	`/etc/X11/fs`	
mailnull	47	47	`/var/spool/mqueue`	
pvm	24	24	`/usr/share/pvm3`	`/bin/bash`
ldap	55	55	`/var/lib/ldap`	
mysql	27	27	`/var/lib/mysql`	

User	UID	GID	Home Directory	Shell
rpcuser	29	29	`/var/lib/nfs`	
junkbust	73	73	`/etc/junkbuster`	
gdm	42	42	`/var/gdm`	
squid	23	23	`/var/spool/squid`	`/dev/null`
nscd	28	28	`/`	
rpm	37	37	`/var/lib/rpm`	
mailman	41	41	`/var/mailman`	
radvd	75	75	`/`	
postgres	26	26	`/var/lib/pgsql`	

2.3 Standard Groups

In Table 2–2, *Standard Groups*, you will find the standard groups as set up by the installation process (as seen in the `/etc/group` file).

Table 2–2   Standard Groups

Group	GID	Members
root	0	root
bin	1	root, bin, daemon
daemon	2	root, bin, daemon
sys	3	root, bin, adm
adm	4	root, adm, daemon
tty	5	
disk	6	root
lp	7	daemon, lp
mem	8	
kmem	9	

Group	GID	Members
wheel	10	root
mail	12	mail
news	13	news
uucp	14	uucp
man	15	
games	20	
gopher	30	
dip	40	
ftp	50	
nobody	99	
users	100	
piranha	60	piranha
ident	98	ident
rpc	32	rpc
pvm	24	pvm
floppy	19	
utmp	22	
slocate	21	
pppusers	44	
popusers	45	
slipusers	46	
rpm	37	rpm
postgres	26	postgres
nscd	28	nscd
wnn	49	wnn

Group	GID	Members
mailnull	47	mailnull
rpcusers	29	
xfs	43	xfs
gdm	42	gdm
apache	48	apache
squid	23	squid
named	25	named
junkbust	73	junkbust
mysql	27	mysql
mailman	41	mailman
ldap	55	ldap

2.4 User Private Groups

Red Hat Linux uses a **user private group** (**UPG**) scheme, which makes UNIX groups easier to use. The UPG scheme does not add or change anything in the standard UNIX way of handling groups; it simply offers a new convention. Whenever you create a new user, by default, they have a unique group. The scheme works as follows:

User Private Group

Every user has a primary group; the user is the only member of that group.

umask = 002

Traditionally, on UNIX systems the umask is 022, which prevents other users *and other members of a user's primary group* from modifying a user's files. Since every user has their own private group in the UPG scheme, this "group protection" is not needed. A umask of 002 will prevent users from modifying other users' private files. The umask is set in /etc/profile.

setgid bit on Directories

If you set the setgid bit on a directory (with chmod g+s *directory*), files created in that directory will have their group set to the directory's group.

Many IT organizations like to create a group for each major project and then assign people to the group if they need to access that group's files. Using this traditional scheme, managing files has been

difficult because when someone creates a file, it is associated with the primary group to which they belong. When a single person works on multiple projects, it is difficult to associate the right files with the right group. Using the UPG scheme, however, groups are automatically assigned to files created within a directory with the setgid bit set, which makes managing group projects that share a common directory very simple.

For example, say you have a big project called *devel*, with many people editing the *devel* files in a devel directory. Make a group called devel, chgrp the devel directory to devel, and add all of the *devel* users to the devel group.

You can add a user to a group using redhat-config-users (see the *Official Red Hat Linux Customization Guide*), or if you prefer to use the command line, use the /usr/sbin/groupadd *group-name* command to create a group. The /usr/bin/gpasswd -a *loginname groupname* command will add a user *loginname* to a group. (See the groupadd and gpasswd man pages if you need more information on their options.) The /etc/group file contains the group information for your system.

If you created the devel group, added users to the devel group, changed the group for devel directory to the devel group, and set the setgid bit for the devel directory, all *devel* users will be able to edit the *devel* files and create new files in the devel directory. The files they create will always retain their devel group status, so other *devel* users will always be able to edit them.

If you have multiple projects like *devel* and users who are working on multiple projects, these users will never have to change their umask or group when they move from project to project. If set correctly, the setgid bit on each project's main directory "selects" the proper group for all files created in that directory.

Since each user's home directory is owned by the user and their private group, it is safe to set the setgid bit on the home directory. However, by default, files are created with the primary group of the user, so the setgid bit would be redundant.

2.4.1 User Private Group Rationale

Although UPG has existed in Red Hat Linux for quite some time, many people still have questions about it, such as why UPG is necessary. Consider the following scenario on a system without UPG:

• You would like to have a group of people work on a set of files in the /usr/lib/emacs/site-lisp directory. You trust a few people to modify the directory but certainly not everyone.

• So, first you create an emacs group:

```
/usr/sbin/groupadd emacs
```

Next, you enter:

```
chown -R root.emacs /usr/lib/emacs/site-lisp
```

to associate the contents of the directory with the emacs group and add the proper users to the group:

        /usr/bin/gpasswd -a <*username*> emacs

- To allow the users to actually create files in the directory you enter:

        chmod 775 /usr/lib/emacs/site-lisp

- But when a user creates a new file it is assigned the group of the user's default group (usually users). To prevent this you enter:

        chmod 2775 /usr/lib/emacs/site-lisp

which causes everything in the directory to be created with the emacs group.

- But the new file needs to be mode 664 for another user in the emacs group to be able to edit it. To do this you make the default umask 002.

- Well, this all works fine, except that if your default group is users, every file you create in your home directory will be writable by everybody in users (usually everyone).

- To fix this, you make each user have a "private group" as their default group.

At this point, by making the default umask 002 and giving everyone a private default group, you can easily set up groups that users can take advantage of without any extra work every time users write files to the group's common directory. Just create the group, add the users, and do the above chown and chmod on the group's directories.

2.5 Shadow Utilities

If you are in a multiuser environment and not using a networked authentication scheme such as Kerberos, you should consider using Shadow Utilities (also known as **shadow passwords**) for the enhanced protection offered for your system's authentication files. During the installation of Red Hat Linux, shadow password protection for your system is enabled by default, as are **MD5 passwords** (an alternative and arguably more secure method of encrypting passwords for storage on your system).

Shadow passwords offer a few distinct advantages over the previous standard of storing passwords on UNIX and Linux systems, including:

- Improved system security by moving the encrypted passwords (normally found in /etc/passwd) to /etc/shadow which is readable only by root

- Information concerning password aging (how long it has been since a password was last changed)

- Control over how long a password can remain unchanged before the user is required to change it

- The ability to use the /etc/login.defs file to enforce a security policy, especially concerning password aging

The `shadow-utils` package contains a number of utilities that support:

- Conversion from normal to shadow passwords and back (`pwconv`, `pwunconv`)

- Verification of the password, group, and associated shadow files (`pwck`, `grpck`)

- Industry-standard methods of adding, deleting and modifying user accounts (`useradd`, `user-mod`, and `userdel`)

- Industry-standard methods of adding, deleting, and modifying user groups (`groupadd`, `group-mod`, and `groupdel`)

- Industry-standard method of administering the `/etc/group` file using `gpasswd`

Note

There are some additional points of interest concerning these utilities:

- The utilities will work properly whether shadowing is enabled or not.

- The utilities have been slightly modified to support Red Hat's user private group scheme. For a description of the modifications, see the `user-add` man page. For more information on user private groups, turn to Section 2.4, *User Private Groups*.

- The `adduser` script has been replaced with a symbolic link to `/usr/sbin/useradd`.

- The tools in the `shadow-utils` package are not Kerberos, NIS, hesiod, or LDAP enabled. New users will be local only. For more information on Kerberos and LDAP, see Chapter 8, *Using Kerberos 5 on Red Hat Linux* and Chapter 15, *Lightweight Directory Access Protocol (LDAP)*.

---

3   Boot Process, Init, and Shutdown

This chapter contains information on what happens when you boot or shut down your Red Hat Linux system.

Note

This chapter focuses on LILO, the default boot loader for Red Hat Linux 7.1 and earlier versions. However, Red Hat Linux 7.2 contains an additional boot loader, GRUB, which is discussed more thoroughly in Chapter 5, *GRUB*. For more information concerning GRUB, see Chapter 5, *GRUB*.

3.1 Introduction

One of the most powerful aspects of Red Hat Linux concerns its open method of starting and stopping the operating system, where it loads specified programs using their particular configurations, permits you to change those configurations to control the boot process, and shuts down in a graceful and organized way.

Beyond the question of controlling of the boot or shutdown process, the open nature of Red Hat Linux makes it much easier to determine the exact source of most problems associated with starting up or shutting down your system. An understanding of this process is quite beneficial for even basic troubleshooting.

3.2 Behind the Scenes of the Boot Process

Note

This section looks at the x86 boot process, in particular. Depending on your system's architecture, your boot process may be slightly different. However, once the kernel is found and loaded by the system, the default Red Hat Linux boot process is identical across all architectures. Please see Section 3.7, *Differences in the Boot Process of Other Architectures* for more information on a non-x86 boot process.

When a computer is booted, the processor looks at the end of the system memory for the **BIOS** (Basic Input/Output System) and runs it. The BIOS program is written into read-only permanent memory

and is always available for use. The BIOS provides the lowest level interface to peripheral devices and controls the first step of the boot process.

The BIOS tests the system, looks for and checks peripherals, and then looks for a drive to use to boot the system. Usually, it checks the floppy drive (or CD-ROM drive on many newer systems) for bootable media, if present, and then it looks to the hard drive. The order of the drives used for booting is usually controlled by a particular BIOS setting on the system. Once Red Hat Linux is installed on a hard drive of a system, the BIOS looks for a **Master Boot Record** (MBR) starting at the first sector on the first hard drive, loads its contents into memory, and passes control to it.

This MBR contains instructions on how to load the boot loader, **GRUB** or**LILO** (*LI*nux *LO*ader), depending on the boot loader you installed. The MBR then loads the boot loader, which takes over the process (if the boot loader is installed in the MBR). In the default Red Hat Linux configuration, GRUB or LILO uses the settings in the MBR to display boot options and allow for user input on which operating system to actually start up.

But this begs the question: How does the boot loader in the MBR know what to do when the MBR is read? LILO actually has already written the instructions there through the use of lilo with the /etc/lilo.conf configuration file. GRUB has also written the instructions in the /boot/grub/grub.conf configuration file. For more information about GRUB, see Chapter 5, *GRUB*.

3.2.1 Options in `/etc/lilo.conf`

Most of the time, you will have no need to change the Master Boot Record on your hard drive unless you need to boot a newly installed operating system or are looking to use a new kernel. If you do need to create a new MBR using LILO but using a different configuration, you will need edit /etc/lilo.conf and run lilo again.

WARNING

If you are planning to edit `/etc/lilo.conf`, be sure to make a backup copy of the file before making any changes. Also, be sure that you have a working boot floppy available so that you will be able to boot the system and make changes to the MBR if there is a problem. See the man pages for `mkbootdisk` for more information on creating a boot disk.

The file /etc/lilo.conf is used by lilo to determine which operating system(s) to utilize or which kernel to start, as well as to know where to install itself (for example, /dev/hda for the first MBR of the IDE hard drive). A sample /etc/lilo.conf file looks like this:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
lba32
default=linux

image=/boot/vmlinuz-2.4.0-0.43.6
 label=linux
 initrd=/boot/initrd-2.4.0-0.43.6.img
 read-only
 root=/dev/hda5

other=/dev/hda1
 label=dos
```

This example shows a system configured to boot two operating systems: Red Hat Linux and DOS. Here is a deeper look at a few of the lines of this file (your /etc/lilo.conf may look a little different):

- boot=/dev/hda tells LILO to look on the first hard disk on the first IDE controller.

- map=/boot/map locates the map file. In normal use, this should not be modified.

- install=/boot/boot.b tells LILO to install the specified file as the new boot sector. In normal use, this should not be altered. If the install line is missing, LILO will assume a default of /boot/boot.b as the file to be used.

- The existence of prompt tells LILO to show you whatever is referenced in the message line. While it is not recommended that you remove the prompt line, if you do remove it, you can still get a prompt by holding down the [Shift] key while your machine starts to boot.

- timeout=50 sets the amount of time that LILO will wait for user input before proceeding with booting the default line entry. This is measured in tenths of a second, with 50 as the default.

- message=/boot/message refers to the screen that LILO displays to let you select the operating system or kernel to boot.

- lba32 describes the hard disk geometry to LILO. Another common entry here is linear. You should not change this line unless you are very aware of what you are doing. Otherwise, you could put your system in a state where it cannot boot.

- default=linux refers to the default operating system for LILO to boot from the options listed below this line. The name linux refers to the label line below in each of the boot options.

- `image=/boot/vmlinuz-2.4.0-0.43.6` specifies the linux kernel to boot with this particular boot option.

- `label=linux` names the operating system option in the LILO screen. In this case, it also is the name that is referred to by the `default` line.

- `initrd=/boot/initrd-2.4.0-0.43.6.img` refers to the **initial ram disk** image that is used at boot time to actually initialize and start the devices that makes booting the kernel possible. The initial ram disk is a collection of machine-specific drivers necessary to operate the hard drive and anything needed to load the kernel. You should never try to share initial ram disks between machines unless they are identical in their hardware configurations and kernel versions (and even then, it is a bad idea).

- `read-only` specifies that the root partition (see the `root` line below) as one that cannot be changed, only read.

- `root=/dev/hda5` tells LILO what disk partition to use as the root partition.

LILO then shows the Red Hat Linux initial screen with the different operating systems or kernels it has been configured to boot. If you only have Red Hat Linux installed and have not changed anything in /etc/lilo.conf, you will only see linux as an option. If you install SMP kernel support, you will see linux-up as an option. If you have set up LILO to boot other operating systems as well, this screen is your chance to select what operating system will boot. Use your arrow keys to highlight the operating system and press [Enter]

If you would like to have a command prompt to enter commands to LILO, press [Ctrl]-[X]. LILO displays a LILO: prompt on the screen and waits for input from the user. (The amount of time LILO waits is set by the timeout line in the /etc/lilo.conf file.) If your /etc/lilo.conf is set to give LILO a choice of operating systems, at this time you could type in the label for whichever operating system you want to boot.

If LILO is booting Linux, it first loads the kernel into memory, which is a vmlinuz file (plus a version number, for example, vmlinuz-2.4.0-xx) located in the /boot directory. Then the kernel passes control to init.

At this point, with the kernel loaded into memory and operational, Linux is already started, although at a very basic level. However, with no applications utilizing the kernel and with no ability for the user to provide meaningful input to the system, not much can be done with it. The init program solves this problem by bringing up the various services that allow the system to perform its particular role.

3.2.2 Init

The kernel finds init in /sbin and executes it, and init which coordinates the rest of the boot process.

When `init` starts, it becomes the parent or grandparent of all of the processes that start up automatically on your Red Hat Linux system. First, it runs the `/etc/rc.d/rc.sysinit` script, which sets your path, starts swapping, checks the filesystems, and so on. Basically, `rc.sysinit` takes care of everything that your system needs to have done at system initialization. Most systems use a clock, so on them `rc.sysinit` uses the `/etc/sysconfig/clock` file to initialize the clock. If you have special serial port processes that need to be initialized, `rc.sysinit` may also run `rc.serial`.

Then, `init` runs the `/etc/inittab` script, which describes how the system should be set up in each **runlevel** and sets the default runlevel. (See Section 3.4, *Init Runlevels* for more information on init runlevels.) This file states, among other things, that `/sbin/update` should be run whenever a runlevel starts. The `update` program is used to flush dirty buffers back to disk.

Whenever the runlevel changes, `init` uses the scripts in `/etc/rc.d/init.d` to start and stop various services, such as your web server, DNS server, and so on. First, `init` sets the source function library for the system (commonly `/etc/rc.d/init.d/functions`), which spells out how to start or kill a program and how to find out the PID of a program. Then, `init` determines the current and the previous runlevel.

Next, `init` starts all of the background processes necessary for the system to run by looking in the appropriate `rc` directory for that runlevel (`/etc/rc.d/rc<x>.d`, where the `<x>` is numbered 0-6). `init` runs each of the kill scripts (their file name starts with a K) with a `stop` parameter. Then, `init` runs all of the start scripts (their file names start with an S) in the appropriate runlevel directory with a `start` so that all services and applications are started correctly. In fact, you can execute these same scripts manually after the system is finished booting with a command like `/etc/rc.d/init.d/httpd stop` or `service httpd stop` logged in as root. This will stop the `httpd` server.

Note

When starting services manually, you should be root. If you get a error when executing `service httpd stop`, you may not have `/sbin` pathed in `/root/.bashrc` (or the correct `.rc` file for your preferred shell). You can either type the full command of `/sbin/service httpd stop` or add `export PATH="$PATH:/sbin"` to your shell `.rc` file. If you edit your shell configuration file, log out and back in as root to make the changed shell configuration file take effect.

None of the scripts that actually start and stop the services are located in `/etc/rc.d/rc<x>.d`. Rather, all of the files in `/etc/rc.d/rc<x>.d` are **symbolic links** that point to actual scripts located in `/etc/rc.d/init.d`. A symbolic link is nothing more than a file that simply points to another file, and they are used in this case because they can be created and deleted without affecting the actual script that kills or starts the service. The symbolic links to the various scripts are numbered

in a particular order so that they start in that order. You can change the order in which the services start up or are killed by changing the name of the symbolic link that refers to the script that actually starts or kills the service. You can give symbolic links the same number as other symbolic links if you want that service start or stop right before or after another service.

For example, for runlevel 5, init looks into the /etc/rc.d/rc5.d directory and might finds the following (your system and configuration may vary):

```
K01pppoe -> ../init.d/pppoe
K05innd -> ../init.d/innd
K10ntpd -> ../init.d/ntpd
K15httpd -> ../init.d/httpd
K15mysqld -> ../init.d/mysqld
K15pvmd -> ../init.d/pvmd
K16rarpd -> ../init.d/rarpd
K20bootparamd -> ../init.d/bootparamd
K20nfs -> ../init.d/nfs
K20rstatd -> ../init.d/rstatd
K20rusersd -> ../init.d/rusersd
K20rwalld -> ../init.d/rwalld
K20rwhod -> ../init.d/rwhod
K25squid -> ../init.d/squid
K28amd -> ../init.d/amd
K30mcserv -> ../init.d/mcserv
K34yppasswdd -> ../init.d/yppasswdd
K35dhcpd -> ../init.d/dhcpd
K35smb -> ../init.d/smb
K35vncserver -> ../init.d/vncserver
K45arpwatch -> ../init.d/arpwatch
K45named -> ../init.d/named
K50snmpd -> ../init.d/snmpd
K54pxe -> ../init.d/pxe
K55routed -> ../init.d/routed
K60mars-nwe -> ../init.d/mars-nwe
K61ldap -> ../init.d/ldap
K65kadmin -> ../init.d/kadmin
K65kprop -> ../init.d/kprop
K65krb524 -> ../init.d/krb524
K65krb5kdc -> ../init.d/krb5kdc
K75gated -> ../init.d/gated
K80nscd -> ../init.d/nscd
K84ypserv -> ../init.d/ypserv
K90ups -> ../init.d/ups
K96irda -> ../init.d/irda
S05kudzu -> ../init.d/kudzu
```

```
S06reconfig -> ../init.d/reconfig
S08ipchains -> ../init.d/ipchains
S10network -> ../init.d/network
S12syslog -> ../init.d/syslog
S13portmap -> ../init.d/portmap
S14nfslock -> ../init.d/nfslock
S18autofs -> ../init.d/autofs
S20random -> ../init.d/random
S25netfs -> ../init.d/netfs
S26apmd -> ../init.d/apmd
S35identd -> ../init.d/identd
S40atd -> ../init.d/atd
S45pcmcia -> ../init.d/pcmcia
S55sshd -> ../init.d/sshd
S56rawdevices -> ../init.d/rawdevices
S56xinetd -> ../init.d/xinetd
S60lpd -> ../init.d/lpd
S75keytable -> ../init.d/keytable
S80isdn -> ../init.d/isdn
S80sendmail -> ../init.d/sendmail
S85gpm -> ../init.d/gpm
S90canna -> ../init.d/canna
S90crond -> ../init.d/crond
S90FreeWnn -> ../init.d/FreeWnn
S90xfs -> ../init.d/xfs
S95anacron -> ../init.d/anacron
S97rhnsd -> ../init.d/rhnsd
S99local -> ../rc.local
```

These symbolic links tell init that it needs to kill pppoe, innd, ntpd, httpd, mysqld, pvmd, rarpd, bootparamd, nfs, rstatd, rusersd, rwalld, rwhod, squid, amd, mcserv, yp-passwdd, dhcpd, smb, vncserver, arpwatch, named, snmpd, pxe, routed, mars-nwe, ldap, kadmin, kprop, krb524, krb5kdc, gated, nscd, ypserv, ups, and irda. After all processes are killed, init looks into the same directory and finds start scripts for kudzu, recon-fig, ipchains, portmap, nfslock, autofs, random, netfs, apmd, identd, atd, pcm-cia, sshd, rawdevices, xinetd, lpd, keytable, isdn, sendmail, gpm, canna, crond, FreeWnn, xfs, anacron, and rhnsd, The last thing init does is run /etc/rc.d/rc.local to run any special scripts configured for that host. At this point, the system is considered to be oper-ating at runlevel 5.

After init has progressed through all of the runlevels, the /etc/inittab script forks a getty process for each virtual console (login prompts) for each runlevel (runlevels 2-5 get all six; runlevel 1, which is single user mode, only gets one console; runlevels 0 and 6 get no virtual consoles). Basi-cally, getty opens tty lines, sets their modes, prints the login prompt, gets the user's name, and then

initiates a login process for that user. This allows users to authenticate themselves to the system and begin to use it.

Also, `/etc/inittab` tells `init` how it should handle a user hitting [Ctrl]-[Alt]-[Delete] at the console. As Red Hat Linux should be properly shut down and restarted rather immediately power-cycled, `init` is told to execute the command `/sbin/shutdown -t3 -r now` when a user hits those keys. In addition, `/etc/inittab` states what `init` should do in case of power failures, if your system has a UPS unit attached to it.

In runlevel 5, `/etc/inittab` runs a script called `/etc/X11/prefdm`. The `prefdm` script runs the preferred X display manager (`gdm` if you are running GNOME or `kdm` if you are running KDE based on the contents of the `/etc/sysconfig/desktop` directory.

At this point, you should be looking at a login prompt. All that, and it only took a few seconds.

3.2.3 SysV Init

As we have seen, the `init` program is run by the kernel at boot time. It is in charge of starting all the normal processes that need to start up with the system. These include the getty processes that allow you to log in, NFS daemons, FTP daemons, and anything else you want to run when your machine boots.

SysV init is the standard init process in the Linux world to control the startup of software at boot time, because it is easier to use and more powerful and flexible than the traditional BSD `init`.

SysV init also differs from BSD init in that the configuration files are in `/etc/rc.d` instead of residing directly in `/etc`. In `/etc/rc.d`, you will find `rc`, `rc.local`, `rc.sysinit`, and the following directories:

```
init.d
rc0.d
rc1.d
rc2.d
rc3.d
rc4.d
rc5.d
rc6.d
```

SysV init represents each of the init runlevels with a separate directory, using `init` and symbolic links in each of the directories to actually stop and start the services as the system moves from runlevel to runlevel.

In summary, the chain of events for a SysV init boot is as follows:

- The kernel looks in `/sbin` for `init`
- `init` runs the `/etc/rc.d/rc.sysinit` script

- `rc.sysinit` handles most of the boot loader's processes and then runs `rc.serial` (if it exists)

- `init` runs all the scripts for the default runlevel

- `init` runs `/etc/rc.d/rc.local`

The default runlevel is decided in `/etc/inittab`. You should have a line close to the top like:

```
id:3:initdefault:
```

The default runlevel is 3 in this example, the number after the first colon. If you want to change it, you can edit `/etc/inittab` by hand. Be very careful when you are editing the `inittab` file.

If you use LILO as your boot loader, you can fix the `inittab` file by rebooting, accessing the `boot:` prompt with [Ctrl]-[X], and typing:

```
boot:  linux single
```

If you are using GRUB as your boot loader, you can fix the `inittab` file with the following steps.

- In the graphical GRUB boot loader screen, select the **Red Hat Linux** boot label and press [e] to edit it.

- Arrow down to the kernel line and press [e] to edit it.

- At the prompt, type `single` and press [Enter].

- You will be returned to the GRUB screen with the kernel information. Press the [b] key to boot the system into single user mode.

This *should* allow you to boot into single-user mode so you can re-edit `inittab` to its previous value.

Next, we will discuss information in the files within `/etc/sysconfig` that define the parameters used by different system services when they start up.

3.3 Sysconfig Information

The following information outlines some of the various files in `/etc/sysconfig`, their function, and their contents. This information is not intended to be complete, as many of these files have a variety of options that are only used in very specific or rare circumstances.

3.3.1 Files in `/etc/sysconfig`

The following files are normally found in `/etc/sysconfig`:

- `amd`

- `apmd`

- `authconfig`

- `cipe`

- `clock`

- `desktop`

- `firewall`

- `harddisks`

- `hwconf`

- `i18n`

- `init`

- `ipchains`

- `iptables`

- `irda`

- `keyboard`

- `kudzu`

- `mouse`

- `network`

- `pcmcia`

- `rawdevices`

- `sendmail`

- `soundcard`

- `ups`

- `vncservers`

It is possible that your system may be missing a few of them if the corresponding program that would need that file is not installed.

Next, we will take a look at each one.

/etc/sysconfig/amd

The `/etc/sysconfig/amd` file contains various parameters used by `amd` allowing for the auto-mounting and automatic unmounting of filesystems.

/etc/sysconfig/apmd

The /etc/sysconfig/apmd file is used by apmd as a configuration for what things to start/stop/change on suspend or resume. It is set up to turn on or off apmd during startup, depending on whether your hardware supports **Advanced Power Management** (**APM**) or if you choose not to use it. apm is a monitoring daemon that works with power management code within the Linux kernel. It can alert you to a low battery if you are using Red Hat Linux on a laptop, among other things.

/etc/sysconfig/authconfig

The /etc/sysconfig/authconfig file sets the kind of authorization to be used on the host. It contains one or more of the following lines:

- USEMD5=*<value>*, where *<value>* is one of the following:
    - yes — MD5 is used for authentication.
    - no — MD5 is not used for authentication.

- USEKERBEROS=*<value>*, where *<value>* is one of the following:
    - yes — Kerberos is used for authentication.
    - no — Kerberos is not used for authentication.

- USELDAPAUTH=*<value>*, where *<value>* is one of the following:
    - yes — LDAP is used for authentication.
    - no — LDAP is not used for authentication.

/etc/sysconfig/clock

The /etc/sysconfig/clock file controls the interpretation of values read from the system clock. Earlier releases of Red Hat Linux used the following values (which are deprecated):

- CLOCKMODE=*<value>*, where *<value>* is one of the following:
    - GMT — Indicates that the clock is set to Universal Time (Greenwich Mean Time).
    - ARC — Indicates the ARC console's 42-year time offset is in effect (for Alpha-based systems only).

Currently, the correct values are:

- `UTC=<value>`, where `<value>` is one of the following boolean values:

  – `true` — Indicates that the clock is set to Universal Time. Any other value indicates that it is set to local time.

- `ARC=<value>`, where `<value>` is the following:

  – `true` — Indicates the ARC console's 42-year time offset is in effect. Any other value indicates that the normal UNIX epoch is assumed (for Alpha-based systems only).

- `ZONE=<filename>` — Indicates the timezone file under `/usr/share/zoneinfo` that `/etc/localtime` is a copy of, such as:

      ZONE="America/New York"

/etc/sysconfig/desktop

The `/etc/sysconfig/desktop` file specifies the desktop manager to be run, such as:

      DESKTOP="GNOME"

/etc/sysconfig/firewall

The `/etc/sysconfig/firewall` file contains various firewall settings. By default, this file (if created) is empty.

/etc/sysconfig/harddisks

The `/etc/sysconfig/harddisks` file allows you to tune your hard drive(s). You can also use `/etc/sysconfig/hardiskhd[a-h]`, to configure parameters for specific drives.

> **WARNING**
>
> **Do not make changes to this file lightly. If you change the default values stored here, you could corrupt all of the data on your hard drive(s).**

The `/etc/sysconfig/harddisks` file may contain the following:

- `USE_DMA=1`, where setting this to 1 enables DMA. However, with some chipsets and hard drive combinations, DMA can cause data corruption. *Check with your hard drive documentation or manufacturer before enabling this.*

- `Multiple_IO=16`, where a setting of 16 allows for multiple sectors per I/O interrupt. When enabled, this feature reduces operating system overhead by 30-50%. *Use with caution.*

- `EIDE_32BIT=3` enables (E)IDE 32-bit I/O support to an interface card.

- `LOOKAHEAD=1` enables drive read-lookahead.

- `EXTRA_PARAMS=` specifies where extra parameters can be added.

/etc/sysconfig/hwconf

The `/etc/sysconfig/hwconf` file lists all the hardware that `kudzu` detected on your system, as well as the drivers used, vendor ID and device ID information. The `kudzu` program detects and configures new and/or changed hardware on a system. The `/etc/sysconfig/hwconf` file is not meant to be manually edited. If you do edit it, devices could suddenly show up as being added or removed.

/etc/sysconfig/i18n

The `/etc/sysconfig/i18n` file sets the default language, such as:

```
LANG="en_US"
```

/etc/sysconfig/init

The `/etc/sysconfig/init` file controls how the system will appear and function during bootup.

The following values may be used:

- `BOOTUP=<value>`, where `<value>` is one of the following:

    - `BOOTUP=color` means the standard color boot display, where the success or failure of devices and services starting up is shown in different colors.

    - `BOOTUP=verbose` means an old style display, which provides more information than purely a message of success or failure.

    - Anything else means a new display, but without ANSI-formatting.

- `RES_COL=<value>`, where `<value>` is the number of the column of the screen to start status labels. Defaults to 60.

- `MOVE_TO_COL=<value>`, where `<value>` moves the cursor to the value in the `RES_COL` line. Defaults to ANSI sequences output by echo `-e`.

- `SETCOLOR_SUCCESS=<value>`, where `<value>` sets the color to a color indicating success. Defaults to ANSI sequences output by echo `-e`, setting the color to green.

- SETCOLOR_FAILURE=*<value>*, where *<value>* sets the color to a color indicating failure. Defaults to ANSI sequences output by echo -e, setting the color to red.

- SETCOLOR_WARNING=*<value>*, where *<value>* sets the color to a color indicating warning. Defaults to ANSI sequences output by echo -e, setting the color to yellow.

- SETCOLOR_NORMAL=*<value>*, where *<value>* sets the color to 'normal'. Defaults to ANSI sequences output by echo -e.

- LOGLEVEL=*<value>*, where *<value>* sets the initial console logging level for the kernel. The default is 7; 8 means everything (including debugging); 1 means nothing except kernel panics. syslogd will override this once it starts.

- PROMPT=*<value>*, where *<value>* is one of the following boolean values:

  - yes — Enables the key check for interactive mode.

  - no — Disables the key check for interactive mode.

/etc/sysconfig/ipchains

The /etc/sysconfig/ipchains file contains information used by the kernel to set up ipchains rules regarding packet filtering.

This file is modified by running the service ipchains save command when valid ipchains rules are in place. You should not manually edit this file. Instead, use the ipchains command to configure the necessary packet filtering rules and then save the rules to this file.

/etc/sysconfig/iptables

Like /etc/sysconfig/ipchains, the /etc/sysconfig/iptables file stores information used by the kernel to provide specialized packet filtering services. However, this file is used by iptables rather than ipchains.

You should not modify this file by hand unless you are familiar with methods used to construct iptables rules. These rules are written to /etc/sysconfig/iptables by the service iptables save command, which stores the current iptables rules by running the /sbin/iptables-save program. Then, when iptables is restarted, such as is the case when the system is booted, the /sbin/iptables-restore program reads the file and reinstitutes the packet filtering rules.

/etc/sysconfig/irda

The /etc/sysconfig/irda file controls how infrared devices on your system are configured at startup.

The following values may be used:

- IRDA=*<value>*, where *<value>* is one of the following boolean values:
  - yes — irattach will be run, which periodically checks to see if anything is trying to connect to the infrared port, such as another notebook computer trying to make a network connection. For infrared devices to work on your system, this line must be set to yes.
  - no — irattach will not be run, preventing infrared device communication.

- DEVICE=*<value>*, where *<value>* is the device (usually a serial port) that handles infrared connections.
- DONGLE=*<value>*, where *<value>* specifies the type of dongle being used for infrared communication. This setting exists for people who use serial dongles rather than real infrared ports. A dongle is a device that is attached to a traditional serial port to communicate via infrared. This line is commented out by default because notebooks with real infrared ports are far more common than computers with add-on dongles.
- DISCOVERY=*<value>*, where *<value>* is one of the following boolean values:
  - yes — Starts irattach in discovery mode, meaning it actively checks for other infrared devices. This needs to be turned on for the machine to be actively looking for an infrared connection (meaning the peer that does not initiate the connection).
  - no — Does not start irattach in discovery mode.

/etc/sysconfig/keyboard
The /etc/sysconfig/keyboard file controls the behavior of the keyboard. The following values may be used:

- KEYBOARDTYPE=sun|pc, which is used on SPARCs only. sun means a Sun keyboard is attached on /dev/kbd, and pc means a PS/2 keyboard connected to a PS/2 port.
- KEYTABLE=*<file>*, where *<file>* is the name of a keytable file. For example: KEYTABLE="us". The files that can be used as keytables start in /usr/lib/kbd/keymaps/i386 and branch into different keyboard layouts from there, all labeled *<file>*.kmap.gz. The first file found beneath /usr/lib/kbd/keymaps/i386that matches the KEYTABLE setting is used.

/etc/sysconfig/kudzu
The /etc/sysconfig/kuzdu allows you to specify a safe probe of your system's hardware by kudzu at boot time. A safe probe is one that disables serial port probing.

- `SAFE=<value>`, where `<value>` is one of the following:
    - `yes` — `kuzdu` does a safe probe.
    - `no` — `kuzdu` does a normal probe.

/etc/sysconfig/mouse

The `/etc/sysconfig/mouse` file is used to specify information about the available mouse. The following values may be used:

- `FULLNAME=<value>`, where `<value>` refers to the full name of the kind of mouse being used.
- `MOUSETYPE=<value>`, where `<value>` is one of the following:
    - `microsoft` — A Microsoft™ mouse.
    - `mouseman` — A MouseMan™ mouse.
    - `mousesystems` — A Mouse Systems™ mouse.
    - `ps/2` — A PS/2 mouse.
    - `msbm` — A Microsoft™ bus mouse.
    - `logibm` — A Logitech™ bus mouse.
    - `atibm` — An ATI™ bus mouse.
    - `logitech` — A Logitech™ mouse.
    - `mmseries` — An older MouseMan™ mouse.
    - `mmhittab` — An mmhittab mouse.

- `XEMU3=<value>`, where `<value>` is one of the following boolean values:
    - `yes` — The mouse only has two buttons, but three mouse buttons should be emulated.
    - `no` — The mouse already has three buttons.

- `XMOUSETYPE=<value>`, where `<value>` refers to the kind of mouse used when X is running. The options here are the same as the `MOUSETYPE` setting in this same file.
- `DEVICE=<value>`, where *<value>* is the mouse device.

In addition, `/dev/mouse` is a symbolic link that points to the actual mouse device.

/etc/sysconfig/network

The /etc/sysconfig/network file is used to specify information about the desired network configuration. The following values may be used:

- NETWORKING=*<value>*, where *<value>* is one of the following boolean values:

  - yes — Networking should be configured.

  - no — Networking should not be configured.

- HOSTNAME=*<value>*, where *<value>* should be the **Fully Qualified Domain Name** (**FQDN**), such as hostname.domain.com, but can be whatever hostname you want.

Note

For compatibility with older software that people might install (such as trn), the /etc/HOSTNAME file should contain the same value as here.

- GATEWAY=*<value>*, where *<value>* is the IP address of the network's gateway.
- GATEWAYDEV=*<value>*, where *<value>* is the gateway device, such as eth0.
- NISDOMAIN=*<value>*, where *<value>* is the NIS domain name.

/etc/sysconfig/pcmcia

The /etc/sysconfig/pcmcia file is used to specify PCMCIA configuration information. The following values may be used:

- PCMCIA=*<value>*, where *<value>* is one of the following:

  - yes — PCMCIA support should be enabled.

  - no — PCMCIA support should not be enabled.

- PCIC=*<value>*, where *<value>* is one of the following:

  - i82365 — The computer has an i82365-style PCMCIA socket chipset.

  - tcic — The computer has a tcic-style PCMCIA socket chipset.

- PCIC_OPTS=*<value>*, where *<value>* is the socket driver (i82365 or tcic) timing parameters.

- CORE_OPTS=*<value>*, where *<value>* is the list of pcmcia_core options.

- CARDMGR_OPTS=*<value>*, where *<value>* is the list of options for the PCMCIA cardmgr (such as -q for quiet mode; -m to look for loadable kernel modules in the specified directory, and so on). Read the cardmgr man page for more information.

/etc/sysconfig/rawdevices

The /etc/sysconfig/rawdevices file is used to configure raw device bindings, such as:

```
/dev/raw/raw1 /dev/sda1
/dev/raw/raw2 8 5
```

/etc/sysconfig/sendmail

The /etc/sysconfig/sendmail file allows messages to be sent to one or more recipients, routing the message over whatever networks are necessary. The file sets the default values for the Sendmail application to run. Its default values are to run as a background daemon, and to check its queue once an hour in case something has backed up.

The following values may be used:

- DAEMON=*<value>*, where *<value>* is one of the following boolean values:

  - yes — Sendmail should be configured to listen to port 25 for incoming mail. yes implies the use of Sendmail's -bd options.

  - no — Sendmail should not be configured to listen to port 25 for incoming mail.

- QUEUE=1h which is given to Sendmail as -q$QUEUE. The -q option is not given to Sendmail if /etc/sysconfig/sendmail exists and QUEUE is empty or undefined.

/etc/sysconfig/soundcard

The /etc/sysconfig/soundcard file is generated by sndconfig and should not be modified. The sole use of this file is to determine what card entry in the menu to pop up by default the next time sndconfig is run. Soundcard configuration information is located in the /etc/modules.conf file.

It may contain the following:

- CARDTYPE=*<value>*, where *<value>* is set to, for example, SB16 for a Soundblaster 16 sound card.

/etc/sysconfig/ups

The /etc/sysconfig/ups file is used to specify information about any **Uninterruptible Power Supplies** (**UPS**) connected to your system. A UPS can be very valuable for a Red Hat Linux system because it gives you time to correctly shut down the system in the case of power interruption. The following values may be used:

- SERVER=*<value>*, where *<value>* is one of the following:

    – yes — A UPS device is connected to your system.

    – no — A UPS device is not connected to your system.

- MODEL=*<value>*, where *<value>* must be one of the following or set to NONE if no UPS is connected to the system:

    – apcsmart — For a APC SmartUPS™ or similar device.

    – fentonups — For a Fenton UPS™.

    – optiups — For an OPTI-UPS™ device.

    – bestups — For a Best Power™ UPS.

    – genericups — For a generic brand UPS.

    – ups-trust425+625 — For a Trust™ UPS.

- DEVICE=*<value>*, where *<value>* specifies where the UPS is connected, such as /dev/ttyS0.

- OPTIONS=*<value>*, where *<value>* is a special command that needs to be passed to the UPS.

/etc/sysconfig/vncservers

The /etc/sysconfig/vncservers file configures how the **Virtual Network Computing** (**VNC**) server starts up. VNC is a remote display system which allows you to view a desktop environment not only on the machine where it is running but across different networks (from a LAN to the Internet) and using a wide variety of machine architectures.

It may contain the following:

- VNCSERVERS=`<value>`, where `<value>` is set to something like `"1:fred"`, to indicate that a VNC server should be started for user fred on display :1. User fred must have set a VNC password using `vncpasswd` before attempting to connect to the remote VNC server.

Note that when you use a VNC server, your communication with it is unencrypted, and so it should not be used on an untrusted network. For specific instructions concerning the use of SSH to secure the VNC communication, please read the information found at http://www.uk.research.att.com/vnc/sshvnc.html. To find out more about SSH, see Chapter 10, *SSH Protocol* or *Official Red Hat Linux Customization Guide*.

3.4 Init Runlevels

The idea behind operating different services at different runlevels essentially revolves around the fact that different systems can be used in a different ways. Some services cannot be used until the system is in a particular state, or **mode**, such as ready for more than one user or has networking available.

There are times in which you may want to operate the system at a lower mode, such as fixing disk corruption problems in runlevel 1 so no other users can possibly be on the system or leaving a server in runlevel 3 without an X session running. In these cases, running services that depend upon a higher system mode to function does not make sense because they will not work correctly anyway. By already having each service assigned to start when its particular runlevel is reached, you ensure an orderly start up process and can quickly change the mode of the machine without worrying about which services to manually start or stop.

Generally, Red Hat Linux operates in runlevel 3 — full multi-user mode. The following runlevels are defined in Red Hat Linux:

- 0 — Halt

- 1 — Single-user mode

- 2 — Not used (user-definable)

- 3 — Full multi-user mode

- 4 — Not used (user-definable)

- 5 — Full multi-user mode (with an X-based login screen)

- 6 — Reboot

The default runlevel for a system to boot to and stop is configured in `/etc/inittab`. For more information on `/etc/inittab`, see Section 3.2.3, *SysV Init*.

Feel free to configure runlevels 2 and 4 as you see fit. Many users configure those runlevels in a way that makes the most sense for them while leaving the standard runlevels 3 and 5 alone. This allows

them to quickly move in and out of their custom configuration without disturbing the normal set of features at the standard runlevels.

If your machine gets into a state where it will not boot due to a bad /etc/inittab or will not let you log in because you have a corrupted /etc/passwd (or if you have simply forgotten your password), boot into single-user mode.

If you are using LILO, you can enter single-user mode by typing **linux single** at the LILO boot: prompt.

If you are using GRUB as your boot loader, you can enter single-user mode using the following steps.

- In the graphical GRUB boot loader screen, select the **Red Hat Linux** boot label and press [e] to edit it.

- Arrow down to the kernel line and press [e] to edit it.

- At the prompt, type single and press [Enter].

- You will be returned to the GRUB screen with the kernel information. Press the [b] key to boot the system into single user mode.

A very bare system will boot, and you will have a command shell from which you can fix things.

If this does not work, you will need to boot by typing **linux init=/bin/bash** at the LILO boot: prompt. This places you at a shell prompt; note that no filesystems other than the root filesystem are mounted, and the root filesystem is mounted in read-only mode. To mount it in read-write mode (to allow editing of a broken /etc/inittab, for example) do:

```
mount -n /proc
mount -o rw,remount /
```

3.4.1 Initscript Utilities

The chkconfig utility in /sbin provides a simple command-line tool for maintaining the /etc/rc.d/init.d directory hierarchy. It relieves system administrators from having to directly manipulate the numerous symbolic links in the directories under /etc/rc.d.

In addition, there is ntsysv that provides a text-based interface, which you may find easier to use than chkconfig's command-line interface.

If you prefer a graphical interface, use the serviceconf program.

All of these utilities must be run as root.

Please refer to the *Official Red Hat Linux Customization Guide* for more information regarding these tools.

3.5 Running Programs at Boot Time

The file /etc/rc.d/rc.local script is run by init at boot time, after all other initialization is complete, and whenever you change runlevels. You can add additional initialization commands here. For instance, you may want to start up additional daemons or initialize a printer.

In addition, if you require serial port setup, you can create and edit /etc/rc.serial, and it will be executed automatically at boot time. This script can run a number of setserial commands to specially configure the system's serial ports. See the setserial man page for more information.

3.6 Shutting Down

To shut down Red Hat Linux, issue the shutdown command. You can read the shutdown man page for complete details, but the two most common uses are:

```
/sbin/shutdown -h now
/sbin/shutdown -r now
```

You must run shutdown as root. After shutting everything down, the -h option will halt the machine, and the -r option will reboot.

Although the reboot and halt commands are now able to invoke shutdown if run while the system is in runlevels 1-5, it is a bad habit to get into, as not all Linux-like operating systems have this feature.

WARNING

If your computer does not power itself down, you should not turn off the computer until you see a message indicating that the system is halted or finished shutting down.

Failure to wait for this message will mean that you may be turning off the machine before your hard drive partitions are unmounted. This can cause filesystem corruption, even to the point where your system may not boot the next time it attempts to start up. Be patient when halting your system.

3.7 Differences in the Boot Process of Other Architectures

Each computer architecture supported by Red Hat Linux boots the operating system in a different way. However, once the Red Hat Linux kernel begins booting and hands off the boot process to init, the same events happen on each architecture in exactly the same way. The only difference is in the way Red Hat Linux finds the kernel to load it in order to start init.

For example, the Alpha architecture uses the aboot boot loader, while the Itanium architecture uses the ELILO boot loader.

Consult the installation information for each of the architectures for detailed information about the different boot methods.

4 The `/proc` Filesystem

The /proc directory contains virtual files that are windows into the current state of the running Linux kernel. This allows the user to peer into a vast array of information, effectively providing them with the kernel's point-of-view within the system. In addition, the user can use the /proc directory to communicate particular configuration changes to the kernel.

4.1 A Virtual Filesystem

In Linux, everything is stored in files. Most users are familiar with the two primary types of files, text and binary. However, the /proc directory contains files that are not part of any filesystem associated with your hard disks, CD-ROM, or any other physical storage device connected to your system (except, arguably, your RAM). Rather, these files are part of a **virtual filesystem**, enabled or disabled in the Linux kernel when it is compiled.

By default, when a Red Hat Linux system starts up, a line in /etc/fstab is responsible for mounting the /proc filesystem.

```
none            /proc        proc         defaults      0 0
```

The status of whether or not /proc is currently mounted can be determined by typing the mount command with no arguments. This will display all of your current mounts, and a line similar to this should appear in the list if /proc is mounted:

```
none on /proc type proc (rw)
```

If you would like to mount /proc manually, type this command:

```
mount -t proc proc /proc
```

The /proc virtual filesystem is a switch in the configuration of the Linux kernel, one that is turned on by default. If, for whatever reason, you would like to completely disable /proc on your system, de-select **/proc file system support** within the **File system** configuration section of config, menu-config, or xconfig when rebuilding your kernel. Alternatively, you can simply comment out the /proc line in /etc/fstab to prevent it from being mounted.

The best way to understand /proc as a virtual filesystem is to list the files in the directory. The following is only a partial excerpt of such a list:

```
[root@bleach /]# ls -l /proc
-r--r--r--    1 root     root             0 May  3 11:42 cmdline
-r--r--r--    1 root     root             0 May  3 11:42 cpuinfo
-r--r--r--    1 root     root             0 May  3 11:42 devices
-r--r--r--    1 root     root             0 May  3 11:42 dma
dr-xr-xr-x    2 root     root             0 May  3 11:42 driver
```

```
-r--r--r--    1 root      root                   0 May  3 11:42 execdomains
-r--r--r--    1 root      root                   0 May  3 11:42 fb
-r--r--r--    1 root      root                   0 May  3 11:42 filesystems
[root@bleach /]#
```

The `/proc` virtual files exhibit some interesting qualities. First, most of them are 0 bytes in size. However, when the file is viewed, it likely contains quite a bit of information. In addition, most of their time and date settings reflect the current time and date, meaning that they are constantly changing.

Various programs use the `/proc` filesystem to discover the parameters of a system so that they can provide better performance and deliver more features.

In addition, a system administrator can use `/proc` as an easy method of accessing information about the state of the kernel, the attributes of the machine, the states of individual processes, and more. Most of the files in this directory, such as `interrupts`, `meminfo`, `mounts`, and `partitions`, provide an up-to-the-moment glimpse of a system's environment. To make things easier, files that contain information covering a similar topic are grouped into virtual directories and sub-directories, such as `/proc/ide`.

4.1.1 Viewing Virtual Files

By using `cat`, `more`, or `less` commands in combination with the files within `/proc`, you can immediately access an enormous amount of information about the system. As an example, if you want to see how the memory registers are currently assigned on your computer:

```
[root@bleach /]# cat /proc/iomem
00000000-0009fbff : System RAM
0009fc00-0009ffff : reserved
000a0000-000bffff : Video RAM area
000c0000-000c7fff : Video ROM
000f0000-000fffff : System ROM
00100000-03ffcfff : System RAM
  00100000-002557df : Kernel code
  002557e0-0026c80b : Kernel data
03ffd000-03ffefff : ACPI Tables
03fff000-03ffffff : ACPI Non-volatile Storage
dc000000-dfffffff : S3 Inc. ViRGE/DX or /GX
e3000000-e30000ff : Lite-On Communications Inc LNE100TX
  e3000000-e30000ff : eth0
e4000000-e7ffffff : Intel Corporation 440BX/ZX - 82443BX/ZX Host bridge
ffff0000-ffffffff : reserved
[root@bleach /]#
```

Or (and more usefully), if were connecting to an unknown machine and wanted to know its CPU type and speed, you can use the following command:

```
[root@bleach /]# cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model   : 6
model name : Celeron (Mendocino)
stepping : 0
cpu MHz  : 416.537
cache size : 128 KB
fdiv_bug : no
hlt_bug  : no
f00f_bug : no
coma_bug : no
fpu  : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags  : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov
bogomips : 830.66
[root@bleach ide]#
```

As you can see, some of the information makes sense immediately, while other areas seem to be in a strange code. Some of the /proc files will be senseless without a legend to guide you. In many cases, utilities exist on the system, such as free and top, that pull data from these files and display it in a useful way.

Another interesting quality of virtual files can be seen when viewing them with the more command, which usually tells gives your location in the file by displaying the percentage of the document you are currently seeing. This percentage number usually climbs the further you navigate down a long file. However, when viewing a /proc virtual file, the percentage amount never changes, always staying at 0%.

A few of the files in /proc are set to only be readable by root, so you may need to become the root user before attempting to read them.

WARNING

Be sure to avoid viewing the kcore file in /proc. This virtual file contains an image of the kernel's memory, and the contents of the file will do strange things to your terminal. You may need to type reset after hitting [Ctrl]-[C] to get back to a proper command line prompt.

4.2 Top-Level Files in `/proc`

Most of the files at the top-level of the `/proc` directory hold key pieces of information about the state of the Linux kernel and your system in general.

It is important to remember that the content of the files in the `/proc` directory and its various sub-directories is entirely dependent on information concerning your system. In other words, do not expect to see the exact same information in the same `/proc` file on two different machines. In addition, depending on the version of the Linux kernel and the devices connected to your system, some of the files described here may not be found in your `/proc` directory. Likewise, additional files or directories may be on your system but are not described here.

Rather than attempting to be a comprehensive collection of these files and the information they contain, the following list is designed to showcase some of the more common and useful ones. The objective is to provide system administrators with a place to find current data on their systems when other tools will not do the job.

4.2.1 `/proc/apm`

This file provides information about the **Advanced Power Management (APM)** state and options on the system. This information is used by the kernel to provide information for the `apm` command.

The output of this file on a system without a battery and constantly connected to an AC power source looks similar to this:

```
1.14 1.2 0x03 0x01 0xff 0x80 -1% -1 ?
```

Executing an `apm` command on these systems results in something similar to this:

```
[root@bleach /proc]# apm -v
APM BIOS 1.2 (kernel driver 1.14)
AC on-line, no system battery
[root@bleach /proc]#
```

For these systems, `apm` may be able to do little more than put the machine in a standby mode, commonly known as "putting the system to sleep." Note that this state is only possible if your system BIOS supports it. Trying to put a system in standby mode that is not designed for it can make the system unstable.

The `apm` command is much more useful on laptops and other portable Linux systems. This is also reflected in their `/proc/apm` files. This is the output from a sample file on a laptop running Linux while plugged into a power outlet:

```
1.14 1.2 0x03 0x01 0x03 0x09 100% -1 ?
```

When the same machine is unplugged from its power source and running on its own batteries for a few minutes, you will see the contents of the apm file change:

```
1.14 1.2 0x03 0x00 0x00 0x01 99% 1792 min
```

In this state, the apm command yields readable information from this data:

```
[ed@blink /]$ apm -v
APM BIOS 1.2 (kernel driver 1.14)
AC off-line, battery status high: 99% (1 day, 5:52)
[ed@blink /]$
```

This demonstrates the connection between data located in raw `/proc` files and the utilities designed to use that information for specific purposes.

4.2.2 `/proc/cmdline`

This file essentially shows the parameters passed to the Linux kernel at the time it is started. A sample `/proc/cmdline` file looks similar to this:

```
auto BOOT_IMAGE=linux ro root=305 BOOT_FILE=/boot/vmlinuz-2.4.2-2
```

The important data contained in the file breaks down in the following way:

- `BOOT_IMAGE=linux`, which tells you the name of the image used to boot the machine.

- `ro`, which signifies that the kernel is loaded read-only.

- `BOOT_FILE=/boot/vmlinuz-2.4.2-2` notes the actual Linux kernel file used to boot the system.

4.2.3 `/proc/cpuinfo`

This file changes based on the type of processor in your system. The output is fairly easy to understand. A sample file looks like this:

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model   : 6
model name : Celeron (Mendocino)
stepping : 0
cpu MHz   : 334.099
cache size : 128 KB
fdiv_bug : no
hlt_bug  : no
f00f_bug : no
coma_bug : no
```

```
fpu  : yes
fpu_exception : yes
cpuid level : 2
wp  : yes
flags  : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov
bogomips : 666.82
```

Quite a bit of information is available here. Among the highlights:

- `processor` — Provides each processor with an identifying number. If you only have one processor, you will only see a 0. However, if you are using a machine with multiple processors, each of the processors will have it own number, increasing by one as you go down the list, and its own collection of information.

- `cpu family` — Authoritatively tells you the type of processor you have in the system. Simply place the number in front of "86" to calculate the value. This is particularly helpful if you are wondering about the architecture of an older system (686, 586, 486, or 386). As RPM packages are occasionally compiled for particular architectures, this value tells you which package to install on the system.

- `model name` — Gives you the popular name of the processor, including its project name.

- `cpu MHz` — Shows the precise speed in megahertz of that particular processor (in thousandths).

- `cache size` — Tells you the amount of level 2 memory cache available to the processor.

- `flags` — Defines a number of different qualities about the processor, such as the presence of a floating point unit (FPU) and the ability to process MMX instructions.

4.2.4 `/proc/devices`

This file displays the various character and block devices currently configured for use with the kernel. It does not include modules that are available but not loaded into the kernel. Sample output from this file looks similar to this:

```
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 ttyS
  5 cua
  7 vcs
 10 misc
 29 fb
 36 netlink
128 ptm
136 pts
```

```
162 raw
180 usb

Block devices:
  1 ramdisk
  2 fd
  3 ide0
  9 md
 22 ide1
```

The output from `/proc/devices` includes the major number and name of the device.

Character devices are similar to **block devices**, except for two basic differences.

First, block devices have a buffer available for requests sent to them, allowing them to order the requests before dealing with them. This comes in very handy with devices designed to store information, such as hard drives, because the ability to order the information before writing it to the device allows it to be placed in more efficient order. Character devices do not require this kind of buffering.

Second, block devices can send and receive information in blocks of a particular size, which can be configured to meet the requirements of the particular device. Character devices send data in as many or few bytes as they see fit, with no preconfigured size.

To discover if a particular device is a block or character device, type the `ls -l <device-name>` command. If the first character in the response is a b, then it is a block device; if it is a c, then it is character device. For example, note the output from a look at some common devices (`hda` is the first IDE hard drive and `tty0` is the first terminal port) from the `/dev` directory:

```
[root@bleach /]# ls -l /dev/hda /dev/tty0
brw-rw----    1 root      disk       3,   0 Mar 23 23:37 /dev/hda
crw--w----    1 truk      truk       4,   0 May  3 16:28 /dev/tty0
[root@bleach /]#
```

More information about devices can be found in `/usr/src/linux-2.4/Documentation/devices.txt`.

4.2.5 `/proc/dma`

This file contains a list of the registered ISA direct memory access (DMA) channels in use. A sample `/proc/dma` files looks like this:

```
4: cascade
```

4.2.6 `/proc/execdomains`

This file lists the execution domains currently supported by the Linux kernel, along with the range of personalities they support.

```
0-255    Linux              [kernel]
```

Think of **execution domains** as a kind of "personality" of a particular operating system. Other binary formats, such as Solaris, UnixWare, and FreeBSD, can be used with Linux. By changing the personality of a task running in Linux, a programmer can change the way the operating system treats particular system calls from a certain binary. Except for the PER_LINUX execution domain, they can be implemented as dynamically loadable modules.

4.2.7 `/proc/fb`

This file contains a list of frame buffer devices, with the frame buffer device number and the driver that controls it. Typical output of `/proc/fb` for systems that contain frame buffer devices looks similar to this:

```
0 VESA VGA
```

4.2.8 `/proc/filesystems`

This file displays a list of the filesystem types currently supported by the kernel. Sample output from a generic kernel's `/proc/filesystems` file looks similar to this:

```
nodev sockfs
nodev tmpfs
nodev shm
nodev pipefs
nodev proc
 ext2
 iso9660
nodev devpts
nodev usbdevfs
nodev autofs
```

The first column signifies whether the filesystem is mounted on a block device, with those containing nodev in this column signifying that they are not mounted on a block device. The second column lists the name of the filesystems supported.

This information is used by the mount command to cycle through the possible filesystems when one is not specified as an argument.

4.2.9 `/proc/interrupts`

This file records the number of interrupts per IRQ on the x86 architecture. A standard `/proc/interrupts` looks similar to this:

```
           CPU0
   0:    8399367              XT-PIC  timer
```

```
    1:          339          XT-PIC    keyboard
    2:            0          XT-PIC    cascade
    5:        80111          XT-PIC    usb-uhci, eth0
    8:            1          XT-PIC    rtc
   12:         6107          XT-PIC    PS/2 Mouse
   14:        60324          XT-PIC    ide0
   15:       541741          XT-PIC    ide1
  NMI:            0
  ERR:            0
```

For a multi-processor machine, this file may look slightly different:

```
            CPU0        CPU1
   0: 1366814704           0           XT-PIC    timer
   1:        128         340    IO-APIC-edge   keyboard
   2:          0           0           XT-PIC    cascade
   8:          0           1    IO-APIC-edge   rtc
  12:       5323        5793    IO-APIC-edge   PS/2 Mouse
  13:          1           0           XT-PIC    fpu
  16:   11184294    15940594   IO-APIC-level   Intel EtherExpress Pro 10/100 Ethernet
  20:    8450043    11120093   IO-APIC-level   megaraid
  30:      10432       10722   IO-APIC-level   aic7xxx
  31:         23          22   IO-APIC-level   aic7xxx
 NMI:          0
 ERR:          0
```

The first column refers to the IRQ number. Each CPU in the system has its own column and its own number of interrupts per IRQ. The next column tells you the type of interrupt, and the last column contains the name of the device that is located at that IRQ.

Each of the types of interrupts seen in this file, which are architecture-specific, mean something a little different. For x86 machines, the following values are common:

- `XT-PIC` — The old AT computer interrupts that have been around for a long time.

- `IO-APIC-edge` — The voltage signal on this interrupt transitions from low to high, creating an edge, where the interrupt occurs and is only signaled once. This kind of interrupt, as well as the `IO-APIC-level` interrupt, are only seen on systems with processors from the 586 family and higher.

- `IO-APIC-level` — Generates interrupts when its voltage signal goes high until the signal goes low again.

4.2.10 `/proc/iomem`

This file shows you the current map of the system's memory for its various devices:

```
00000000-0009fbff : System RAM
0009fc00-0009ffff : reserved
000a0000-000bffff : Video RAM area
000c0000-000c7fff : Video ROM
000f0000-000fffff : System ROM
00100000-03ffcfff : System RAM
  00100000-002557df : Kernel code
  002557e0-0026c80b : Kernel data
03ffd000-03ffefff : ACPI Tables
03fff000-03ffffff : ACPI Non-volatile Storage
dc000000-dfffffff : S3 Inc. ViRGE/DX or /GX
e3000000-e30000ff : Lite-On Communications Inc LNE100TX
  e3000000-e30000ff : eth0
e4000000-e7ffffff : Intel Corporation 440BX/ZX - 82443BX/ZX Host bridge
ffff0000-ffffffff : reserved
```

The first column displays the memory registers used by each of the different types of memory. The second column tells the kind of memory located within those registers. In particular, this column will even tell you which memory registers are used by the kernel within the system RAM or, if you have multiple Ethernet ports on your NIC, the memory registers assigned for each port.

4.2.11 `/proc/ioports`

In a way similar to `/proc/iomem`, `/proc/ioports` provides a list of currently registered port regions used for input or output communication with a device. This file can be quite long, with a beginning similar to this:

```
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
```

The first column gives the actual IO port address range reserved for the device listed in the second column.

4.2.12 `/proc/isapnp`

This file lists **Plug and Play (PnP)** cards in ISA slots on the system. This is most often seen with sound cards but may include any number of devices. A /proc/isapnp file with Soundblaster entry in it looks similar to this:

```
Card 1 'CTL0070:Creative ViBRA16C PnP' PnP version 1.0 Product version 1.0
  Logical device 0 'CTL0001:Audio'
    Device is not active
    Active port 0x220,0x330,0x388
    Active IRQ 5 [0x2]
    Active DMA 1,5
    Resources 0
      Priority preferred
      Port 0x220-0x220, align 0x0, size 0x10, 16-bit address decoding
      Port 0x330-0x330, align 0x0, size 0x2, 16-bit address decoding
      Port 0x388-0x3f8, align 0x0, size 0x4, 16-bit address decoding
      IRQ 5 High-Edge
      DMA 1 8-bit byte-count compatible
      DMA 5 16-bit word-count compatible
      Alternate resources 0:1
        Priority acceptable
        Port 0x220-0x280, align 0x1f, size 0x10, 16-bit address decoding
        Port 0x300-0x330, align 0x2f, size 0x2, 16-bit address decoding
        Port 0x388-0x3f8, align 0x0, size 0x4, 16-bit address decoding
        IRQ 5,7,2/9,10 High-Edge
        DMA 1,3 8-bit byte-count compatible
        DMA 5,7 16-bit word-count compatible
```

This file can be quite long, depending on the number of devices displayed here and their requirements or requests for resources.

Each card lists its name, PnP version number, and product version number. If the device is active and configured, this file will also reveal the port and IRQ numbers for the device. In addition, to ensure better compatibility, the card will specify preferred and acceptable values for a number of different parameters. The goal here is to allow the PnP cards to work around one another and avoid IRQ and port conflicts.

4.2.13 `/proc/kcore`

This file represents the physical memory of the system and is stored in the core file format. Unlike most /proc files, kcore does display a size. This value is given in bytes and is equal to the size of physical memory (RAM) used plus 4KB.

Do not try to `cat` or otherwise attempt to view this file. Its contents are designed to be examined by a debugger, such as `gdb`, the GNU Debugger.

Only the root user has the rights to view this file.

4.2.14 /proc/kmsg

This file is used to hold messages generated by the kernel. These messages are then picked up by other programs, such as `klogd`.

4.2.15 /proc/ksyms

This file holds the kernel exported symbol definitions used by the modules tools to dynamically link and bind loadable modules.

```
e003def4 speedo_debug [eepro100]
e003b04c eepro100_init [eepro100]
e00390c0 st_template [st]
e002104c RDINDOOR [megaraid]
e00210a4 callDone [megaraid]
e00226cc megaraid_detect [megaraid]
```

The second column refers to the name of a kernel function, and the first column lists the memory address of that function in the kernel. The last column reveals the name of the module loaded to provide that function.

4.2.16 /proc/loadavg

This file provides a look at load average, or the utilization of the processor, over time, as well as giving additional data used by `uptime` and other commands. A sample `loadavg` file looks similar to this:

```
0.20 0.18 0.12 1/80 11206
```

The first three columns measure CPU utilization of the last 1, 5, and 10 minute periods. The fourth column shows the number of currently running processes and the total number of processes. The last column displays the last process ID used.

4.2.17 /proc/locks

This files displays the files currently locked by the kernel. The content of this file contains kernel internal debugging data and can vary greatly, depending on the use of the system. A sample `locks` file of a very lightly loaded system looks similar to this:

```
1: FLOCK  ADVISORY  WRITE 807 03:05:308731 0 EOF c2a260c0 c025aa48 c2a26120
2: POSIX  ADVISORY  WRITE 708 03:05:308720 0 EOF c2a2611c c2a260c4 c025aa48
```

Each lock is assigned a unique number at the beginning of each line. The second column refers to the class of lock used, with FLOCK signifying the older-style UNIX file locks from a `flock` system call and POSIX representing the newer POSIX locks from the `lockf` system call.

The third column can have two values. ADVISORY means that the lock does not prevent other people from accessing the data; it only prevents other attempts to lock it. MANDATORY means that no other access to the data is permitted while the lock is held. The fourth column reveals whether the lock is allowing the holder READ or WRITE access to the file, and the fifth column shows the ID of the process holding the lock.

The sixth column shows the ID of the file being locked, in the format of *MAJOR-DEVICE:MINOR-DEVICE:INODE-NUMBER*. The seventh column shows the start and end of the file's locked region. The remaining columns point to internal kernel data structures used for specialized debugging and can be ignored.

4.2.18 `/proc/mdstat`

This file contains the current information for multiple-disk, RAID configurations. If your system does not contain such a configuration, then your mdstat file will look similar to this:

```
Personalities :
read_ahead not set
unused devices: <none>
```

Things really do not get interesting unless you have md devices created and in use. In that case, you can use mdstat to give you a picture of what is currently happening with your md*X* devices.

This `/proc/mdstat` file shows a system with its md0 configured as a RAID 1 device. It is currently re-syncing the disks, and the percentage completed and estimated time remaining can be seen:

```
Personalities : [linear] [raid1]
read_ahead 1024 sectors
md0: active raid1 sda2[1] sdb2[0] 1943840 blocks [2/2] [UU] resync=1% finish=12.3min
algorithm 2 [3/3] [UUU]
unused devices: <none>
```

4.2.19 `/proc/meminfo`

This is one of the more commonly used `/proc` files, as it reports back plenty of valuable information about the current utilization of RAM on the system. A system with 256MB of RAM and 384MB of swap space might have a `/proc/meminfo` file similar to this one:

```
        total:    used:    free:  shared: buffers:  cached:
Mem:  261709824 253407232  8302592       0 120745984 48689152
Swap: 402997248     8192 402989056
MemTotal:       255576 kB
```

```
MemFree:            8108 kB
MemShared:             0 kB
Buffers:          117916 kB
Cached:            47548 kB
Active:           135300 kB
Inact_dirty:       29276 kB
Inact_clean:         888 kB
Inact_target:          0 kB
HighTotal:             0 kB
HighFree:              0 kB
LowTotal:         255576 kB
LowFree:            8108 kB
SwapTotal:        393552 kB
SwapFree:         393544 kB
```

Much of the information here is used by the `top` command. In fact, the output of the `free` command is even similar in appearance to the contents and structure of `meminfo`. By looking directly at `meminfo`, more memory details are revealed:

- `Mem` — Displays the current state of physical RAM in the system, including a full breakdown of total, used, free, shared, buffered, and cached memory utilization in bytes.

- `Swap` — Displays the total, used, and free amounts of swap space, in bytes.

- `MemTotal` — Total amount of physical RAM, in kilobytes.

- `MemFree` — The amount of physical RAM, in kilobytes, left unused by the system.

- `MemShared` — Unused with 2.4 and higher kernels but left in for compatibility with earlier kernel versions.

- `Buffers` — The amount of physical RAM, in kilobytes, used for file buffers.

- `Cached` — The amount of physical RAM, in kilobytes, used as cache memory.

- `Active` — The total amount of buffer or page cache memory, in kilobytes, that is in active use.

- `Inact_dirty` — The total amount of buffer or cache pages, in kilobytes, that might be freeable.

- `Inact_clean` — The total amount of buffer or cache pages in kilobytes that are definitely free and available.

- `Inact_target` — The net amount of allocations per second, in kilobytes, averaged over one minute.

- `HighTotal` and `HighFree` — The total and free amount of memory, respectively, that is not directly mapped into kernel space. The `HighTotal` value can vary based on the type of kernel used.

- `LowTotal` and `LowFree` — The total and free amount of memory, respectively, that is directly mapped into kernel space. The `LowTotal` value can vary based on the type of kernel used.

- `SwapTotal` — The total amount of swap available, in kilobytes.

- `SwapFree` — The total amount of swap free, in kilobytes.

4.2.20 `/proc/misc`

This file lists miscellaneous drivers registered on the miscellaneous major device, which is number 10:

```
135 rtc
  1 psaux
134 apm_bios
```

The first column is the minor number of each device, and the second column shows the driver in use.

4.2.21 `/proc/modules`

This file displays a list of all modules that have been loaded by the system. Its contents will vary based on the configuration and use of your system, but it should be organized in a similar manner to this sample `/proc/modules` file output:

```
tulip                38544  1 (autoclean)
ide-cd               26848  0 (autoclean)
cdrom                27232  0 (autoclean) [ide-cd]
autofs               11264  1 (autoclean)
ipchains             38976  0 (unused)
usb-uhci             20720  0 (unused)
usbcore              49664  1 [usb-uhci]
```

The first column contains the name of the module. The second column refers to the memory size of the module, in bytes. The third column tells you whether the module is currently loaded (`1`) or unloaded (`0`). The final column states if the module can unload itself automatically after a period without use (`autoclean`) or if it is not being utilized (`unused`). Any module with a line containing a name listed in brackets (`[` or `]`) tells you that this module depends upon another module to be present in order to function.

4.2.22 `/proc/mounts`

This file provides a quick list of all mounts in use by the system:

```
/dev/root / ext2 rw 0 0
/proc /proc proc rw 0 0
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hda1 /boot ext2 rw 0 0
/dev/hda7 /home ext2 rw 0 0
```

```
none /dev/pts devpts rw 0 0
automount(pid696) /misc autofs rw 0 0
```

The output found here is similar to contents of `/etc/mtab`, except that `/proc/mount` can be more current.

The first column specifies the device that is mounted, with the second column revealing the mount-point. The third column tells the filesystem type, and the fourth column tells you if it is mounted read-only (`ro`) or read-write (`rw`). The fifth and sixth columns are dummy values designed to match the format used in `/etc/mtab`.

4.2.23 `/proc/mtrr`

This file refers to the current Memory Type Range Registers (MTRRs) in use with the system. If your system's architecture supports MTRRs, your `mtrr` might look something like this:

```
reg00: base=0x00000000 (   0MB), size=  64MB: write-back, count=1
```

MTRRs are used with Intel P6 family of processors (Pentium Pro and higher), and they are used to control processor access to memory ranges. When using a video card on a PCI or AGP bus, a properly configured `mtrr` file can increase performance over 150%.

Most of the time, this value is properly configured for you. For more information on MTRRs and manually configuring this file, please see http://web1.linuxhq.com/kernel/v2.3/doc/mtrr.txt.html.

4.2.24 `/proc/partitions`

For very detailed information on the various partitions currently available to the system,

```
major minor  #blocks  name      rio rmerge rsect ruse wio wmerge wsect wuse runni
    3     0   6297480 hda 103927 109145 1549044 1461980 66873 30417 780568 60414
    3     1     56196 hda1 299 1995 4588 1300 17 9 52 5450 0 5210 6750
    3     2         1 hda2 0 0 0 0 0 0 0 0 0 0 0 0
    3     5   4610623 hda5 95638 62150 1262322 1304320 63580 16715 644512 539971
    3     6    136521 hda6 6808 22109 231336 148110 2384 13484 127608 485020 0 1
    3     7   1494013 hda7 1182 22891 50798 8250 892 209 8396 151240 0 86990 159
```

Most of the information here is of little importance to most users, except for the following lines:

• `major` — The major number of the device with this partition. The major number in our example (3) corresponds with the `ide0` device in `/proc/devices`, letting us know the kind of device driver used to interact with that partition.

• `minor` — The minor number of the device with this partition. This serves to separate the partitions into different physical devices and relates to the number at the end of the name of the partition.

• `#blocks` — Lists the number of physical disk blocks contained in a particular partition.

- name — The name of the partition.

4.2.25 `/proc/pci`

This file contains a full listing of every PCI device on your system. Depending on the number of PCI devices you have, /proc/pci can get rather long. An example from this file on a basic system looks similar to this:

```
Bus  0, device   0, function  0:
  Host bridge: Intel Corporation 440BX/ZX - 82443BX/ZX Host bridge (rev 3).
    Master Capable.  Latency=64.
    Prefetchable 32 bit memory at 0xe4000000 [0xe7ffffff].
Bus  0, device   1, function  0:
  PCI bridge: Intel Corporation 440BX/ZX - 82443BX/ZX AGP bridge (rev 3).
    Master Capable.  Latency=64.  Min Gnt=128.
Bus  0, device   4, function  0:
  ISA bridge: Intel Corporation 82371AB PIIX4 ISA (rev 2).
Bus  0, device   4, function  1:
  IDE interface: Intel Corporation 82371AB PIIX4 IDE (rev 1).
    Master Capable.  Latency=32.
    I/O at 0xd800 [0xd80f].
Bus  0, device   4, function  2:
  USB Controller: Intel Corporation 82371AB PIIX4 USB (rev 1).
    IRQ 5.
    Master Capable.  Latency=32.
    I/O at 0xd400 [0xd41f].
Bus  0, device   4, function  3:
  Bridge: Intel Corporation 82371AB PIIX4 ACPI (rev 2).
    IRQ 9.
Bus  0, device   9, function  0:
  Ethernet controller: Lite-On Communications Inc LNE100TX (rev 33).
    IRQ 5.
    Master Capable.  Latency=32.
    I/O at 0xd000 [0xd0ff].
    Non-prefetchable 32 bit memory at 0xe3000000 [0xe30000ff].
Bus  0, device  12, function  0:
  VGA compatible controller: S3 Inc. ViRGE/DX or /GX (rev 1).
    IRQ 11.
    Master Capable.  Latency=32.  Min Gnt=4.Max Lat=255.
    Non-prefetchable 32 bit memory at 0xdc000000 [0xdfffffff].
```

This output shows a list of all PCI devices, sorted in the order of bus, device, and function. Beyond providing the name and version of the device, which is always nice to know when you forget the brand of your network interface card, this list also gives you detailed IRQ information so you can quickly look for conflicts.

4.2.26 `/proc/slabinfo`

This file gives information about memory usage on the slab level. Linux kernels greater than 2.2 use **slab pools** to manage memory above the page level. Commonly used objects have their own slab pools.

The `/proc/slabinfo` file can be rather long, but it starts off similar to this:

```
slabinfo - version: 1.1
kmem_cache             59     78    100    2    2    1
ip_fib_hash            10    113     32    1    1    1
ip_conntrack            0      0    352    0    0    1
urb_priv                0      0     32    0    0    1
uhci_desc            1038   1062     64   18   18    1
clip_arp_cache          0      0    128    0    0    1
ip_mrt_cache            0      0     96    0    0    1
tcp_tw_bucket           0      0    128    0    0    1
tcp_bind_bucket         6    113     32    1    1    1
tcp_open_request        0      0     96    0    0    1
inet_peer_cache         0      0     64    0    0    1
ip_dst_cache           26     40    192    2    2    1
```

The values in this file occur in the following order: cache name, number of active objects, number of total objects, size of the object, number of active slabs (blocks) of the objects, total number of slabs of the objects, and the number of pages per slab.

It should be noted that active in this case means to be in use. An active object is one that is in use, and an active slab is one that contains any used objects.

4.2.27 `/proc/stat`

This file keeps track of a variety of different statistics about the system since it was last restarted. The contents of `/proc/stat`, which can be quite long, begins something like this:

```
cpu  7361636 3040186 1150480 23431255
cpu0 7361636 3040186 1150480 23431255
page 213089 98198
swap 28914 15951
intr 37566857 34983557 1313 0 4 4 128683   <CONTENT-SNIPPED>
disk_io: (3,0):(171639,103942,1549132,67697,784888)
ctxt 323724291
btime 988921599
processes 14882
kstat.input_fastpath: 0
kstat.input_slowpath: 0
kstat.inputqueue_got_packet: 0
```

```
kstat.inputqueue_no_packet: 0
```

Some of the more popular statistics include:

- `cpu` — Measures the number of **jiffies** (1/100ths of a second) that the system has been in user mode, user mode with low priority (nice), system mode, and the idle task, respectively. The total for all CPUs is given at the top, and each individual CPU is listed below with its own statistics.

- `page` — The number of pages the system has paged in and out from disk.

- `swap` — The number of swap pages the system has brought in and out.

- `intr` — The number of interrupts the system has experienced.

- `btime` — The boot time, measured in the number of seconds since January 1, 1970, otherwise known as the **epoch**.

4.2.28 `/proc/swaps`

This file measures swap space and its utilization. For a system with only one swap partition, the output of /proc/swap may look similar to this:

```
Filename    Type  Size Used Priority
/dev/hda6                  partition 136512 20024 -1
```

While some of this information can be found in other /proc files, swap provides for a very quick snapshot of every swap filename, type of swap space, and total and used sizes (in kilobytes). The priority column is useful when multiple swap files are in use, and some of them are preferred over others, such as if they are on faster hard disks. The lower the priority, the more likely the swap file will be used.

4.2.29 `/proc/uptime`

This file contains information about how long the system has on since its last restart. The output of /proc/uptime is quite minimal:

```
350735.47 234388.90
```

The first number tells you the total number of seconds the system has been up. The second number tells you how much of that time, also in seconds, the machine has spent idle.

4.2.30 `/proc/version`

This files tells you the versions of the Linux kernel and gcc, as well as the version of Red Hat Linux installed on the system:

```
Linux version 2.4.2-2 (root@porky.devel.redhat.com) (gcc version 2.96 20000731
(Red Hat Linux 7.1 2.96-79)) #1 Sun Apr 8 20:41:30 EDT 2001
```

This information is used for a variety of purposes, including providing the version data at the standard login prompt.

4.3 Directories in `/proc`

Common groups of information concerning the kernel is grouped into directories and sub-directories within `/proc`.

4.3.1 Process Directories

Every `/proc` directory contains quite a few directories named with a number. A listing of them starts off like this:

```
[root@bleach /]# ls -l /proc
total 0
dr-xr-xr-x    3 root     root           0 May  8 10:56 1
dr-xr-xr-x    3 truk     truk           0 May  8 10:56 1038
dr-xr-xr-x    3 root     root           0 May  8 10:56 518
dr-xr-xr-x    3 rpc      rpc            0 May  8 10:56 532
dr-xr-xr-x    3 rpcuser  rpcuser        0 May  8 10:56 547
dr-xr-xr-x    3 root     root           0 May  8 10:56 7
dr-xr-xr-x    3 daemon   daemon         0 May  8 10:56 708
dr-xr-xr-x    3 root     root           0 May  8 10:56 8
dr-xr-xr-x    3 xfs      xfs            0 May  8 10:56 880
[root@bleach /]#
```

These directories are called **process directories**, as they refer to a process's ID and contain information specific to that process. The owner and group of each process directory is set to the user running the process. When the process is terminated, its `/proc` process directory vanishes. However, while the process is running, a great deal of information specific to that process is contained in the process directory's various files.

Each of the process directories contains the following files:

- `cmdline` — Contains the command line arguments that started the process. The output of `cmdline` file for the `sshd` process looks like this:

      /usr/sbin/sshd

- `cpu` — Provides specific information about the utilization of each of the system's CPUs. A process running on a dual CPU system produces output similar to this:

      cpu 11 3
      cpu0 0 0
      cpu1 11 3

- `cwd` — A link to the current working directory for the process.

- `environ` — Gives a list of the environment variables for the process. The environment variable is given in all upper-case characters, and the value is in lower-case characters.

- `exe` — A link to the executable of this process.

- `fd` — A directory containing all of the file descriptors for a particular process. These are given in numbered links:

```
[root@bleach /]# ls -l /proc/14466/fd
total 0
lrwx------    1 root     root            64 May  8 11:31 0 -> /dev/null
lrwx------    1 root     root            64 May  8 11:31 1 -> /dev/null
lrwx------    1 root     root            64 May  8 11:31 2 -> /dev/null
lrwx------    1 root     root            64 May  8 11:31 3 -> /dev/ptmx
lrwx------    1 root     root            64 May  8 11:31 4 -> socket:[7774817]
lrwx------    1 root     root            64 May  8 11:31 5 -> /dev/ptmx
lrwx------    1 root     root            64 May  8 11:31 6 -> socket:[7774829]
lrwx------    1 root     root            64 May  8 11:31 7 -> /dev/ptmx
[root@bleach /]#
```

- `maps` — Contains memory maps to the various executables and library files associated with this process. This file can be rather long, depending upon the complexity of the process, but sample output from the `sshd` process begins like this:

```
08048000-08080000 r-xp 00000000 03:05 226209      /usr/sbin/sshd
08080000-08082000 rw-p 00037000 03:05 226209      /usr/sbin/sshd
08082000-080c3000 rwxp 00000000 00:00 0
40000000-40016000 r-xp 00000000 03:05 304721      /lib/ld-2.2.2.so
40016000-40017000 rw-p 00015000 03:05 304721      /lib/ld-2.2.2.so
40017000-40018000 rw-p 00000000 00:00 0
40019000-4001b000 r-xp 00000000 03:05 144433      /lib/security/pam_stack.so
4001b000-4001c000 rw-p 00001000 03:05 144433      /lib/security/pam_stack.so
```

- `mem` — The memory held by the process.

- `root` — A link to the root directory of the process.

- `stat` — A status of the process.

- `statm` — A status of the memory in use by the process. Sample `statm` files look like this:

```
140 72 63 22 0 50 22
```

The seven columns relate to different memory statistics for the process. In order of how they are displayed, from right to left, they report different aspects of the memory used:

1. Total program size, in kilobytes

2. Size of memory portions, in kilobytes

3.   Number of pages that are shared

4.   Number of pages are code

5.   Number of pages of data/stack

6.   Number of pages of library

7.   Number of dirty pages

• `status` — Provides the status of the process in a form that is much more readable than `stat` or `statm`. Sample output for `sshd` looks similar to this:

```
Name: sshd
State: S (sleeping)
Pid: 14466
PPid: 723
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 32
Groups:
VmSize:      3596 kB
VmLck:          0 kB
VmRSS:        288 kB
VmData:       552 kB
VmStk:         28 kB
VmExe:        224 kB
VmLib:       2596 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000001000
SigCgt: 0000000000012000
CapInh: 0000000000000000
CapPrm: 00000000fffffeff
CapEff: 00000000fffffeff
```

Other than the process's name and ID, the state (such as `S (sleeping)` or `R (running)` and user/group ID running the process is available, as well as much more detailed data regarding memory usage.

/proc/self

The `/proc/self` directory is a link to the currently running process. This allows a process to look at itself without having to know its process ID.

Within a shell environment, a listing of the `/proc/self` directory produces the same contents as listing the process directory for that process.

4.3.2 `/proc/bus`

This directory contains information specific to the various busses available on the system. So, for example, on a standard system containing ISA, PCI, and USB busses, current data on each of these busses is available in its directory under /proc/bus.

The contents of the sub-directories and files available varies greatly on the precise configuration of your system. However, each of the directories for each of the bus types contains at least one directory for each bus of that type. These individual bus directories, usually signified with numbers, such as 00, contains binary files that refer to the various devices available on that bus.

So, for example, a system with a USB bus but no USB devices connected to it has a /proc/bus/usb directory containing several files:

```
[root@bleach /]# ls -l /proc/bus/usb
total 0
dr-xr-xr-x    1 root      root                0 May  3 16:25 001
-r--r--r--    1 root      root                0 May  3 16:25 devices
-r--r--r--    1 root      root                0 May  3 16:25 drivers
[root@thoth /]# ls -l /proc/bus/usb/001
total 1
-rw-r--r--    1 root      root               18 May  3 16:25 001
[root@bleach /]#
```

The /proc/bus/usb directory contains files that track the various devices on any USB busses, as well as the drivers required to utilize them. The 001 directory contains all devices on the first (and only) USB bus. By looking at the contents of the devices file, we can see that it is the USB root hub on the motherboard:

```
[root@bleach /]# cat /proc/bus/usb/devices
T:  Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#=  1 Spd=12  MxCh= 2
B:  Alloc=  0/900 us ( 0%), #Int=  0, #Iso=  0
D:  Ver= 1.00 Cls=09(hub ) Sub=00 Prot=00 MxPS= 8 #Cfgs=  1
P:  Vendor=0000 ProdID=0000 Rev= 0.00
S:  Product=USB UHCI Root Hub
S:  SerialNumber=d400
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr=  0mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   8 Ivl=255ms
[root@bleach /]#
```

4.3.3 `/proc/driver`

This directory contains information for specific drivers in use by the kernel.

A common file found here is `rtc`, which provides output from the driver for the system's **Real Time Clock (RTC)**, the device that keeps the time while the system is switched off. Sample output from `/proc/driver/rtc` looks like this:

```
rtc_time : 18:06:33
rtc_date : 2001-05-08
rtc_epoch : 1900
alarm  : 02:01:19
DST_enable : no
BCD  : yes
24hr  : yes
square_wave : no
alarm_IRQ : no
update_IRQ : no
periodic_IRQ : no
periodic_freq : 1024
batt_status : okay
```

For more information about the RTC, review `/usr/src/linux-2.4/Documentation/rtc.txt`.

4.3.4 `/proc/fs`

This directory contains specific filesystem, file handle, inode, dentry and quota information. This information is actually located in `/proc/sys/fs`.

4.3.5 `/proc/ide`

This directory holds an assorted array of information about IDE devices on the system. Each IDE channel is represented as a separate directory, such as `/proc/ide/ide0` and `/proc/ide/ide1`. In addition, a `drivers` file is also available, providing the version number of the various drivers used on the IDE channels:

```
ide-cdrom version 4.59
ide-floppy version 0.97
ide-disk version 1.10
```

Many chipsets also provide an informational file in this directory that gives additional data concerning the drives connected through the various channels. For example, a generic Intel PIIX4 Ultra 33 chipset produces a `/proc/ide/piix` that will tell you whether DMA or UDMA is enabled for the devices on the IDE channels:

```
                                Intel PIIX4 Ultra 33 Chipset.
-------------- Primary Channel --------------- Secondary Channel -------------
              enabled                          enabled
-------------- drive0 --------- drive1 ------- drive0 ---------- drive1 ------
```

```
DMA enabled:    yes             no              yes             no
UDMA enabled:   yes             no              no              no
UDMA enabled:   2               X               X               X
UDMA
DMA
PIO
```

Navigating into the directory for an IDE channel, such as `ide0` for the first channel, provides addi-
tional information. The `channel` file provides the channel number, while the `model` tells you the
bus type for the channel (such as `pci`).

Device Directories

Some of the most useful data can be found in the device directories within the channel directory. Each
device, such as a hard drive or CD-ROM, on that channel will have its own directory containing its
own collection of information and statistics. The contents of these directories vary according to the
type of device connected. Some of the more useful files common to different devices include:

- `cache` — The device's cache.

- `capacity` — The capacity of the device, in 512 byte blocks.

- `driver` — The driver and version used to control the device.

- `geometry` — The physical and logical geometry of the device.

- `media` — The type of device, such as a `disk`.

- `model` — The model name or number of the device.

- `settings` — A collection of current parameters of the device. This file usually contains quite a
  bit of useful, technical information. A sample `settings` file for a standard IDE hard disk looks
  similar to this:

```
name                value       min     max         mode
----                -----       ---     ---         ----
bios_cyl            784         0       65535       rw
bios_head           255         0       255         rw
bios_sect           63          0       63          rw
breada_readahead    4           0       127         rw
bswap               0           0       1           r
current_speed       66          0       69          rw
file_readahead      0           0       2097151     rw
ide_scsi            0           0       1           rw
init_speed          66          0       69          rw
io_32bit            0           0       3           rw
keepsettings        0           0       1           rw
lun                 0           0       7           rw
```

max_kb_per_request	64	1	127	rw
multcount	8	0	8	rw
nice1	1	0	1	rw
nowerr	0	0	1	rw
number	0	0	3	rw
pio_mode	write-only	0	255	w
slow	0	0	1	rw
unmaskirq	0	0	1	rw
using_dma	1	0	1	rw

4.3.6 `/proc/irq`

This directory is used to set IRQ to CPU affinity, which allows you to connect a particular IRQ to only one CPU. Alternatively, you can exclude a CPU from handling any IRQs.

Each IRQ has its own directory, allowing for each IRQ to be configured different from any other. The `/proc/irq/prof_cpu_mask` file is a bitmask that contains the default values for the `smp_affinity` file in the IRQ directory. The values in `smp_affinity` specify which CPUs handle that particular IRQ.

More information is available in the `/usr/src/linux-2.4/Documentation/filesystems/proc.txt` file.

4.3.7 `/proc/net`

This directory provides a comprehensive look at various networking parameters and statistics. Each of the files covers a specific range of information related to networking on the system:

- `arp` — Contains the kernel's ARP table. This file is particularly useful for connecting hardware address to an IP address on a system.

- `atm` — A directory containing files with various **Asynchronous Transfer Mode (ATM)** settings and statistics. This directory is primarily used with ATM networking and ADSL cards.

- `dev` — Lists the various network devices configured on the system, complete with transmit and receive statistics. This file will quickly tell you the number of bytes each interface has sent and received, the number of packets inbound and outbound, the number of errors seen, the number of packets dropped, and much more.

- `dev_mcast` — Displays the various Layer2 multicast groups each device is listening to.

- `igmp` — Lists the IP multicast addresses which this system joined.

- `ip_fwchains` — Reveals any current firewall chains.

- `ip_fwnames` — Lists all firewall chain names. By default, with no firewall chains configured, the three original chains are given:

```
input ACCEPT 1 0 93537 0 12626171
forward ACCEPT 1 0 0 0 0
output ACCEPT 1 0 14270 0 3796697
```

- `ip_masquerade` — Provides a table of masquerading information.

- `ip_mr_cache` — List of the multicast routing cache.

- `ip_mr_vif` — List of multicast virtual interfaces.

- `netstat` — Contains a broad yet detailed collection of networking statistics, including TCP timeouts, SYN cookies sent and received, and much more.

- `psched` — List of global packet scheduler parameters.

- `raw` — List of raw device statistics.

- `route` — Displays the kernel's routing table.

- `rt_cache` — Contains the current routing cache.

- `snmp` — List of Simple Network Management Protocol (SNMP) data for various networking protocols in use.

- `sockstat` — Provides socket statistics. For example:

  ```
  sockets: used 105
  TCP: inuse 7 orphan 0 tw 0 alloc 7 mem 1
  UDP: inuse 3
  RAW: inuse 0
  FRAG: inuse 0 memory 0
  ```

- `tcp` — Contains detailed TCP socket information.

- `tr_rif` — The token ring RIF routing table.

- `udp` — Contains detailed UDP socket information.

- `unix` — Lists UNIX domain sockets currently in use.

- `wireless` — Lists wireless interface data.

4.3.8 `/proc/scsi`

In the same way the `/proc/ide` directory only exists if an IDE controller is connected to the system, the `/proc/scsi` directory is only available if you have a SCSI host adapter.

The primary file here is `/proc/scsi/scsi`, which contains a list of every recognized SCSI device. For example, if a system contains a SCSI CD-ROM, tape drive, hard drives, and RAID controller, this file will look similar to this:

```
Attached devices:
Host: scsi1 Channel: 00 Id: 05 Lun: 00
  Vendor: NEC       Model: CD-ROM DRIVE:466 Rev: 1.06
  Type:   CD-ROM                           ANSI SCSI revision: 02
Host: scsi1 Channel: 00 Id: 06 Lun: 00
  Vendor: ARCHIVE  Model: Python 04106-XXX Rev: 7350
  Type:   Sequential-Access               ANSI SCSI revision: 02
Host: scsi2 Channel: 00 Id: 06 Lun: 00
  Vendor: DELL     Model: 1x6 U2W SCSI BP  Rev: 5.35
  Type:   Processor                        ANSI SCSI revision: 02
Host: scsi2 Channel: 02 Id: 00 Lun: 00
  Vendor: MegaRAID Model: LD0 RAID5 34556R Rev: 1.01
  Type:   Direct-Access                    ANSI SCSI revision: 02
```

From this listing, the type of devices, as well as the model name, vendor, and SCSI channel/ID data is available.

In addition, each SCSI driver used by the system has its own directory in `/proc/scsi`, which contains files specific to each SCSI controller that uses that driver. So, for the example system above, `aic7xxx` and `megaraid` directories are present, as those two drivers are being utilized. The files in each of the directories typically contain IO address range, IRQ, and statistics for the particular SCSI controller that utilizes that driver. While each controller can report a different type and amount of information, expect the output from most of these files to be very helpful and easy to read. The Adaptec AIC-7880 Ultra SCSI host adapter's file in our example system produces the following output:

```
Adaptec AIC7xxx driver version: 5.1.20/3.2.4
Compile Options:
  TCQ Enabled By Default : Disabled
  AIC7XXX_PROC_STATS     : Enabled
  AIC7XXX_RESET_DELAY    : 5

Adapter Configuration:
          SCSI Adapter: Adaptec AIC-7880 Ultra SCSI host adapter
                        Ultra Narrow Controller
    PCI MMAPed I/O Base: 0xfcffe000
 Adapter SEEPROM Config: SEEPROM found and used.
     Adaptec SCSI BIOS: Enabled
                   IRQ: 30
                  SCBs: Active 0, Max Active 1,
                        Allocated 15, HW 16, Page 255
            Interrupts: 33726
      BIOS Control Word: 0x18a6
   Adapter Control Word: 0x1c5f
   Extended Translation: Enabled
Disconnect Enable Flags: 0x00ff
```

```
      Ultra Enable Flags: 0x0020
 Tag Queue Enable Flags: 0x0000
Ordered Queue Tag Flags: 0x0000
Default Tag Queue Depth: 8
    Tagged Queue By Device array for aic7xxx host instance 1:
      {255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255}
    Actual queue depth per device for aic7xxx host instance 1:
      {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}

Statistics:

(scsi1:0:5:0)
  Device using Narrow/Sync transfers at 20.0 MByte/sec, offset 15
  Transinfo settings: current(12/15/0/0), goal(12/15/0/0), user(12/15/0/0)
  Total transfers 0 (0 reads and 0 writes)
              < 2K       2K+      4K+      8K+     16K+     32K+     64K+    128K+
    Reads:       0         0        0        0        0        0        0        0
    Writes:      0         0        0        0        0        0        0        0


(scsi1:0:6:0)
  Device using Narrow/Sync transfers at 10.0 MByte/sec, offset 15
  Transinfo settings: current(25/15/0/0), goal(12/15/0/0), user(12/15/0/0)
  Total transfers 132 (0 reads and 132 writes)
              < 2K       2K+      4K+      8K+     16K+     32K+     64K+    128K+
    Reads:       0         0        0        0        0        0        0        0
    Writes:      0         0        0        1      131        0        0        0
```

From this screen, you can see the transfer speed to the various SCSI devices connected to the controller based on channel ID, as well as detailed statistics concerning the amount and sizes of files read or written by that device. By referring to the `/proc/scsi/scsi` file, we can see that this controller is communicating with the CD-ROM at 20 megabits per second, while the tape drive is only connected at 10 megabits per second.

4.3.9 `/proc/sys`

This directory is special and different from the others in `/proc`, as it not only provides a lot of information about the system but also allows you to make configuration changes to a running kernel.

WARNING

Never attempt to tweak your kernel's settings on a production system using the various files in the /proc/sys directory. Occasionally, changing a setting may render the kernel unstable, requiring a reboot of the system. As this would obviously disrupt any users currently using the system, use a similar development system to try out changes before utilizing them on any production machines.

Before attempting to change a value in /proc/sys, be sure you know the valid options for that file and the expected outcome.

A good way to determine if a particular file can configured or is only designed to provide information is to list it. If the file is writable, you may use it to configure the kernel in a certain way. For example, a partial listing of /proc/sys/fs looks like this:

```
[root@bleach /]# ls -l /proc/sys/fs
<snip>
-r--r--r--    1 root      root                  0 May 10 16:14 dentry-state
-rw-r--r--    1 root      root                  0 May 10 16:14 dir-notify-enable
-r--r--r--    1 root      root                  0 May 10 16:14 dquot-nr
-rw-r--r--    1 root      root                  0 May 10 16:14 file-max
-r--r--r--    1 root      root                  0 May 10 16:14 file-nr
<snip>
[root@bleach /]#
```

In this listing, the files dir-notify-enable and file-max can be written to and, therefore, can be used to configure the kernel. The other files only provide feedback on the kernel's current settings.

Changing a value within a /proc/sys file is done by echoing the new value into the file. For example, to enable the System Request Key on a running kernel, type the command:

```
[root@bleach /]# echo "1" > /proc/sys/kernel/sysrq
```

This will change the sysrq file's value from 0 to 1. By default, the System Request Key is enabled in the Red Hat Linux kernel, but the 0 value is passed to /proc/sys/kernel/sysrq at boot time by rc.sysinit calling sysctl to pass the /etc/sysctl.conf settings to the kernel via the /proc directory. This turns off the System Request Key feature, which may cause a security problem for an unattended console. However, it does permit the root user to enable the feature at any time on a running kernel, if necessary.

The purpose of the System Request Key is to allow you to immediately instruct the kernel to do a number of important activities by using a simple key combination, such as immediately shutting down

or restarting a system, syncing all mounted filesystems, or dumping important information to your console. This feature is most useful when using a development kernel or if you are experiencing system freezes. For more information on the System Request Key, refer to `/usr/src/linux-2.4/Documentation/sysrq.txt`.

A few `/proc/sys` configuration files contain more than one value. In order to correctly send new values to them, place a space character between each value passed with the `echo` command, such as is done in this example:

```
[root@bleach /]# echo "4 2 45" > /proc/sys/kernel/acct
```

Note

Any configuration changes you make by `echo`ing values into `/proc/sys` files will disappear when the system is restarted. To make your configuration changes take effect at the time the system is booted, see Section 4.4, *Using `sysctl`*.

The `/proc/sys` directory contains several different directories that control different aspects of a running kernel.

/proc/sys/dev

This directory provides parameters for particular devices on the system. Most systems have at least two directories, `cdrom` and `raid`, but customized kernels can have others, such as `parport`, which provides the ability to share one parallel port between multiple device drivers.

The `cdrom` directory contains a file called `info`, which reveals a number of important CD-ROM parameters:

```
CD-ROM information, Id: cdrom.c 3.12 2000/10/18

drive name:  hdc
drive speed:  32
drive # of slots: 1
Can close tray:  1
Can open tray:  1
Can lock tray:  1
Can change speed: 1
Can select disk: 0
Can read multisession: 1
Can read MCN:  1
Reports media changed: 1
Can play audio:  1
```

```
Can write CD-R:   0
Can write CD-RW: 0
Can read DVD:    0
Can write DVD-R: 0
Can write DVD-RAM: 0
```

This file can be quickly scanned to discover the qualities of an unknown CD-ROM, at least in the eyes of the kernel. If multiple CD-ROMs are available on a system, each device is given its own column of information.

Various files in /proc/sys/dev/cdrom, such as `autoclose` and `checkmedia`, can be used to control the system's CD-ROM. Simply `echo` a `1` to the correct file to turn on the feature or a `0` to disable a feature.

If RAID support is compiled into the kernel, a /proc/sys/dev/raid directory will be available with at least two files in it: `speed_limit_min` and `speed_limit_max`. These settings come into play in order to throttle or accelerate the speed the RAID device is utilized for particularly I/O intensive tasks, such as resyncing the disks.

/proc/sys/fs

This directory contains an array of options and information concerning various aspects of the filesystem, including quota, file handle, inode, and dentry information.

The `binfmt_misc` directory is used to provide kernel support for miscellaneous binary formats.

The important files in /proc/sys/fs include:

*   `dentry-state` — Provides the status of the directory cache. The file looks similar to this:

    ```
    57411 52939 45 0 0 0
    ```

    The first number reveals the total number of directory cache entries, while the second number displays the number of unused entries. The third number tells the number of seconds between when a directory has been freed and when it can be reclaimed, and the fourth measures the pages currently requested by the system. The last two numbers are not used and currently display only zeros.

*   `dquot-nr` — Shows the maximum number of cached disk quota entries.

*   `file-max` — Allows you to change the maximum number of file handles that the kernel will allocate. Raising the value in this file can resolve errors caused by a lack of available file handles.

*   `file-nr` — Displays the number of allocated file handles, used file handles, and the maximum number of file handles, in that order.

*   `overflowgid` and `overflowuid` — Defines the fixed group ID and user ID, respectively, for use with filesystems that only support 16-bit group and user IDs.

- `super-max` — Controls the maximum number of superblocks available.
- `super-nr` — Displays the current number of superblocks in use.

`/proc/sys/kernel`

This directory contains a variety of different configuration files that directly affect the operation of the kernel. Some of the most important files include:

- `acct` — Controls the suspension of process accounting based on the percentage of free space available on the filesystem containing the log. By default, the file looks like this:

      4 2 30

  The second value sets the threshold percentage of free space when logging will be suspended, while the first value dictates the percentage of free space required for logging to resume. The third value sets the interval in seconds that the kernel polls the filesystem to see if logging should be suspended or resumed.

- `cap-bound` — Controls the **capability bounding** settings, which provide a list of capabilities that any process on the system can do. If a capability is not listed here, then no process, no matter how privileged, can do it. The primary idea is to make the system more secure by ensuring that certain things cannot happen, at least beyond a certain point in the boot process.

  The various values that are possible here are beyond the scope of this manual, so consult the kernel documentation for more information.

- `ctrl-alt-del` — Controls whether [Ctrl]-[Alt]-[Delete] will gracefully restart the computer using `init` (value `0`) or force an immediate reboot without syncing the dirty buffers to disk (value `1`).

- `domainname` — Allows you to configure the system's domain name, such as `domain.com`.

- `hostname` — Allows you to configure the system's host name, such as `host.domain.com`.

- `hotplug` — Configures the utility to be used when a configuration change is detected by the system. This is primarily used with USB and Cardbus PCI. The default value of `/sbin/hotplug` should not be changed unless you are testing a new program to fulfill this role.

- `modprobe` — Sets the location of the program to be used to load kernel modules when necessary. The default value of `/sbin/modprobe` signifies that `kmod` will call it to actually load the module when a kernel thread calls `kmod`.

- `msgmax` — Sets the maximum size of any message sent from one process to another, which is set to `8192` bytes by default. You should be careful about raising this value, as queued messages between processes are stored in non-swappable kernel memory, and any increase in `msgmax` would increase RAM requirements for the system.

- `msgmnb` — Sets the maximum number of bytes in a single message queue. By default, `16384`.

- `msgmni` — Sets the maximum number of message queue identifiers. By default, `16`.

- `osrelease` — Lists the Linux kernel release number. This file can only be altered by changing the kernel source and recompiling.

- `ostype` — Displays the type of operating system. By default, this file is set to `Linux`, and this value can only be changed by changing the kernel source and recompiling.

- `overflowgid` and `overflowuid` — Defines the fixed group ID and user ID, respectively, for use with system calls on architectures that only support 16-bit group and user IDs.

- `panic` — Defines the number of seconds the kernel will postpone rebooting the system when a kernel panic is experienced. By default, the value is set to `0`, which disables automatic rebooting after a panic.

- `printk` — This file control a variety of settings related to printing or logging error messages. Each error message reported by the kernel has a **loglevel** associated with it that defines the importance of the message. The loglevel values break down in this order:

  - `0` — Kernel emergency. The system is unusable.

  - `1` — Kernel alert. Action must be taken immediately.

  - `2` — Condition of the kernel is considered critical.

  - `3` — General kernel error condition.

  - `4` — General kernel warning condition.

  - `5` — Kernel notice of a normal but significant condition.

  - `6` — Kernel informational message.

  - `7` — Kernel debug-level messages.

  Four values are found in the `printk` file:

  ```
  6   4   1   7
  ```

  Each of these values defines a different rule for dealing with error messages. The first value, called the **console loglevel**, defines the lowest priority of messages that will be printed to the console. (Note that, the lower the priority, the higher the loglevel number.) The second value sets the default loglevel for messages without an explicit loglevel attached to them. The third value sets the lowest possible loglevel configuration for the console loglevel. The last value sets the default value for the console loglevel.

- `rtsig-max` — Configures the maximum number of POSIX realtime signals that the system may have queued at any one time. The default value is `1024`.

- `rtsig-nr` — The current number of POSIX realtime signals queued by the kernel.

- `sem` — This file configures semaphore settings within the kernel. A **semaphore** is a System V IPC object that is used to control utilization of a particular process.

- `shmall` — Sets the total amount of shared memory that can be used at one time on the system, in bytes. By default, this value is `2097152`.

- `shmmax` — Sets the largest shared memory segment size allowed by the kernel, in bytes. By default, this value is `33554432`. The kernel supports much larger values than this, however.

- `shmmni` — Sets the maximum number of shared memory segments for the whole system. By default, this value is `4096`

- `sysrq` — Activates the System Request Key, if this value is set to anything other than the default of `0`.

- `threads-max` — Sets the maximum number of threads to be used by the kernel, with a default value of `4095`.

- `version` — Displays the date and time the kernel was last compiled. The first field in this file, such as `#3`, relates to the number of times a kernel was built from the source base.

The `random` directory stores a number of values related to generating random numbers for the kernel.

`/proc/sys/net`

This directory contains assorted directories of its own concerning various networking topics, including assorted protocols and centers of emphasis. Various configurations at the time of kernel compilation make available different directories here, such as `appletalk`, `ethernet`, `ipv4`, `ipx`, and `ipv6`. Within these directories, you can adjust the assorted networking values for that configuration on a running system.

Given the wide variety of possible networking options available with Linux and the great amount of space required to discuss them, only the most common `/proc/sys/net` directories will be discussed.

The `core` directory contains a variety of settings that control the interaction between the kernel and networking layers. The most important files there are:

- `message_burst` — Tenths of seconds required to write a new warning message. This is used to prevent Denial of Service (DoS) attacks, and the default setting is `50`.

- `message_cost` — Also used to prevent DoS attacks by placing a cost on every warning message. The higher the value of this file (default of 5), the more likely the warning message will be ignored.

The idea is that an attacker could bombard your system with requests that generate errors and fill up your logs or require all of your system's resources to handle error logging. The settings in `message_burst` and `message_cost` are designed to be modified based on your system's acceptable risk versus the need for comprehensive logging.

- `netdev_max_backlog` — Sets the maximum number of packets allowed to queue when a particular interface receives packets faster than the kernel can process them. The default value for this file is `300`.

- `optmem_max` — Configures the maximum ancillary buffer size allowed per socket.

- `rmem_default` — Sets the receive socket buffer's default size in bytes.

- `rmem_max` — Sets the receive socket buffer's maximum size in bytes.

- `wmem_default` — Sets the send socket buffer's default size in bytes.

- `wmem_max` — Sets the send socket buffer's maximum size in bytes.

Given the widespread use of IP networks with Linux, a look at the most important files in the `ipv4` reveals additional, powerful, networking settings. Many of these settings, used in the proper conjunction with one another, are very useful in preventing attacks on your system.

CAUTION

However, be sure you know what you are doing, as well as what you expect to happen, before changing any of these settings. In addition, you should make any changes locally, as an erroneous change may affect your remote connectivity to the system.

Here are some of the most important files in the `ipv4` directory:

- `icmp_destunreach_rate`, `icmp_echoreply_rate`, `icmp_paramprob_rate` and `icmp_timeexeed_rate` — Sets the maximum ICMP send packet rate, in hundredths of a second on Intel systems, to hosts under different conditions. A setting of `0` removes any delay and is not a good idea.

- `icmp_echo_ignore_all` and `icmp_echo_ignore_broadcasts` — Allows the kernel to ignore ICMP ECHO packets from every host or only those originating from broadcast and multicast addresses, respectively. A `0` allows the kernel to respond, while a `1` ignores the packets.

- `ip_default_ttl` — Sets the default **Time To Live (TTL)**, which limits the number of hops a packet may make before reaching its destination. Increasing this value can diminish system performance.

- `ip_forward` — Permits interfaces on the system to forward packets to one other. By default, this file is set to `0` to disable forwarding, but setting this file to `1` will enable forwarding.

- `ip_local_port_range` — Specifies the range of ports to be used by TCP or UDP when a local port is needed. The first number is the lowest port to be used, and the second number specifies the highest port. Any systems that expect to require more ports than the default 1024 to 4999 should use the 32768 to 61000 range in this file.

- `tcp_syn_retries` — Provides a limit on the number of times your system will re-transmit a SYN packet when attempting to make a connection.

- `tcp_retries1` — Sets the number of permitted re-transmissions attempting to answer an in-coming connection. Default of 3.

- `tcp_retries2` — Sets the number of permitted re-transmissions of TCP packets. Default of 15.

For a complete list of files and options available, see `/usr/src/linux-2.4/Documenta-tion/networking/ip-sysctl.txt`.

A number of other directories exist within the `/proc/sys/net/ipv4` directory cover specific topics. The `conf` directory allows each of the systems interfaces to be configured in different ways, including the use of a default settings for unconfigured devices (in the `default` subdirectory) and settings that override all special configurations (in the `all` subdirectory).

In order to control connections between direct neighbors, meaning any other system directly connected to your system, the `neigh` directory allows special configurations for each interface. This would allow you to treat systems differently that you might trust more due to their relatively proximity or relation to your system. At the same time, it also makes it easy to put strict rules in place for systems several hops away.

Routing over IPV4 also has its own directory, `route`. Unlike `conf` and `neigh`, the `route` directory contains specifications that apply to routing with any interfaces on the system. Many of these settings, such as `max_size`, `max_delay`, and `min_delay`, relate to controlling the size of the routing cache. To clear the routing cache, simply write any value to the `flush` file.

Additional information about these directories and the possible values for their configuration files can be found in `/usr/src/linux-2.4/Documentation/filesystems/proc.txt`.

`/proc/sys/vm`

This directory facilitates the configuration of the Linux kernel's virtual memory (VM) subsystem. The kernel makes extensive and intelligent use of virtual memory, which is commonly called swap space.

The following files are commonly found in the `/proc/sys/vm` directory:

- `bdflush` — Sets various values related to the `bdflush` kernel daemon.

- `buffermem` — Allows you to control the percentage amount of total system memory to be used for buffer memory. Typical output for this file looks like this:

  ```
  2       10        60
  ```

  The first and last values set the minimum and maximum percentage of memory to be used as buffer memory, respectively. The middle value sets the percentage of system memory dedicated to buffer memory where the memory management subsystem will begin to clear buffer cache more than other kinds of memory to compensate for a general lack of free memory.

- `freepages` — Displays various values related to free pages of system memory. This file looks similar to this:

  ```
  223 446 669
  ```

  The first value shows the minimum number of free pages permitted before the kernel takes over control of allocating additional memory. The second value gives the number of free pages before the kernel begins swapping aggressively to preserve performance. The third value is the number of free pages that the system attempts to keep available at all times.

- `kswapd` — Sets various values concerned with the kernel swap-out daemon, `kswapd`. This file has three values:

  ```
  512 32 8
  ```

  The first value sets the maximum number of pages that `kswapd` will attempt to free in a single attempt. The larger this number, the more aggressively the kernel can move to free pages. The second value sets the minimum number of times that `kswapd` attempts to free a page. The third value sets the number of pages `kswapd` attempts to write in a single attempt. Proper tuning of this final value can improve performance on a system using a lot of swap space by telling the kernel to write pages in large chunks, minimizing the number of disk seeks.

- `max_map_count` — Configures the maximum number of memory map areas a process may have. In most cases, the default value of `65536` is appropriate.

- `overcommit_memory` — Contains a value that, when set to something other than the default of `0`, allows the kernel to skip a standard check to see if there is enough memory before allocating it.

- `pagecache` — Controls the amount of memory used by the page cache. The values in `page-cache` are percentages, and they work in a similar way as `buffermem` to enforce minimums and maximums of available page cache memory.

- `page-cluster` — Sets the number of pages read in a single attempt. The default value of `4`, which actually relates to 16 pages, is appropriate for most systems.

- `pagetable_cache` — Controls the number of page tables that are cached on a per-processor basis. The first and second values relate to the minimum and maximum number of page tables to set aside, respectively.

Additional information on these various files can be found in `/usr/src/linux-2.4/Documentation/sysctl/vm.txt`.

4.3.10 `/proc/sysvipc`

This directory contain information about System V IPC resources. The files in this directory relate to System V IPC calls for messages (`msg`), semaphores (`sem`), and shared memory (`shm`).

4.3.11 `/proc/tty`

This directory contains information about the available and currently used tty devices on the system. Originally called a teletype device, any character-based data terminals are called **tty devices**.

In Linux, there are three different kinds of tty devices. **Serial devices** are used with serial connections, such as over a modem or using a serial cable. **Virtual terminals** create the common console connection, such as the virtual consoles available when pressing [Alt]-[<F-key>] at the system console. **Pseudo terminals** create a two-way communication that is used by some higher level applications, such as X11.

The `drivers` file is a list of the current tty devices in use:

```
serial              /dev/cua        5  64-127 serial:callout
serial              /dev/ttyS       4  64-127 serial
pty_slave           /dev/pts      136   0-255 pty:slave
pty_master          /dev/ptm      128   0-255 pty:master
pty_slave           /dev/ttyp       3   0-255 pty:slave
pty_master          /dev/pty        2   0-255 pty:master
/dev/vc/0           /dev/vc/0       4       0 system:vtmaster
/dev/ptmx           /dev/ptmx       5       2 system
/dev/console        /dev/console    5       1 system:console
/dev/tty            /dev/tty        5       0 system:/dev/tty
unknown             /dev/vc/%d      4    1-63 console
```

The `/proc/tty/driver/serial` file lists the usage statistics and status of each of the serial tty lines.

In order that tty devices can be used in a similar way as network devices, the Linux kernel will enforce **line discipline** on the device. This allows the driver to place a specific type of header with every block of data transmitted over the device, making it possible for the remote end of the connection to be able to see that block of data as just one in a stream of data blocks. SLIP and PPP are common line disciplines, and each are commonly used to connect systems to one other over a serial link.

Registered line disciplines are stored in the `ldiscs` file, with detailed information available in the `ldisc` directory.

4.4 Using `sysctl`

Setting kernel parameters in the `/proc/sys` directory need not be a manual process or one that required echoing values into a virtual file, hoping they are correct. The `sysctl` command can make viewing, setting, and automating special kernel settings very easy.

To get a quick overview of all settings configurable in the `/proc/sys` directory, type the `sysctl -a` command as root. This will create a large, comprehensive list, a small portion of which looks something like this:

```
net.ipv4.route.min_delay = 2
kernel.sysrq = 0
kernel.sem = 250      32000      32      128
```

This is the same basic information you would see if you viewed each of the files individually. The only difference is the file location. The `/proc/sys/net/ipv4/route/min_delay` is signified by `net.ipv4.route.min_delay`, with the directory slashes replaced by dots and the `proc.sys` portion assumed.

As we have seen in Section 4.3.9, *`/proc/sys`*, you can manually assign values to writable files by echoing the value to the file. The `sysctl` command is able to do the same thing by typing the `sysctl -w <file>="<new-value>"` command. For example, to activate the System Request Key, the following command is required:

```
[root@bleach /]# sysctl -w kernel.sysrq="1"
kernel.sysrq = 1
[root@bleach /]#
```

While quickly setting single values like this in `/proc/sys` is helpful during testing, it does not work as well on a production system, as all `/proc/sys` special settings are lost when the machine is rebooted. To preserve the settings that you like to make permanently to your kernel, add them to the `/etc/sysctl.conf` file.

Every time the system boots, the `/etc/rc.d/rc.sysinit` script is executed by `init`. This script contains a command to execute `sysctl` using `/etc/sysctl.conf` as the values to set. Therefore, any values added to `/etc/sysctl.conf` will take effect right after the system boots with no need to reconfigure and rebuild the kernel to incorporate the change.

4.5 Additional Resources

As the file contents throughout the `/proc` directory provide a look at the state of your system's kernel, which may also be specially configured for a certain purpose, this chapter should not be seen as

comprehensive guide to this information. Rather, you should definitely investigate additional sources of information about `/proc` before tweaking your kernel.

4.5.1 Installed Documentation

Most of the best `/proc` documentation is probably already available on your system.

- `/usr/src/linux-2.4/Documentation/filesystems/proc.txt` — Contains assorted, but limited, information about all aspects of `/proc`.

- `/usr/src/linux-2.4/Documentation/sysrq.txt` — An overview of System Request Key options.

- `/usr/src/linux-2.4/Documentation/sysctl` — A directory containing a variety of `sysctl` tips, including modifying values that concern the kernel (`kernel.txt`), accessing filesystems (`fs.txt`), and virtual memory use (`vm.txt`).

- `/usr/src/linux-2.4/Documentation/networking/ip-sysctl.txt` — A look at various IP networking options and their meanings to the kernel.

- Perhaps the most authoritative information on `/proc` can be found by reading the kernel source code. Make sure the `kernel-source` RPM is installed on your system and look in the `/usr/src/linux-2.4` directory for the source.

4.5.2 Useful Websites

- http://www.linuxhq.com — This site maintain a complete database of source, patches, and documentation for various versions of the Linux kernel.

5   GRUB

Before Red Hat Linux can start up on your system, it must be told to boot by special instructions placed on a **boot loader**, code that exists on your primary hard drive or other media device that knows how to start the Linux kernel.

If you are using an x86 system that only boots Red Hat Linux, and if you only use one version of the Linux kernel, then the exact process used by the boot loader to start Red Hat Linux is not all that important. The Red Hat Linux installation program allows you to quickly and easily configure the boot loader in your hard drive's **Master Boot Record (MBR)** to load the operating system.

However, if you wish to have the choice to boot from multiple Linux kernels or other operating systems, then knowledge of the method used by Red Hat Linux to provide the necessary options to you, as well as an understanding of the boot process and what to change, is essential.

This chapter discusses GRUB, the method Red Hat Linux uses to launch operating systems on x86 systems, and looks at various command and configuration options that allow you to control the boot process.

5.1 Defining GRUB

GNU GRUB (GRand Unified Bootloader) is a program that installs a boot loader to the MBR, which exists at the beginning sectors of a disk. It allows you to place specific instructions in the MBR that loads a GRUB menu or command environment, permitting you to start the operating system of your choice, pass special instructions to kernels when they boot, or discover system parameters (such as available RAM) before booting.

5.1.1 x86 Boot Process

When an x86 machine first starts up, the BIOS of the system checks for particular system parameters, such as the amount of RAM in the machine, the date and time, disk devices present, and the order in which those disk devices should be checked for bootable media. Usually, the BIOS is configured to check the machine's floppy disk drive or CD-ROM drive (or both) before attempting to boot from the hard drive. If no bootable media can be found in those removable devices, the BIOS usually turns to the first few sectors of the first hard drive for instructions on where to find and load an operating system. These first few sectors, commonly called the MBR, start the process of bringing up a pre-selected operating system, a GRUB menu of operating system options, or a GRUB command line interface to execute special options.

The process of loading GRUB, and then the operating system, involves several stages:

1. *Loading the primary boot loader, commonly called Stage 1.*

The primary boot loader must exist in the very small space allocated for the MBR, which is less than 512 bytes. Therefore, the only thing the primary boot loader accomplishes is loading the secondary boot loader, due to the fact that there is not enough space in the MBR for anything else.

2. *Loading the secondary boot loader, commonly called Stage 2.*

    The secondary boot loader actually brings up the advanced functionality that allows you to load a specific operating system. With GRUB, this is the code that allows you to display a menu or type commands.

3. *Loading the operating system, such as the Linux kernel, on a specified partition.*

    Once GRUB has received the correct instructions for the operating system to start, either from its command line or configuration file, it finds the necessary boot file and hands off control of the machine to that operating system.

Note

Some filesystems, as well as filesystem configurations, may require a Stage 1.5 file that essentially bridges the gap between the primary and secondary boot loaders.

For example, if your Stage 2 boot loader file is on a partition using a filesystem that the Stage 1 boot loader cannot access, it is possible to direct the Stage 1 boot loader to load additional instructions from the Stage 1.5 file that allows it to read the Stage 2 boot loader file. For more information, consult the GRUB info pages.

This boot method is called **direct loading**, because instructions to directly load the operating system are used, with no intermediary code between the boot loaders and the operating system's main files (such as the kernel). The boot process used by different operating systems may differ slightly from the above, however. For example, Microsoft's DOS and Windows operating systems tend to completely overwrite anything on the MBR when they are installed without incorporating any of the current MBR's configuration. This destroys any other information stored in the MBR by other operating systems, such as Red Hat Linux. The Microsoft operating systems, as well as various other proprietary operating systems, are loaded using a **chain loading** boot method. With this method, the MBR simply points to the first sector of the partition holding the operating system, where it finds the special files necessary to actually boot that operating system.

GRUB supports both boot methods, allowing you to use it with almost any operating system, most popular filesystems, and almost any hard disk your BIOS can recognize.

5.1.2 GRUB Features

GRUB contains a number of features that make it preferable to other available boot loaders. These are some of the most important:

• *GRUB provides a true command-based, pre-OS environment on x86 machines to allow maximum flexibility in loading operating systems with certain options or gathering information about the system.*

Many non-x86 architectures have employed pre-OS environments for years that allow you to control how the system boots from a command line. While some command features are available with LILO and other x86 boot loaders, GRUB contains a greater number of features.

• *GRUB supports Logical Block Addressing (LBA) mode.*

LBA places the addressing conversion used to find files on the drive in the drive's firmware, and it is used on many IDE and all SCSI hard disks. Before LBA, hard drives could encounter a 1024-cylinder limit, where the BIOS could not find a file after that point, such as a boot loader or kernel files. LBA support allows GRUB to boot operating systems from partitions beyond the 1024-cylinder limit, so long as your BIOS supports LBA mode (most do).

• *GRUB's configuration file is read from the disk every time the system boots, preventing you from having to write over the MBR every time you change the boot options.*

Most boot loaders are not sophisticated enough to read configuration files and use them to set up boot options. For example, to change a LILO boot configuration, such as changing the default operating system to boot, you must change a LILO configuration file and run a command that overwrites the system's MBR with the new configuration data. This is more risky than GRUB's method, because a misconfigured MBR would leave you with an unbootable system. With GRUB, if you misconfigure the configuration file and reboot, it will simply drop you to a command line and allow you to manually type commands that will launch the operating system. The MBR is not touched except to update the Stage 1, Stage 2, or menu configuration file locations, and this is rarely necessary.

Note

When changes are made to the GRUB configuration file, it is not necessary to restart GRUB. Any changes made are automatically detected. If you do restart GRUB, you will be dropped to the command line GRUB shell.

5.1.3 Installing GRUB

If during the Red Hat Linux installation process you did not install GRUB, here is how you can install it and make it your default boot loader.

Note

If you are already using LILO as your boot loader, you do not have to remove it in order to use GRUB. Once installed, GRUB will be the default boot loader for your system.

First, make sure you have the latest GRUB package available. You can also use the GRUB package from the Red Hat Linux installation CD-ROMs.

Next, run the command /sbin/grub-install <*location*> from a shell prompt, where <*location*> is the location where GRUB should be installed, such as /dev/hda1.

When you reboot your system, you will see the GRUB boot loader.

5.2 Terminology

One of the most important things to understand before using GRUB is how the program refers to devices, such as your hard drives, and partitions. This information is very important if you want to configure GRUB to boot multiple operating systems.

5.2.1 Device Names

The first hard drive of a system will be called (hd0) by GRUB. The first partition on that drive is called (hd0,0) by GRUB, and the fifth partition on the second hard drive is called (hd1,4). In general, the naming convention for filesystems when using GRUB breaks down in this way:

    (<*type-of-device*><*bios-device-number*>,<*partition-number*>)

The parentheses and comma are very important in the name. The <*type-of-device*> refers to whether a hard disk (hd) or floppy disk (fd) is being specified.

The <*bios-device-number*> is the number of the device according to your system's BIOS, starting with 0. The primary IDE hard drive is numbered 0, while the secondary IDE hard drive is numbered 1. The ordering is roughly equivalent to the way the Linux kernel arranges the devices by letters, where the a in hda relates to 0, the b in hdb relates to 1, and so on.

Note

Remember that GRUB's numbering system for devices starts at `0`, and not `1`. This is one of the most common mistakes made by new GRUB users.

The `<partition-number>` relates to the number of a specific partition on that device. Like the `<bios-device-number>`, the numbering of partitions starts at `0`. While most partitions are specified by numbers, if your system uses BSD partitions, they are signified by letters, such as `a` or `c`.

GRUB uses the following rules when naming devices and partitions:

- It does not matter if your hard drives are IDE or SCSI. All hard drives start with `hd`. Floppy disks start with `fd`.

- To specify an entire device without respect to its partitions, simply leave off the comma and the partition number. This is important when telling GRUB to configure the MBR for a particular disk. For example, `(hd0)` specifies the first device and `(hd3)` specifies the fourth device.

- It is very important, if you have multiple hard drives, to know their order according to the BIOS. This is rather simple to do if you have only IDE or SCSI drives, but if you have a mix of them, things get a bit more tricky.

5.2.2 File Names

When typing commands to GRUB involving a file, such as a menu list to use when allowing the booting of multiple operating systems, you must include the file immediately after the specifying the device and partition. A sample file specification to an absolute filename is organized like this:

    (<type-of-device><bios-device-number>,<partition-number>)/path/to/file

Most of the time, you will be specifying files by their path on that partition plus the file's name. This is rather straightforward.

You can also specify files to GRUB that do not actually appear in the filesystem, such as a chain loader that appears in the first few blocks of a partition. To specify these files, you have to provide a **blocklist**, which tells GRUB, block by block, where the file is located in the partition. As a file can be comprised of several different sets of blocks, there is a specific way to write blocklists. Each file's section location is described by an offset number of blocks and then a number of blocks from that offset point, and the sections are put together in order, separated by commas.

In other words, consider the following blocklist:

    0+50,100+25,200+1

This blocklist tells GRUB to use a file that starts at the first block on the partition and uses blocks 0 through 49, 99 through 124, and 199.

Knowing how to write blocklists is useful when using GRUB to load operating systems that use chain loading, such as Microsoft Windows. You can leave off the offset number of blocks if you are starting at block 0. As an example, the chain loading file in the first partition of the first hard drive would have the following name:

```
(hd0,0)+1
```

You can also use the chainloader command with a similar blocklist designation at the GRUB command line after setting the correct device and partition as root:

```
chainloader +1
```

5.2.3 GRUB's Root Filesystem

Some users are confused by the use of the term "root filesystem" with GRUB. It is important to remember that GRUB's root filesystem has nothing to do with the Linux root filesystem.

The GRUB root filesystem is the root partition for a particular device. GRUB uses this information to mount the device and load files from it, among other things.

With Red Hat Linux, once GRUB has loaded its root partition that contains the Linux kernel, the kernel command can be executed with the location of the kernel file as an option. Once the Linux kernel boots, it sets its own root filesystem, and that is the one most people associate with Linux. The original GRUB root filesystem and its mounts are forgotten. They only existed to boot the kernel file.

See the notes given for the root and kernel commands in Section 5.4, *Commands* for more information.

5.3 Interfaces

GRUB features three powerful interfaces that provide different levels of functionality. Each of these interfaces allow you to boot operating systems, and you can even move between interfaces within the GRUB environment.

5.3.1 Menu Interface

If GRUB was automatically configured for you by the Red Hat Linux installation program, this is the interface you already know. A menu of operating systems or kernels preconfigured with their own boot commands exists in this interface as a list, by name, after the system first starts. You can use the arrow keys to select an option other than the default selection and press the [Enter] key to boot it. Alternatively, a timeout period may have been set, so that GRUB will start loading the default option.

From the menu interface, you can also press the [e] key to edit the highlighted menu entry's commands or the [c] key to move to a command line interface.

5.3.2 Menu Entry Editor Interface

To get to this interface, you must press the [e] key from the menu interface. The GRUB commands for that entry are displayed here, and you can alter these command lines before booting the operating system by adding ([o] after the current line; [O] before it), editing ([e]), or deleting ([d]) them.

After you have made your changes and the lines look like you want, you can hit the [b] key to execute them and boot the operating system. The [Esc] key forgets your changes and takes you to the standard menu interface. The [c] key will take you to the command line interface.

5.3.3 Command Line Interface

This is the most basic GRUB interface, but it is also the one that gives you the most control. You can type any relevant GRUB commands here followed by the [Enter] key to execute them. This interface features some advanced shell-like features, including [Tab] key completion, based on context, and [Ctrl] key combinations when typing commands, such as [Ctrl]-[a] to move to the beginning of a line, and [Ctrl]-[e] to move to the end of a line. In addition, the arrow, [Home], [End], and [Delete] keys work as they do in the bash shell.

5.3.4 Order of Interface Use

When the GRUB environment starts upon loading the second stage boot loader, it looks for its configuration file. When found, it uses the configuration file to build the menu list of operating systems to load and moves you into the menu interface.

If the configuration file cannot be found, or if the configuration file is unreadable, GRUB will take you to the command line interface to allow you to manually type the commands necessary to boot an operating system.

If the configuration file is not valid, GRUB will print out the error and ask for input. This can be very helpful, because you will be able to see precisely where the problem occurred and fix it in the file. Pressing any key will take you back to the menu interface, where you can edit that menu option and correct the problem based on the error reported by GRUB. If your correction fails, the error is reported and you are allowed to start again.

5.4 Commands

GRUB contains a number of different commands that may be executed interactively in the command line interface. Some of the commands accept options after their name, and these options should be separated from the command and other options on that line by space characters.

The following list gives the most useful commands:

- `boot` — Boots the operating system or chain loader that has been previously specified and loaded.

- `chainloader <file-name>` — Loads the specified file as a chain loader. To grab the file at the first sector of the specified partition, you can use `+1` as the file's name.

- `displaymem` — Displays the current use of memory, based on information from the BIOS. This is useful if you are not sure how much RAM a system has and have yet to boot it.

- `initrd <file-name>` — Allows you to specify an initial RAM disk to use when booting, necessary when the kernel needs certain modules in order to boot properly.

- `install` *<stage-1>* *<install-disk>* *<stage-2>* p *<config-file>* — Installs GRUB to your MBR. This allows the GRUB interfaces to come up when the system is rebooted.

> **WARNING**
>
> **This command will write over anything else in your MBR. If executed, anything you have been using to boot your operating system other than GRUB will be lost.**
>
> **Make sure you know what you are doing before you execute this command.**

This command can be configured in several different ways. However, you must specify a *<stage-1>*, which signifies a device, partition, and file where the first boot loader image can be found, such as `(hd0,0)/grub/stage1`. In addition, you must specify the disk where the stage 1 boot loader should be installed, such as `(hd0)`.

The *<stage-2>* section tells the stage 1 boot loader where the stage 2 boot loader is located, such as `(hd0,0)/grub/stage2`. The `p` option tells the `install` command that a menu configuration file is being specified in the *<config-file>* section, such as `(hd0,0)/grub/grub.conf`.

- `kernel <kernel-file-name> <option-1> <option-N>` — Specifies the kernel file to load from GRUB's root filesystem when using direct loading to boot the operating system. Options can follow the `kernel` command that will be passed to the kernel when it is loaded.

For Red Hat Linux, you may have a line that looks like this:

```
kernel /vmlinuz root=/dev/hda5
```

This line specifies that the `vmlinuz` file is loaded from GRUB's root filesystem, such as `(hd0,0)`. An option is also passed to the kernel specifying that the root filesystem for the Linux

kernel when it loads should be on hda5, the fifth partition on the first IDE hard drive. Multiple options may be placed after this option, if you need them.

- root *<device-and-partition>* — Configures GRUB's root partition to be the particular device and partition, such as (hd0,0), and mounts the partition so that files can be read.

- rootnoverify *<device-and-partition>* — Does the same thing as the root command but does not mount the partition.

Commands other than these are available. Type info grub for a full list of all commands.

5.5 The Menu Configuration File

The configuration file, which is used to create the list in GRUB's menu interface of operating systems to boot, essentially allows the user to select a pre-set group of commands to execute. The commands given in Section 5.4, *Commands* can used, as well as some special commands that may only be used in the configuration file.

5.5.1 Special Configuration File Commands

The following commands can only be used in the GRUB menu configuration file:

- color *<normal-color>* *<selected-color>* — Allows you to set up specific colors to be used in the menu, where two colors are configured as the foreground and background. You can just use the color names, such as red/black. An example line might look like this:

      color red/black green/blue

- default *<title-name>* — The default entry title name that will be loaded if the menu interface times out.

- fallback *<title-name>* — If used, the entry title name to try if first attempt failed.

- hiddenmenu — If used, it prevents the GRUB menu interface from being displayed, loading the default entry when the timeout period expires. The user can see the standard GRUB menu by pressing the [Esc] key.

- password *<password>* — If used, it prevents the user who does not know the password from editing the entries for this menu option.

   Optionally, you can specify an alternate menu configuration file after the *<password>*, so that, if the password is know, GRUB will restart the second stage of the boot loader and use this alternate configuration file to build the menu. If this alternate file is left out of the command, then a user who knows the password is able to edit the current configuration file.

- timeout — If used, sets the amount of time, in seconds, before GRUB loads the entry designated by the default command.

- `splashimage` — Specifies the location of the splash screen image to be used when GRUB boots.

- `title` — Sets a title to be used with a particular group of commands used to load an operating system.

The # character can be used to place comments in the menu configuration file.

5.5.2 Configuration File Structure

The commands to set the global preferences for the GRUB menu interface are placed at the top of the file, followed by the different entries for each of the operating systems to boot.

A very basic GRUB menu configuration file designed to boot either Red Hat Linux or Microsoft Windows 2000 might look like this:

```
default linux
timeout 10
color green/black light-gray/blue

# section to load linux
title linux
root (hd0,1)
kernel /vmlinuz root=/dev/hda5
boot

# section to load Window 2000
title windows
rootnoverify (hd0,0)
chainloader +1
```

This file would tell GRUB to build a menu with Red Hat Linux as the default operating system, set to autoboot it after 10 seconds. Two sections are given, one for each operating system entry, with commands specific to this system's partition table.

Actually configuring a GRUB menu configuration file to boot multiple operating systems is beyond the scope of this chapter. Please consult Section 5.6, *Additional Resources* for the detailed information required to boot the various operating systems possible with GRUB.

5.6 Additional Resources

This chapter is only intended as an introduction to GRUB and its many options. Consult the following resources to discover more about how GRUB works and how to configure it to boot non-Linux operating systems, among other tasks.

5.6.1 Installed Documentation

- `/usr/share/doc/grub-0.90` — `/usr/share/doc/grub-0.90` is the directory where the GRUB docs are placed on the filesystem.

- Type the `man grub` command to view the GRUB man page, which contains a list of options to use when loading the GRUB shell.

- The GRUB info page, accessible when typing the `info grub` command, contains a tutorial, a user reference manual, a programmer reference manual, and an FAQ document.

5.6.2 Useful Websites

- http://www.gnu.org/software/grub — The home page of the GNU GRUB project. This site contains information concerning the state of GRUB development and an FAQ.

- http://www.uruk.org/orig-grub — The original GRUB documentation before the project was handed off to the Free Software Foundation for further development.

- http://www.redhat.com/mirrors/LDP/HOWTO/mini/Multiboot-with-GRUB.html — Investigates various uses for GRUB, including booting operating systems other than Linux.

- http://www.linuxgazette.com/issue64/kohli.html — An introductory article discussing the configuration of GRUB on your system from scratch, including an overview of GRUB command line options.

6   X Servers and Clients

While the heart of Red Hat Linux is the kernel, for workstation users, the X environment is the face of the operating system. The kernel provides the engine for everything that happens, managing processes and resources virtually unseen. However, the day-to-day Linux users spend the majority of their time in a graphical desktop environment, opening applications, resizing windows, and scrolling through text.

This chapter is designed to serve as an introduction to the behind-the-scenes world of XFree86 and how X operates on your machine to provide advanced desktop functionality.

6.1 The Power of X

Linux began as a powerful server-based operating system, excelling at efficiently processing compli-cated programs requiring high CPU utilization and handling requests from hundreds or thousands of clients through network connections. However, because of its open nature and stability, Linux has quickly developed into a popular GUI-based operating system for workstations, in the home or busi-ness.

In the UNIX world, windowing environments have existed for decades, predating many of the main-stream operating systems. UNIX machines utilize the **X Windows System**, which uses a client-server relationship for creating a graphical (GUI) environment for the user. An X server process is started that the X client processes can connect to via a network or local connection. The server process handles the communication with the hardware, such as the video card, monitor, keyboard, and mouse. The X client exists in the userspace, issuing requests to the X server for certain things to be done using the hardware it controls.

On Red Hat Linux systems, the XFree86 server fills the role of the X server. As an open source software project operating on a grand scope with hundreds of developers around the world, XFree86 features rapid development, a wide degree of support for various hardware devices and architectures, and the ability to run on different operating systems and platforms.

Most Red Hat Linux desktop users are unaware of the XFree86 server running on their system. They are much more concerned with the particular desktop environment in which they spend most of their time. The Red Hat Linux installation program does an excellent job of configuring your XFree86 server during the installation process, ensuring that X performs optimally when first started.

The X server performs many difficult tasks using a wide array of hardware, requiring a very detailed configuration of certain files. If some aspect of your system changes, such as the monitor or video card, XFree86 will need to be reconfigured. In addition, if you are troubleshooting a problem with XFree86 that cannot be solved using a configuration utility, such as `Xconfigurator`, you may need to access these configuration files.

CAUTION

Xconfigurator should not be used to configure XFree86 while the X server is currently active. If your system defaults to starting up directly into X (runlevel 5), you should switch to runlevel 3 prior to running Xconfigurator. Alternatively, if your system defaults to starting up in text mode (runlevels 1-4), you should make sure that X is not running while configuring X via Xconfigurator, otherwise you may experience hardware lockups and possibly data corruption.

6.2 The XFree86 Server

Red Hat Linux uses XFree86 4 as the base X Window System, which includes the various necessary X libraries, fonts, utilities, documentation, and development tools. To provide maximum compatibility with older hardware, as well as hardware that isn't currently well supported by XFree86 4, Red Hat Linux also provides the older XFree86 3 Server packages. Both XFree86 Server versions differ in design methodologies, and as such vary in features as well as configuration details.

The primary X server (XFree86 4) includes many cutting edge XFree86 technology enhancements such as hardware 3D acceleration support, the XRender extension for anti-aliased fonts, a modular driver based design, support for modern video hardware and input devices, and many other features.

In contrast, the XFree86 3 server packages are standalone non-modular X servers, each of which act as a display driver for a select set of specific video hardware. If you're using an XFree86 3 server, you need to have installed the specific X server for your particular video card. These older servers do not support many features found only in the newer XFree86 4 servers, and are only included for compatibility. The XFree86 3 backward compatibility servers are now deprecated and will eventually be removed from some future release of Red Hat Linux.

The Red Hat Linux installation program will install the base components of XFree86, any optional XFree86 packages you may have chosen to install, the XFree86 4 X server, and any XFree86 3 X server packages that may be needed for your hardware.

The X Window system resides primarily in two locations in the filesystem.

/usr/X11R6

A directory containing X client binaries (the bin directory), assorted header files (the include directory), libraries (the lib directory), manual pages (the man directory), and various other X documentation (the doc directory).

/etc/X11

The `/etc/X11` directory heirarchy contains all of the configuration files for the various components that make up the X Window System. This includes configuration files for the X server itself, the font server (xfs), xdm, and many other base components. Display managers such as gdm and kdm, as well as various window managers, and other X tools also store their configuration in this heirarchy.

XFree86 version 4 server is a single binary executable (`/usr/X11R6/bin/XFree86`). This server dynamically loads various X server modules at runtime from `/usr/X11R6/lib/modules`) including video drivers, font engine drivers, and other special modules as needed. Some of these modules are automatically loaded by the server, whereas some are optional features that you must specify in the XFree86 4 server's configuration file `/etc/X11/XF86Config-4` before they can be used. The video drivers for XFree86 4 are located in the `/etc/X11R6/lib/modules/drivers` directory. The DRI hardware accelerated 3D drivers are located in the `/etc/X11R6/lib/modules/dri`

XFree86 version 3 servers are individual X server binaries, each of which drive a certain range of video hardware. The installed XFree86 server binaries can be found in the `/usr/X11R6/bin` directory with names in the format `XF86_<server-type>`, where `<server-type>` is the name of the server used. There are many different XFree86 3 servers including the basic `XF86_VGA16` and `XF86_SVGA` servers, and the more specialized accelerated servers such as `XF86_Mach64`, `XF86_S3`, `XF86_AGX`, and others.

6.2.1 XFree86 Server Configuration Files

The XFree86 server configuration files are stored in the `/etc/X11` directory. Since the XFree86 4 and XFree86 3 servers use a different configuration file syntax which are not compatible with each other, each has its own separate configuration file. The XFree86 4 server uses `/etc/X11/XF86Config-4`, while the XFree86 3 uses `/etc/X11/XF86Config` for X server configuration. When Red Hat Linux is installed, configuration files for both XFree86 versions are created using information gathered during the installation process. If you use `Xconfigurator` to reconfigure a new video card, both configuration files are regenerated.

Both configuration files are made up by a collection of sections, each of which defines a particular aspect of the XFree86 server's operation. Many of the configuration sections are similar or even identical in both files, however there are many differences also. One notable difference is that the `XF86Config-4` configuration file used by the XFree86 4 server contains new sections such as `ServerLayout` and `Module`, that are not found in the version 3 configuration file. The XFree86 4 server allows the usage of multiple input devices, such as mice, keyboards, and drawing tablets. Each input device is presented in its own `InputDevice` section, where it is assigned an identifying name that tells you about the device. The XFree86 3 server by contrast configures the mouse and keyboard via the directives `Keyboard` and `Pointer`.

While there is rarely a need to manually edit these files, it is useful to know about the various sections and optional parameters found in them.

Each section begins with a `Section "<section-name>"` line and ends with an `EndSection` line. Within each of the sections, you will find several lines containing an option name and at least one option value, occasionally seen in quotes. Given the similarities between the two types of configuration files, the following list explores the most useful sections of an XFree86 version 4 file and the roles of various popular settings.

Device

Specifies information about the video card used by the system. You must have at least one `Device` section in your configuration file. You may have multiple `Device` sections in the case of multiple video cards or multiple settings that can run a single card. The following options are required or widely used:

- `BusID` — Specifies the bus location of the video card. This option is only necessary for systems with multiple cards and must be set so that the `Device` section will use the proper settings for the correct card.

- `Driver` — Tells XFree86 which driver to load in order to use this video card device.

- `Identifier` — Provides a unique name for this video card. Usually, this name is set to the exact name of the video card used in this `Device` section.

- `Screen` — An optional setting used when a video card has more than one **head**, or connector to go out to a separate monitor. If you have multiple monitors connected to one video card, then separate `Device` sections must exist for each of them with a different `Screen` value for each `Device` section. The value accepted by this option is a number, starting at `0` and increasing by one for each head on the video card.

- `VideoRam` — The amount of RAM available on the video card, in kilobytes. This setting is not normally necessary, as the XFree86 server can usually probe the video card to autodetect the amount of video RAM. There is some hardware out there which XFree86 cannot correctly autodetect however, so this option lets you manually specify the exact amount of video RAM.

DRI

The **Direct Rendering Infrastructure (DRI)**, is an interface which primarily allows 3D software applications to take advantage of the 3D hardware acceleration capabilities on modern supported video hardware. In addition, the DRI can improve 2D hardware acceleration performance when using drivers that have been enhanced to use the DRI for 2D operations. When DRI is enabled in the `Module` section by loading the DRI module, this section configures options specific to DRI.

Different video cards use DRI in different ways. Before changing any DRI values, read the `/usr/X11R6/lib/X11/doc/README.DRI` file for specific information about your particular video card.

Files

> Points the XFree86 server to specific files that are used when it starts. These files contain information about particular services needed by the server. The most common options include:
>
> • FontPath — Sets the locations where the XFree86 server can find fonts. Different fixed paths to directories holding font files can be placed here, separated by commas. However, by default, Red Hat Linux uses xfs as the font server and points FontPath to unix/:7100. This tells the XFree86 server to obtain font information by using UNIX-domain sockets for inter process communication.
>
> See Section 6.5, *Fonts* for more information concerning XFree86 and fonts.
>
> • ModulePath — Allows you to set up multiple directories to use for storing modules loaded by the XFree86 server.
>
> • RgbPath — Tells the XFree86 server where the RGB color database is located on the system. This database file defines all valid color names in XFree86 and ties them to specific RGB values.

InputDevice

> ******* Configures an input device such as a mouse, keyboard used to submit information into the system using the XFree86 server. Most systems have at least two InputDevice sections, keyboard and mouse. Each section includes these two lines:
>
> • Driver — Tells XFree86 the name of the driver to load to use the device.
>
> • Identifier — Sets the name of the device, usually the name of the device followed by a number, starting with 0 for the first device. For example, the first keyboard InputDevice would have an Identifier of "Keyboard0".
>
> Most InputDevice sections contain lines that assign specific options to that particular device. Each of these lines start with Option and contain the name of the option in quotes, followed by the value to assign to that option. Mice usually receive options such as Protocol, such as PS/2, and Device, which designates the mouse to use for this section. The InputDevice section is well commented, allowing you to configure additional options for your particular devices by uncommenting certain lines.

Module

> Tells the XFree86 server which modules from the /etc/X11R6/lib/modules directory to load. Modules provide the XFree86 server with additional functionality. You should not edit these values.

Monitor

Refers to the type of monitor used by the system. There may be several `Monitor` sections, one
for each monitor in use with the machine, with one `Monitor` section as the minimum.

WARNING

Be careful when manually editing values in the options of the `Mon-`
`itor` section. Inappropriate values in this section could damage
or destroy your monitor. Consult the documentation that came
with your monitor for the safe operating parameters available.

The following options are usually configured:

- `HorizSync` — Tells XFree86 the range of horizontal sync frequencies compatible with
  the monitor, in kHz. These values are used as a guide by the XFree86 server so that it will
  know whether to use a particular `Modeline` entry's values with this monitor.

- `Identifier` — Provides a unique name for this monitor, usually numbering each monitor
  starting at `0`. The first monitor would be named `"Monitor0"`, the second `"Monitor1"`,
  and so on.

- `Modeline` — Used to specify the video modes used by the monitor at particular resolu-
  tions, with certain horizontal sync and vertical refresh resolutions. `Modeline` entries are
  usually preceded by a comment that explains what the mode line specifies.

  If your configuration file does not include comments for the various mode lines, you can
  scan over the values (or **mode descriptions**) to uncover what the mode line is attempting
  to do. See the `XF86Config` man page for detailed explanations of each mode description
  section.

- `ModelName` — An optional parameter that displays the model name of the monitor.

- `VendorName` — An optional parameter that displays the vendor that manufactured the
  monitor.

- `VertRefresh` — Lists the vertical refresh range frequencies supported by the monitor,
  in Hz. These values are used as a guide by the XFree86 server so that it will know whether
  to use a particular `Modeline` entry's values with this monitor.

`Screen`

Binds together a particular `Device` and `Monitor` that can be utilized as a pair and contain
certain settings. You must have at least one `Screen` section in your configuration file. The
following options are common:

- `DefaultDepth` — Tells the `Screen` section the default color depth to try, in bits. `8` is the default, `16` provides thousands of colors, and `32` displays millions of colors.

- `Device` — Signifies the name of the `Device` section to use with this `Screen` section.

- `Identifier` — Identifies the `Screen` section, so that it can be referred to by a `Server-Layout` section and be utilized.

- `Monitor` — Tells the name of the `Monitor` section to be used with this `Screen` section.

You may also have a `Display` subsection within the `Screen` section which tells the XFree86 server the color depth (`Depth`) and resolution (`Mode`) to try first when using this particular monitor and video card.

ServerFlags

Contains miscellaneous global XFree86 server settings. These settings may be overridden by options placed in the `ServerLayout` section. Among the most useful settings:

- `DontZap` — Prevents the use of the [Ctrl]-[Alt]-[Backspace] key combination to immediately terminate the XFree86 server.

- `DontZoom` — Prevents the use of the [Ctrl]-[Alt]-[Keypad-Plus] and [Ctrl]-[Alt]-[Keypad-Minus] to cycle through configured video resolutions.

ServerLayout

Binds together a `Screen` section with the necessary `InputDevice` sections and various options to create a unified collection of preferences used by the XFree86 server as it starts. If you have more than one `ServerLayout` section and the one to use is not specified on the command line when bringing up the XFree86 server, the first `ServerLayout` section in the configuration file is used.

The following options are used in a `ServerLayout` section:

- `Identifier` — A unique name used to describe this `ServerLayout` section.

- `InputDevice` — The names of any `InputDevice` sections to be used with the XFree86 server. Most users will only have two lines here, `Keyboard0` and `Mouse0`, the first keyboard and mouse configured for the system. The options `CoreKeyboard` and `CorePointer` refer to the fact that these are the preferred keyboard and mouse, respectively, to use with the XFree86 server.

- `Screen` — The name of the `Screen` section to use. The number to the left of the name of the `Screen` section refers to the particular screen number to use in a multi-head configuration. For standard single-head video cards, this value is `0`. The numbers to the right give the X and Y absolute coordinates for the upper-left corner of the screen, by default `0  0`.

For more information, refer to the XF86Config man page which contains a comprehensive list of detailed options.

To review the current configuration of your XFree86 server, type the xset -q command. This will provide you with information about your keyboard, pointer, screen saver, and font paths.

6.3 Desktop Environments and Window Managers

The configuration of an XFree86 server is useless until accessed by an X client that will use it to display a program using the hardware controlled by the X server. X clients are programs designed to take advantage of the X server's hardware, usually to provide interactivity with a user.

You do not have to run a complicated window manager in conjunction with a particular desktop environment to use X client applications. Assuming that you are not already in an X environment and do not have an .xinitrc file in your home directory, type the xinit command to start X with a basic terminal window (the default xterm application). You will see that this basic environment utilizes your keyboard, mouse, video card, and monitor with the XFree86 server, using the server's hardware preferences. Type **exit** at the xterm prompt to leave this basic X environment.

Of course, most computer users require more features and utility from their GUI. Developers have added layers of features to create highly developed and interactive environments that utilize the full power of the XFree86 server. These layers break into two fundamental groups based on their purpose.

6.3.1 Window Managers

Window managers are X client programs that control the way other X clients are positioned, resized, or moved. Window managers can also provide titlebars to windows, keyboard focus by keyboard or pointer (the mouse), and user-specified key and mouse button bindings. Window managers work with a collection of different X clients, wrapping around the program, making it look a certain way and appear on the screen in a particular place.

A variety of window managers are included with Red Hat Linux:

- twm — The minimalist Tab Window Manager, which provides the most basic toolset of any of the window managers.

- fvwm2 — A derivative of the twm window manager incorporating a 3D look with low memory requirements.

- enlightenment — A resource-intensive window manager with a very polished default environment.

- sawfish — The default window manager for the GNOME desktop environment, which can be used without GNOME.

- `WindowMaker` — The fully-featured GNU window manager designed to emulate the look and feel of the NEXTSTEP environment.

These window managers can be run as individual X clients to gain a better sense of their differences. Type the `xinit` `<path-to-window-manager>` command, where `<path-to-win-dow-manager>` is the location of the window manager binary file. The binary file can be found by typing `which` `<window-manager-name>` or looking for the name of the window manager in a `bin` directory.

6.3.2 Desktop Environments

A **desktop environment** brings together assorted X clients that can be run together using similar methods, utilizing a common development environment.

Desktop environments are different from window managers, which only control the appearance and placement of X client windows. Desktop environments contain advanced features that allow X clients and other running processes to communicate with one another. This allows all applications written to work in that environment to commonly integrate and be used in new ways, such as permitting drag-and-drop behavior with text.

GNOME is the default desktop environment for Red Hat Linux, using the GTK+ base widget toolkit and miscellaneous other widgets that extend the base functionality. KDE, another desktop environment, uses a different toolkit called Qt. GNOME and KDE both contain advanced productivity applications, such as word processors, spreadsheets, and control panel devices that allow you to have complete control of the look-and-feel of your user experience. Both environments can run standard X client applications, and some KDE applications can run in GNOME.

When you start X using the `startx` command, a pre-specified desktop environment is utilized. To change the default desktop environment used when X starts, open a terminal and type the `switchdesk` command. This brings up a graphical utility that allows you to select the desktop environment or window manager to use the next time X starts.

Desktop environments utilize window managers to provide the consistency in appearance between different applications. KDE contains its own window manager, called `kwm`, specifically for this functionality.

For information on the customization of the GNOME and KDE desktop environments, see the *Official Red Hat Linux Getting Started Guide*.

6.4 Runlevels

Most users run X from one of two runlevels: 3 or 5. Runlevel 3 places your system in multi-user mode with full networking capabilities. The machine will boot to a text-based login prompt with all necessary preconfigured services started. Most servers are run in runlevel 3, as X is not necessary to

provide any services utilized by most users. Runlevel 5 is similar to 3, except that it automatically starts X and provides a graphical login screen. Many workstation users prefer this method, because it never forces them to see a command prompt.

The default runlevel used when your system boots can be found in the `/etc/inittab` file. If you have a line in that file that looks like `id:3:initdefault:`, then your system will boot to runlevel 3. If you have a line that looks like `id:5:initdefault:`, your system is set to boot into runlevel 5. As root, change the runlevel number in this file to set a different default. Save the file and restart your system to verify that it boots to the correct runlevel. More information on runlevels can be found in Section 3.4, *Init Runlevels*.

Each of these runlevels start X in a slightly different way.

6.4.1 Runlevel 3: `startx`

When in runlevel 3, the preferred way to start an X session is to type the `startx` command. `startx` is a front-end to the `xinit` program, which launches the XFree86 server and connects the X clients to it. Because you must already be logged into the system at runlevel 3 to be able to type commands, `startx` is only designed to bring up certain X clients, such as a desktop environment, in a particular manner. It does not provide any user authentication.

When `startx` begins, it looks for a user-defined `.xinitrc` file in the user's home directory to define the X clients to run. If it does not find that file present, it will run the system default `/etc/X11/xinit/xinitrc` script instead. The `startx` script then does the same thing with the `.xserverrc` file, looking for it in the user's home directory and then running the default `/etc/X11/xinit/xserverrc` script if it cannot be found. Since many different X clients exist, the `xinitrc` files are very important. The `xserverrc` script is less important. It only sets the X server to connect to the X clients. Since the default X server is already configured with the `/etc/X11/X` link, Red Hat Linux does not install a default `xserverrc`.

The default `xinitrc` script then looks for user-defined files and default system files, including `.Xresources`, `.Xmodmap`, and `.Xkbmap` in the user's home directory and `Xresources`, `Xmodmap`, and `Xkbmap` in the `/etc/X11` directory. The `Xmodmap` and `Xkbmap` files, if they exist, are used by the `xmodmap` utility to configure the keyboard. The `Xresources` files are read to assign specific preference values to particular applications.

After setting these options, the `xinitrc` script executes all scripts in `/etc/X11/xinit/xini-trc.d`. One important script in this directory is `xinput`, which configures settings such as the default language to use and the desktop environment to start (from `/etc/sysconfig/desktop`).

Next, the `xinitrc` script tries to execute `.Xclients` in the user's home directory, and turns to `/etc/X11/init/Xclients` if it cannot be found. The purpose of the `Xclients` file is to start the desktop environment or, possibly, just a basic window manager. The `.Xclients` script in the user's home directory starts the user-specified desktop environment or window manager in

the `.Xclients-default` file. If `.Xclients` does not exist in the user's home directory, the standard `/etc/X11/init/Xclients` script attempts to start another desktop environment, try-ing GNOME first and then KDE. If a desktop environment cannot be found by this point, `Xclients` cycles through a list of window managers to find the one to start, after attempting the default window manager listed in the `.wm_style` file in the user's home directory.

By this point, the preferred X client applications should be started, along with the XFree86 server. If you need more details concerning starting X in runlevel 3, refer to the `startx` and `xinit` man pages and read through the scripts named above.

6.4.2 Runlevel 5: `prefdm`

Runlevel 5 uses a slightly different method to start X. When the system starts, no one is logged into the system by default. In order for a user to start a session, they must log in to the system. In runlevel 5, users authenticating themselves at the console use a **display manager**, a special X client that allows the user to submit their login name and password.

Depending on the desktop environments installed on your specific Red Hat Linux system, three dif-ferent display managers are available to handle the user authentication. The xdm display manager is the original X authentication tool. xdm only allows you to log in and start an X session, nothing more. The gdm display manager, designed to work with the GNOME desktop environment, and kdm display manager, used with the KDE desktop environment, allow you to set the desktop environment, or **session**, you would like to use after authentication. Additionally, you can restart or halt the system from the login screen. The gdm display manager also allows you to configure the language you would like to use.

When the system enters runlevel 5, a line in the `/etc/inittab` file specifies that the `prefdm` script is executed in order to determine the preferred display manager to bring up for user authentication. The `prefdm` script uses the preferences stated in the `/etc/sysconfig/desktop` file to find the proper display manager. If no desktop environment is specified, `prefdm` cycles through the gdm, kdm, and xdm display managers to find one to use. Once one is found, `prefdm` launches it to handle the user login.

Each of the display managers look to the `/etc/X11/xdm/Xsetup_0` file to set up the login screen. Once the user logs into the system, the `/etc/X11/xdm/GiveConsole` script runs to assign own-ership of the console to the user. Then, the `/etc/X11/xdm/Xsession` script runs to accomplish many of the tasks normally done by the `xinitrc` script when starting X in runlevel 3, including setting system and user resources, as well as running the scripts in the `/etc/X11/xinit/xini-trc.d` directory.

The user can specify which desktop environment they want to utilize when they authenticate using the gdm or kdm display managers by selecting it from the **Session** menu. If the desktop environ-ment is not specified in the display manager, the `/etc/X11/xdm/Xsession` script will check the

.xsession and .Xclients files in the user's home directory to decide which desktop environment to load. As a last resort, the /etc/X11/xinit/Xclients file is used to select a desktop environment or window manager to use in the same way as runlevel 3.

When the user finishes an X session on the default display (:0) and logs out, the /etc/X11/xdm/TakeConsole script runs and reassigns ownership of the console to the root user. The original display manager, which continued running after the user logged in, takes control by spawning a new display manager. This restarts the XFree86 server, displays a new login window, and starts the entire process over again.

For more information about how display managers control user authentication, read the xdm man page.

6.5 Fonts

Red Hat Linux uses xfs (X Font Server) to provide fonts to the XFree86 server and the X client applications that connect to it. While it is possible to not use xfs and place the paths to font directories in your XF86Config and XF86Config-4 configuration files, xfs has several advantages:

*   *It is easier to add and remove fonts, including editing the font path.* The **font path** is a collection of paths in the filesystem where font files are stored. The xfs service keeps the font path out of the XFree86 configuration files, making it easier to edit.

*   *Fonts may be stored on one machine acting as a networked font server, and the fonts can be shared among muliple X servers over the network.* A common set of fonts can be maintained in one place, and easily shared between all users.

*   *More types of fonts are supported.* xfs can handle TrueType, Type1, and bitmap fonts.

The XFree86 configuration files know whether to use xfs or hard coded font paths because of the FontPath setting in their Files sections. By default, the FontPath is set to unix/:7100. This tells the XFree86 server to connect to port 7100 using an inner-machine communication link. The xfs server listening on this port and will respond with font information when queried by the XFree86 server.

The xfs service must be running when X is started. If it is not, you will be returned to a command prompt with an error similar to failed to set default font path 'unix/:7100'. Check to see if xfs is running using the ps aux | grep xfs command. By default, xfs is set to start in runlevels 2, 3, 4, and 5, covering all runlevels where you would run X. If xfs is not running on your system, you can start it as root using the service xfs start command. Use the serviceconf or chkconfig utilities to force it to start at the correct runlevels.

6.5.1 xfs Configuration

The /etc/rc.d/init.d/xfs script starts the xfs server. Several options can be configured in the /etc/X11/fs/config file:

- `alternate-servers` — Sets a list of alternate font servers to be used if this font server is not available. A comma must separate every font server in the list.

- `catalogue` — An ordered list of font paths to use that contain the font files. A comma must follow every font path before a new font path can be started in the list.

  You can use the string `:unscaled` immediately after the font path to make the unscaled fonts in that path load first. Then, you can specify the entire path again, so that other scaled fonts will also be loaded.

- `client-limit` — Sets the number of clients this font server will service before refusing to handle any more. The default is `10`.

- `clone-self` — Decides if the font server will clone a new version of itself when the `client-limit` is hit. By default, this option is `on`. Set it to `off` to disable this feature.

- `default-point-size` — Sets the default point size for any font that does not specify this value. The value for this option is set in decipoints. The default of `120` corresponds to 12 point fonts.

- `default-resolutions` — Specifies a list of resolutions supported by the XFree86 server. Each resolution in the list must be separated by a comma.

- `deferglyphs` — Tells `xfs` whether to defer the loading of **glyphs**, which is an image used to visually represent a font. You can disable this feature (`none`), enable this feature for all fonts (`all`), or turn this this feature on only for 16-bit fonts (`16`), which are largely used with Asian languages.

- `error-file` — Allows you to specify the path and file name of a locations where `xfs` errors can be logged.

- `no-listen` — Tells `xfs` not to listen using a particular protocol. By default, this option is set to `tcp` to prevent `xfs` from listening on TCP ports, primarily for security reasons. If you plan on using xfs to serve fonts to networked workstations on a LAN, you need to remove the `tcp` from this line.

- `port` — Specifies the TCP port that `xfs` will listen on if `no-listen` does not exist or is commented out.

- `use-syslog` — Tells `xfs` to use the system error log if set to `on`.

6.5.2 Adding Fonts

When using `xfs`, adding fonts to your system is rather straightforward. Use the `chkfontpath --list` command to see the font paths currently configured on your system. To add new fonts in a new directory, follow these instructions as the root user:

1. Create a font directory, such as `/usr/share/fonts`, and place the fonts inside that directory. Be sure to set the permissions correctly; it is only necessary that the files can be read, no other permissions are necessary.

2. Type the `chkfontpath --add <font-directory-path>` command, where the `<font-directory-path>` is the full path to the directory holding the fonts. This will add this font path to the `xfs` configuration file.

Note

You must have a `fonts.dir` file in your new font directory for the `chkfontpath` command to work correctly. The creation of the `fonts.dir` file, as well as any other files used by `xfs` with these fonts, is beyond the scope of this document.

Many font collections available for Linux include these files for you, it may not be necessary to create them by hand.

---

3. Restart `xfs` using the `service xfs restart` command. You will also need to restart your X session.

4. Typing the `chkfontpath --list` command will show the new font path. Any fonts you added are now available for use.

The Red Hat Support website contains more information on this subject, see http://www.red-hat.com/support for additional help documents.

6.6 Additional Resources

Much more can be said about the XFree86 server, the clients that connect to it, and the assorted desktop environments and window managers. Advanced users interested in tweaking their XFree86 configuration will find these additional sources of information useful.

6.6.1 Installed Documentation

- `/usr/X11R6/lib/X11/doc` — Contains assorted documentation for XFree86, including:

  – README — Briefly describes the XFree86 architecture and how to get additional information about the XFree86 project as a new user.

  – README.Config — Explains the advanced configuration options open to XFree86 version 3 users.

– `RELNOTES` — For advanced users that want to read about the latest features available in XFree86.

- The following man pages cover particular aspects of the XFree86 server and configuring a Linux system to use an X environment:

    – `SuperProbe` — Provides an explanation of the `SuperProbe` program and useful options that can be used when it is run from the command line.

    – `Xconfigurator` — Looks at how the `Xconfigurator` program configures various aspects of the XFree86 server, examining the advanced options available that can save you time.

    – `XF86Config` — Contains information about the XFree86 configuration files, including the meaning and syntax for the different sections within the files.

    – `XFree86` — The primary man page for all XFree86 information, details the difference between local and network X server connections, explores common environmental variables, lists command line options, and provides helpful administrative key combinations.

    – `Xserver` — Focuses on the display server that X clients utilize, locally or over a network connection.

6.6.2 Useful Websites

- http://www.xfree86.org — Home page of the XFree86 project, which produces the XFree86 open source version of the X Window System. XFree86 is bundled with Red Hat Linux to control the necessary hardware and provide a GUI environment.

- http://dri.sourceforge.net — Home page of the DRI (Direct Rendering Infrastructure) project. The DRI is the core hardware 3D acceleration component of XFree86, and their website proves various resources that may prove helpful.

- http://www.redhat.com/mirrors/LDP/HOWTO/XFree86-HOWTO — A HOWTO document detailing the manual installation and custom configuration of XFree86.

- http://www.gnome.org — The home of the GNOME project, the default XFree86 desktop environment in Red Hat Linux.

- http://www.kde.org — The home of the KDE desktop environment, where you can find the latest announcements and documentation.

6.6.3 Related Books

- *The Concise Guide to XFree86 for Linux* by Aron Hsiao; Que — Provides an expert's view concerning the operation of XFree86 on Linux systems.

- *The New XFree86* by Bill Ball; Prima Publishing — Provides a good, overall look at XFree86 and its relationship with the popular desktop environments, such as GNOME and KDE.

- *Beginning GTK+ and GNOME* by Peter Wright; Wrox Press, Inc. — Introduces programmers to the GNOME architecture, showing them how to get started with GTK+.

- *GTK+/GNOME Application Development* by Havoc Pennington; New Riders Publishing — An advanced look into the heart of GTK+ programming, focusing on sample code and a thorough look at the APIs available.

- *KDE 2.0 Development* by David Sweet and Matthias Ettrich; Sams Publishing — Instructs beginning and advanced developers in how to take advantage of the many environment guidelines required to built QT applications for KDE.

Part II   Security Reference

7 Pluggable Authentication Modules (PAM)

Programs that give privileges to users must properly authenticate (verify the identity of) each user. When you log in to a system, you provide your username and password, and the login process uses the username and password to authenticate the login — to verify that you are who you say you are. Forms of authentication other than passwords are possible, and the passwords can be stored in different ways.

Pluggable Authentication Modules (PAM) is a way of allowing the system administrator to set an authentication policy without having to recompile authentication programs. With PAM, you control how particular authentication modules are plugged into a program by editing that program's PAM configuration file in /etc/pam.d.

Most Red Hat Linux users will never need to alter PAM configuration files for any of their programs. When you use RPM to install programs that require authentication, they automatically make the changes necessary to do normal password authentication using PAM. However, if you need to customize your configuration, you must understand the structure of a PAM configuration file. More information can be found in Section 7.2.2, *PAM Modules*.

7.1 Advantages of PAM

When used correctly, PAM provides many advantages for a system administrator, such as the following:

- A common authentication scheme that can be used with a wide variety of applications.
- PAM can be implemented with various applications without having to recompile the applications to specifically support PAM.
- Great flexibility and control over authentication for the administrator and application developer.
- Application developers do not need to develop their program to use a particular authentication scheme. Instead, they can focus purely on the details of their program.

7.2 PAM Configuration Files

The directory /etc/pam.d contains the PAM configuration files. In earlier versions of PAM, /etc/pam.conf was used. The pam.conf file is still read if no /etc/pam.d/ entry is found, but its use is deprecated.

Each application (or *service*, as applications designed to be used by many users are commonly known) has its own file. Each line in the file has five elements: **service name**, **module type**, **control flag**, **module path**, and **arguments**.

7.2.1 PAM Service Names

The service name of every PAM-enabled application is the name of its configuration file in /etc/pam.d. Each program which uses PAM defines its own service name.

For example, the login program defines the service name login, ftpd defines the service name ftp, and so on.

In general, the service name is the name of the program used to *access* the service, not the program used to *provide* the service.

7.2.2 PAM Modules

PAM includes four different types of modules for controlling access to a particular service:

* An auth module provides the actual authentication (perhaps asking for and checking a password) and sets credentials, such as group membership or Kerberos tickets.

* An account module checks to make sure that access is allowed for the user (the account has not expired, the user is allowed to log in at this time of day, and so on).

* A password module is used to set passwords.

* A session module is used after a user has been authenticated. A session module performs additional tasks which are needed to allow access (for example, mounting the user's home directory or making their mailbox available).

These modules may be *stacked*, or placed upon one another, so that multiple modules are used. The order of a module stack is very important in the authentication process, because it makes it very easy for an administrator to require that several conditions exist before allowing user authentication to occur.

For example, rlogin normally uses at least four stacked authentication methods, as can be seen in its PAM configuration file:

```
auth        required      /lib/security/pam_nologin.so
auth        required      /lib/security/pam_securetty.so
auth        required      /lib/security/pam_env.so
auth        sufficient    /lib/security/pam_rhosts_auth.so
auth        required      /lib/security/pam_stack.so service=system-auth
account     required      /lib/security/pam_stack.so service=system-auth
password    required      /lib/security/pam_stack.so service=system-auth
session     required      /lib/security/pam_stack.so service=system-auth
```

Before someone is allowed to `rlogin`, PAM verifies that the `/etc/nologin` file does not exist, that they are not trying to log in remotely as root over an unencrypted network connection, and that any environmental variables can be loaded. Then, a successful `rhosts` authentication is performed before the connection is allowed. If `rhosts` authentication fails, then standard password authentication is done.

New PAM modules can be added at any time, and PAM-aware applications can then be made to use them. For example, if you create a one-time-password creation method and write a PAM module to support it, PAM-aware programs can immediately use the new module and password method without being recompiled or otherwise modified in any way. As you can imagine, this is very beneficial, because it lets you mix-and-match, as well as test, authentication methods very quickly with different programs without having to recompile the programs.

Documentation on writing modules is included with the system in `/usr/share/doc/pam-<version-number>`.

7.2.3 PAM Control Flags

All PAM modules generate a success or failure result when checked. Control flags tell PAM what do with the result. Since modules can be stacked in a particular order, control flags give you the ability to set the importance of a module in respect to the modules that follow it.

Again, consider the `rlogin` PAM configuration file:

```
auth        required      /lib/security/pam_nologin.so
auth        required      /lib/security/pam_securetty.so
auth        required      /lib/security/pam_env.so
auth        sufficient    /lib/security/pam_rhosts_auth.so
auth        required      /lib/security/pam_stack.so service=system-auth
account     required      /lib/security/pam_stack.so service=system-auth
password    required      /lib/security/pam_stack.so service=system-auth
session     required      /lib/security/pam_stack.so service=system-auth
```

After the module type is specified, the control flags decide how important that particular module type should be considered in the overall goal of allowing access to the service to that user.

Four types of control flags are defined by the PAM standard:

- `required` flagged modules must be successfully checked in order for the authentication to be allowed. If a `required` module check fails, the user is not notified until all other modules of the same module type have been checked.

- `requisite` flagged modules also must be successfully checked in order for the authentication to be successful. However, if a `requisite` module check fails, the user is notified immediately with a message reflecting the first failed `required` *or* `requisite` module.

- `sufficient` flagged modules checks are ignored if they fail. But, if a `sufficient` flagged module is successfully checked and no `required` flagged modules above it have failed, then no other modules of this module type are checked and this module type is considered to have successfully been checked as a whole.

- `optional` flagged modules are not crucial for the overall success or failure of that module type's authentication. The only time they play a role is when no other modules of that module type have succeeded or failed. In this case, the success or failure of an `optional` flagged module determines the overall PAM authentication for that module type.

A newer control flag syntax that allows for even more control is now available for PAM. Please see the PAM docs located in `/usr/share/doc/pam-<version-number>` for information on this new syntax.

7.2.4 PAM Module Paths

Module paths tell PAM where to find the pluggable module to be used with the module type specified. Usually, it is provided as the full path to the module, such as `/lib/security/pam_stack.so`. However, if the full path is not given (in other words, the path does not start with a `/`), then the module indicated is assumed to be in `/lib/security`, the default location for PAM modules.

7.2.5 PAM Arguments

PAM uses arguments to pass information to a pluggable module during authentication for a particular module type. These arguments allow the PAM configuration files for particular programs to use a common PAM module but in different ways.

For example, the `pam_userdb.so` module uses secrets stored in a Berkeley DB file to authenticate the user. (Berkeley DB is an open source database system designed to be embedded in many application to track particular types of information.) The module takes a `db` argument, specifying the Berkeley DB filename to use, which can be different for different services.

So, the `pam_userdb.so` line in a PAM configuration file look like this:

```
auth        required  /lib/security/pam_userdb.so db=path/to/file
```

Invalid arguments are ignored and do not otherwise affect the success or failure of the PAM module. When an invalid argument is passed, an error is usually written to `/var/log/messages`. However, as the reporting method is controlled by the PAM module, so it is up to the module to correctly log the error.

7.2.6 PAM Configuration File Samples

A sample PAM application configuration file looks like this:

```
#%PAM-1.0
auth       required   /lib/security/pam_securetty.so
auth       required   /lib/security/pam_unix.so shadow nullok
auth       required   /lib/security/pam_nologin.so
account    required   /lib/security/pam_unix.so
password   required   /lib/security/pam_cracklib.so
password   required   /lib/security/pam_unix.so shadow nullok use_authtok
session    required   /lib/security/pam_unix.so
```

The first line is a comment (any line starting with a # character is a comment). Lines two through four stack three modules to use for login authentication.

```
auth       required   /lib/security/pam_securetty.so
```

Line two makes sure that *if* the user is trying to log in as root, the tty on which they are logging in is listed in the /etc/securetty file, *if* that file exists.

```
auth       required   /lib/security/pam_unix.so nullok
```

Line three causes the user to be asked for a password and the password to be checked, using the information store in /etc/passwd, and if it exists, /etc/shadow.

```
auth       required   /lib/security/pam_nologin.so
```

Line four checks to see if the file /etc/nologin exists. If /etc/nologin exists and the user is not root, the authentication fails.

Note that all three auth modules are checked, *even if the first auth module fails*. This strategy prevents the user from knowing why their authentication was not allowed. Knowing why their authentication failed might allow them to break the authentication more easily on their next try. You can change this behavior by changing required to requisite. If any requisite module returns failure, PAM fails immediately without calling any other modules.

```
account    required   /lib/security/pam_unix.so
```

The fifth line causes any necessary account verification to be done. For example, if shadow passwords have been enabled, the pam_unix.so module will check to see if the account has expired or if the user has not changed their password within the grace period allowed.

```
password   required   /lib/security/pam_cracklib.so
```

The sixth line tests a newly changed password by seeing whether the password can easily be determined by a dictionary-based password cracking program.

```
password   required   /lib/security/pam_unix.so shadow nullok use_authtok
```

The seventh line specifies that if the `login` program changes the user's password, it should use the `pam_unix.so` module to do so. (This will happen only if an `auth` module has determined that the password needs to be changed — for example, if a shadow password has expired.)

```
session   required   /lib/security/pam_unix.so
```

The eighth and final line specifies that the `pam_unix.so` module should be used to manage the session. Currently, that module does not do anything; it could be replaced by any necessary module or supplemented by stacking.

Note that the order of the lines within each file matters. While the order in which `required` modules are called does not matter much, there are other control flags available. While `optional` is rarely used, `sufficient` and `requisite` cause order to become important.

As the next example, we will review the `auth` configuration for `rlogin`:

```
#%PAM-1.0
auth      required     /lib/security/pam_nologin.so
auth      required     /lib/security/pam_securetty.so
auth      required     /lib/security/pam_env.so
auth      sufficient   /lib/security/pam_rhosts_auth.so
auth      required     /lib/security/pam_stack.so service=system-auth
```

First, `pam_nologin.so` checks to see if `/etc/nologin` exists. If is does, no one can log in except for root.

```
auth      required     /lib/security/pam_securetty.so
```

Second, `pam_securetty.so` keeps root logins from occurring on insecure terminals. This effectively disallows all root `rlogin` attempts. If you wish to allow them (in which case you should be behind a good firewall or not be connected to the Internet), see Section 7.4, *Using* `rlogin, rsh, and rexec with PAM`.

```
auth      required     /lib/security/pam_env.so
```

Third, the `pam_env.so` module loads the environmental variables specified in `/etc/security/pam_env.conf`.

```
auth      sufficient   /lib/security/pam_rhosts_auth.so
```

Fourth, if `pam_rhosts_auth.so` authenticates the user using `.rhosts` in the user's home directory, PAM immediately authenticates the `rlogin` without moving on to do a normal password authentication with `pam_stack.so`. If `pam_rhosts_auth.so` fails to authenticate the user, that failed authentication is ignored.

```
auth      required     /lib/security/pam_stack.so service=system-auth
```

Fifth, if `pam_rhosts_auth.so` has failed to authenticate the user, the `pam_stack.so` module performs normal password authentication, and is passed the `service=system-auth` argument.

Note

If you do not want to prompt for a password when the `securetty` check fails and determines that the user is trying to login as root remotely, you can change the `pam_securetty.so` module from `required` to `requisite`. Alternatively, if you want to allow root logins remotely (which is not a good idea), you can comment out this line.

7.3 **Shadow Passwords**

If you are using shadow passwords, `pam_unix.so` will automatically detect that they are in use and will use them to authenticate users.

Please refer to *Official Red Hat Linux Customization Guide* for more information on shadow passwords.

7.4 **Using `rlogin`, `rsh`, and `rexec` with PAM**

For security reasons, `rexec`, `rsh`, and `rlogin` are not enabled by default in Red Hat Linux 7.2. You should use the OpenSSH suite of tools instead. Information concerning the OpenSSH tools can be found in Chapter 10, *SSH Protocol* and the *Official Red Hat Linux Customization Guide*.

If you must use `rexec`, `rsh`, and `rlogin`, and if you need to use them as root, you will need to make a few modifications to the `/etc/securetty` file. All three of these tools have PAM configuration files that require the `pam_securetty.so` PAM module, so you must edit `/etc/securetty` to allow root access.

Before you can log in as root using these tools, you first have to have them properly set up. First, install the `rsh-server` RPM, which is included with Red Hat Linux 7.2. See the *Official Red Hat Linux Customization Guide* if you need assistance using RPM.

Next, run `serviceconf` and enable `rexec`, `rsh`, and `rlogin`. See the `serviceconf` man page if you need help using this tool.

Finally, restart `xinetd` with `/sbin/service xinetd restart` to activate the `serviceconf` changes. At this point, all users except root will be able to use `rexec`, `rsh`, and `rlogin`.

To allow root to use these tools, add the names of the tools you wish to allow to the `/etc/securetty`. If you wanted to enable root login using `rexec`, `rsh`, and `rlogin`, add the following lines to `/etc/securetty`:

```
rexec
rsh
rlogin
```

To allow root to log in using these tools via `telnet` (an even worse idea but necessary in some environments), add a few more lines:

```
pts/0
pts/1
```

7.5 Additional Resources

Much more information about PAM is available than what is covered in this chapter. Various additional sources of information exist and will prove invaluable in helping to configure and use PAM on your system.

7.5.1 Installed Documentation

- `pam` man page — Good introductory information on PAM, including the structure and purpose of the PAM configuration files.

- `/usr/share/doc/pam-<version-number>` — Contains excellent HTML documentation on PAM, including a *System Administrators' Guide*, a *Module Writers' Manual*, and an *Application Developers' Manual*. Also contains a copy of the PAM standard, DCE-RFC 86.0.

7.5.2 Useful Websites

- http://www.kernel.org/pub/linux/libs/pam — The primary distribution website for the Linux-PAM project, containing information on various PAM modules and applications in use or in development, a FAQ, and additional PAM documentation.

In addition to these sources, we suggest that you read as many configuration file examples as possible when beginning to work with PAM. Many websites offer code examples, both for administrators who want to change default configuration files and for application developers who want to use PAM with their programs.

8   Using Kerberos 5 on Red Hat Linux

Kerberos is a secure system for providing network authentication services. Authentication means:

• The identities of entities on the network are verified.

• Traffic on the network is from the source who claims to have sent it.

Kerberos uses passwords to verify the identity of users, and these passwords are never sent over the network in an unencrypted form.

For information on configuring a Kerberos 5 server or client, refer to the *Official Red Hat Linux Customization Guide*.

8.1 Why Use Kerberos?

Most conventional network systems use password-based authentication schemes. When a user needs to authenticate to a service running on a network server, they type in their password for each service that requires authentication. Their password is sent over the network, and the server verifies their identity using the password.

Transmission of passwords in plaintext using this method, while commonly done, is a tremendous security risk. Any system cracker with access to the network and a packet analyzer (also known as a packet sniffer) can intercept any passwords sent this way.

The primary design goal of Kerberos is to ensure that passwords are *never* sent across a network unencrypted and are preferably never sent over the network at all. The proper use of Kerberos will eradicate the threat of packet sniffers intercepting passwords on your network.

8.2 Why Not Use Kerberos?

Kerberos removes a common and severe security threat, so why is it not in use on every network? For several reasons, Kerberos may be difficult to implement:

• No quick solution exists for migrating user passwords from a standard UNIX password database (such as /etc/passwd or /etc/shadow) to a Kerberos password database. Migration is technically feasible, but this issue is beyond the scope of this chapter. For help deciding whether a password migration makes sense for your Kerberos installation, see the Kerberos FAQ Question 2.23 or the information referenced in Section 8.6, *Additional Resources* for more detailed information concerning this issue.

• Kerberos is only partially-compatible with the Pluggable Authentication Modules (PAM) system used by most servers running Red Hat Linux. For more information on this issue, see Section 8.5, *Kerberos and PAM*.

- For an application to use Kerberos, its source must be modified to make the appropriate calls into the Kerberos libraries. For some applications, this may require too much programming effort. For other applications, changes must be made to the protocol used between network servers and their clients. Again, this may require extensive programming. Furthermore, it may be impossible to make certain closed-source applications work with Kerberos.

- Kerberos assumes that you are using trusted hosts on an untrusted network. Its primary goal is to prevent plaintext passwords from being sent across that network. However, if anyone other than the proper user has physical access to any of the hosts, especially the one that issues tickets used for authentication, the entire Kerberos authentication system is at risk of being compromised.

- Finally, if you decide to use Kerberos on your network, you must realize that it is an all-or-nothing proposition. If *any* services that transmit plaintext passwords remain in use, passwords can still be compromised, and your network gains no net benefit from the use of Kerberos. To secure your network with Kerberos, you must either **kerberize** (make it work with Kerberos) *all* applications that send plaintext passwords or stop using those insecure applications on your network.

8.3 Kerberos Terminology

Like any other system, Kerberos has its own terminology. Before we talk about how it works, here is a list of terms that you will need to know:

ciphertext

Encrypted data.

client

An entity on the network (a user, a host, or an application) that can get a ticket from Kerberos.

credential cache or ticket file

A file which contains the keys for encrypting communications between a user and various network services. Kerberos 5 provides a framework for using other cache types (such as shared memory), but files are better supported.

key

Data used when encrypting or decrypting other data. Encrypted data cannot be decrypted without the proper key or extremely good guessing.

Key Distribution Center (KDC)

A service that issues Kerberos tickets, usually run on the same host as the Ticket Granting Server.

key table or keytab

A file that includes an unencrypted list of principals and their keys. Servers retrieve the keys they need from keytab files instead of using `kinit`. The default keytab file is `/etc/krb5.keytab`. The `kadmind` command is the only service that uses any other file (it uses `/var/kerberos/krb5kdc/kadm5.keytab`).

plaintext

Unencrypted data.

principal

A user or service that can authenticate using Kerberos. A principal's name is in the form "*root*[/*instance*]@*REALM*". For a typical user, the *root* is the same as their login ID. The *instance* is optional. If the principal has an instance, it is separated from the root with a forward slash ("/"). An empty string ("") is actually a valid instance (which differs from the default, *NULL* instance), but using it can be confusing. All principals in a realm have their own key, which is derived from their password (for users) or randomly set (for services).

realm

A network that uses Kerberos, composed of one or a few servers (also known as KDCs) and a potentially very large number of clients.

service

A program or computer accessed over the network.

ticket

A temporary set of electronic credentials that verify the identity of a client for a particular service.

Ticket Granting Service (TGS)

Issues tickets for a desired service that are used by the user to actually gain access to the service. The TGS usually runs on the same host as the KDC.

Ticket Granting Ticket (TGT)

A special ticket which allows the client to obtain additional tickets without applying for them from the KDC.

8.4 How Kerberos Works

Now that you have heard a few of the terms that Kerberos uses, here is a simplified explanation of how a Kerberos authentication system works:

On a "normal" network which uses passwords to authenticate users, when a user requests a network service that requires authentication, the user is prompted to type in their password. The password is transmitted in plaintext over the network, and access to the network service is granted.

As mentioned previously, the central problem solved by Kerberos is how to use passwords for authentication without sending them over the network. On a kerberized network, the Kerberos database contains principals and their keys (for users, their keys are derived from their passwords). The Kerberos database also contains keys for all of the network services.

When a user on a kerberized network logs in to their workstation, their principal is sent to the Key Distribution Center (KDC) as a request for a Ticket Granting Ticket (TGT). This request can be sent by the login program (so that it is transparent to the user) or can be sent by the kinit program after the user logs in.

The KDC checks for the principal in its database. If the principal is found, the KDC creates a TGT, encrypts it using the user's key, and sends it back to the user.

The login program or kinit decrypts the TGT using the user's key (which it computes from the user's password). The TGT, which is set to expire after a certain period of time, is stored in your credentials cache. An expiration time is set so that a compromised TGT can only be used for a certain period of time, usually eight hours (unlike a compromised password, which could be used until changed). The user will not have to re-enter their password until the TGT expires or they logout and login again.

When the user needs access to a network service, the client uses the TGT to request a ticket for the service from the Ticket Granting Service (TGS), which runs on the KDC. The TGS issues a ticket for the desired service, which is used to authenticate the user.

As you might have guessed, this explanation is a simplified description of Kerberos authentication events. If you need a more in-depth explanation of how Kerberos works, see Section 8.6, *Additional Resources*.

Note

Kerberos depends on certain network services to work correctly. First, Kerberos requires approximate clock synchronization between the machines on your network. If you have not set up a clock syncing program for your network, you will need to do so. Also, since certain aspects of Kerberos rely on the Domain Name System (DNS), be sure that the DNS entries and hosts on your network are all correctly configured. See the *Kerberos V5 System Administrator's Guide*, provided in PostScript and HTML formats, in /usr/share/doc/krb5-server-*<version-number>*, for more information on these issues.

8.5 Kerberos and PAM

Currently, kerberized services do not make use of Pluggable Authentication Modules (PAM) at all — a kerberized server bypasses PAM completely. Applications that use PAM can make use of Kerberos for password checking if the `pam_krb5` module (provided in the `pam_krb5` package) is installed. The `pam_krb5` package contains sample configuration files that will allow services like `login` and `gdm` to authenticate users and obtain initial credentials using their passwords. If access to network servers is always done using kerberized services (or services that use GSS-API, like IMAP), the network can be considered reasonably safe.

Careful system administrators will not add Kerberos password checking to all network services, because most of the protocols used by these services do not encrypt the password before sending it over the network — obviously something to avoid.

8.6 Additional Resources

Kerberos can be a challenge for new users to understand, implement and configure. For more examples and instructions on using Kerberos, refer to the following sources of information:

8.6.1 Installed Documentation

- `/usr/share/doc/krb5-server-<version-number>` — The *Kerberos V5 Installation Guide* and the *Kerberos V5 System Administrator's Guide*, in PostScript and HTML formats, are installed by the `krb5-server` RPM.

- `/usr/share/doc/krb5-workstation-<version-number>` — The *Kerberos V5 UNIX User's Guide*, in PostScript and HTML formats, is installed by the `krb5-workstation` RPM.

8.6.2 Useful Websites

- http://web.mit.edu/kerberos/www — The Kerberos home page on MIT's website.

- http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html — The Kerberos Frequently Asked Questions (FAQ).

- ftp://athena-dist.mit.edu/pub/kerberos/doc/usenix.PS — Link to a PostScript version of *Kerberos: An Authentication Service for Open Network Systems* by Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. This document is the original paper describing Kerberos.

- http://web.mit.edu/kerberos/www/dialogue.html — *Designing an Authentication System: a Dialogue in Four Scenes* originally by Bill Bryant in 1988, modified by Theodore Ts'o in 1997. This document is a conversation between two developers who are thinking through the creation of a

Kerberos-style authentication system. The conversational style of the discussion make this a good starting place for people who are completely unfamiliar with Kerberos.

• http://www.ornl.gov/~jar/HowToKerb.html — Practical advice on kerberizing your network.

9   TCP Wrappers and `xinetd`

Controlling access to network services can be a challenge. Firewalls are useful for controlling access in and out of a particular network, but they can be difficult to configure. TCP wrappers and `xinetd` control access to services by hostname and IP addresses. In addition, these tools also include logging and utilization management capabilities that are easy to configure.

9.1 Purpose of TCP Wrappers

Many modern network services, such as SSH, Telnet, and FTP, make use of **TCP wrappers**, a program that is designed to stand between an incoming request and the requested service. TCP wrappers is installed by default with a server-class installation of Red Hat Linux, providing many advantages over running a variety of different services, each with their own access control methods.

The idea behind TCP wrappers is that, rather than allowing an incoming client connection to communicate directly with a network service daemon running as a separate process on a server system, the target of the request is "wrapped" by another program, allowing a greater degree of access control and logging of who is attempting to use the service.

The functionality behind TCP wrappers is provided by `libwrap.a`, a library that network services, such as `xinetd`, `sshd`, and `portmap`, are compiled against. Additional network services, even networking programs you may write, can be compiled again `libwrap.a` to provide this functionality. Red Hat Linux bundles the necessary TCP wrapper programs and library in the `tcp_wrappers-<version>` RPM file.

9.1.1 TCP Wrapper Advantages

When someone attempts to access a network service using TCP wrappers, a small wrapper program reports the name of the service requested and the client's host information. The wrapper program does not directly send any information back to the client, and after the access control directives are satisfied, the wrapper gets out of the way, not placing any additional overhead on the communication between the client and server.

TCP wrappers provide two basic advantages over other network service control techniques:

- *The connecting client is unaware that TCP wrappers are in use.* Legitimate users will not notice anything different, and attackers never receive any additional information about why their attempted connections failed.

- *TCP wrappers operate in a manner that is separate from the applications the wrapper program protects.* This allows many applications to share a common set of configuration files for simpler management.

9.2 Host-Based Access Control Lists

Host-based access for services that use TCP wrappers is controlled by two files: `hosts.allow` and `hosts.deny`. These files, located in the `/etc` directory, use a simple format to control access by particular systems or users to certain services on a server.

The default rule is to allow anyone to access to the services if no rules are specified in either `hosts.allow` and `hosts.deny`. Rules in `hosts.allow` take precedence over rules specified in `hosts.deny`. Even if a rule specifically denying all access to a particular service is defined in `hosts.deny`, hosts specifically given access to the service in `hosts.allow` are allowed to access it. All rules in each file take effect from the top down, so the order in which the rules are placed can be important.

Any changes to these files take effect immediately on the affected services; restarting of the services is not required.

9.2.1 Formatting Rules

All access control rules are placed on lines within `hosts.allow` and `hosts.deny`, and any blank lines or lines that start with the comment character (#) are ignored. Each rule needs to be on a separate line.

The rules must be formatted in the following manner:

    <daemon_list>: *<client_list>*[: *<shell_command>*]

Each of these options refer to a different part of the rule:

•   `daemon_list` — A collection of one or more process names or special wildcards, separated by whitespace.

•   `client_list` — One or more hostnames, host addresses, patterns, or wildcards, separated by whitespace, to use when a particular process name matches a requested service.

•   `shell_command` — An optional component that specifies something to be done in the event a rule is utilized.

Patterns are particularly helpful when specifying groups of clients that may or may not access a certain service. By placing a "`.`" character at the beginning of a string, all hosts that share the end of that string are applied to that rule. So, `.domain.com` would catch both `system1.domain.com` and `system2.domain.com`. The "`.`" character at the end of a string has the same effect, except going the other direction. This is primarily used for IP addresses, as a rule pertaining to `192.168.0.` would apply to the entire class C block of IP addresses. Netmask expressions can also be used as a pattern to control access to a particular group of IP addresses. You can even use asterisks (`*`) or question marks (`?`) to select entire groups of hostnames or IP addresses, so long as you do not use them in the same string as the other types of patterns.

If your list of hostnames that may access a particular service is too long or is difficult to control within `host.allow` or `hosts.deny`, you can also specify the full path to a file (such as `/etc/telnet.hosts.deny`). This file should contain different hostnames, host addresses, or patterns, separated by whitespace, that you want to allow or deny access to that service. This method also works well to share access control lists between various services, as changes would only need to be made in one file to affect each service.

The following wildcards may be used in the access control rules instead of using specific hosts or groups of hosts:

- `ALL` — Matches every client with that particular service or even every service using access control. The `ALL` wildcard can also apply to daemons.

- `LOCAL` — Matches any host that does not contain a "`.`" character.

- `KNOWN` — Matches any host where the hostname and host address are known or where the user is known.

- `UNKNOWN` — Matches any host where the hostname or host address are unknown or where the user is unknown.

- `PARANOID` — Matches any host where the hostname does not match the host address.

> **⟨CAUTION⟩**
>
> The `KNOWN`, `UNKNOWN`, and `PARANOID` wildcards should be used very carefully, as a disruption in name resolution may make prevent legitimate users from gaining access to a network service.

The access control language also contains a powerful operator, `EXCEPT`, which allows separate lists to be combined within the same rule line. When `EXCEPT` is used between two lists, the first list applies unless an entry from the second list matches an entity covered by the first list. `EXCEPT` can be used with daemon or client lists. Consider the following `hosts.allow` example:

```
# all domain.com hosts are allowed to connect
# to all services except cracker.domain.com
ALL: .domain.com EXCEPT cracker.domain.com

# 123.123.123.* addresses can use all services except FTP
ALL EXCEPT in.ftpd: 123.123.123.
```

Note

Organizationally, it usually makes more sense to use `EXCEPT` operators spar-
ingly, choosing instead to place the exceptions to the rule in the other access
control file. This allows all administrators to quickly scan the appropriate
files to see what hosts should be allowed or denied access to which services,
without having to work through various `EXCEPT` operators and work out the
appropriate logic.

The best way to manage access control with `hosts.allow` and `hosts.deny` is to use the two
files together to achieve the desired results. Users that wish to prevent any hosts other than specific
ones from accessing services usually place `ALL: ALL` in `hosts.deny`. Then, they place lines in
`hosts.allow`, such as `portmap, in.telnetd: 10.0.1.24` or `in.ftpd: 10.0.1.`
`EXCEPT 10.0.1.1`, to selectively let certain hosts in.

Alternatively, some administrators allow anyone to use network services except for specific hosts.
In this case, nothing is placed in `hosts.allow` and any necessary restrictions are placed in
`hosts.deny`, such as `in.fingerd: 192.168.0.2`.

> **WARNING**
>
> **Be very careful about using hostnames and domain names in both access
> control files, especially `hosts.deny`. Various tricks could be used by
> an attacker to circumvent rules specifying them by name. In addition,
> if your system selectively allows access based on hostname and domain
> name information, any disruption in DNS service would prevent even
> authorized users from using network services.**
>
> **Using IP addresses whenever possible can prevent many problems when
> constructing access control rules, especially those that deny access.**

Beyond simply allowing or denying access to services for certain hosts, the access control language
also supports the use of shell commands when that rule is utilized. These shell commands are most
commonly used with deny rules to set up **booby traps**, which usually trigger actions that log informa-
tion about failed attempts to a special file or email an administrator. This is an example of a booby trap
located in the `hosts.deny` file which will write a log line containing the date and client information
every time a host from the 10.0.1.0 to 10.0.1.255 range attempts to connect via Telnet:

```
in.telnetd: 10.0.1.: (/bin/echo `date` %c >> /var/log/telnet.log) &
```

Various **expansions** containing specific information about the client, server, and process involved are available to the shell commands:

- %a — The client's IP address.

- %A — The server's IP address.

- %c — Various types of client information, such as the username and hostname, or the username and IP address.

- %d — The daemon process name.

- %h — The client's hostname (or IP address, if the hostname is unavailable).

- %H — The server's hostname (or IP address, if the hostname is unavailable).

- %n — The client's hostname. If unavailable, unknown is printed. If the client's hostname and host address do not match, paranoid is printed.

- %N — The server's hostname. If unavailable, unknown is printed. If the server's hostname and host address do not match, paranoid is printed.

- %p — The daemon process ID.

- %s — Various types of server information, such as the daemon process and the host or IP address of the server.

- %u — The client's username. If unavailable, unknown is printed.

For a full examination of shell commands, as well as some additional access control examples, review the hosts_access(5) man page.

Note

Special attention must be given to portmap when using it with host access control lists. Only IP addresses or the ALL option should be used when specifying hosts to allow or deny, as host names are not supported. In addition, changes to the host access control lists that concern portmap may not take affect immediately.

As widely used services, such as NIS and NFS, depend on portmap to operate, be aware of these limitations before depending on hosts.allow and hosts.deny to control access by certain hosts.

9.3 Access Control Using `xinetd`

The benefits offered by TCP wrappers are only multiplied when the `libwrap.a` library is used in conjunction with `xinetd`, a super-daemon that provides additional access, logging, binding, redirection and resource utilization control.

Red Hat Linux configures a variety of popular network services to be used with `xinetd`, including FTP, IMAP, POP, and telnet. When any of these services are accessed via their port numbers in `/etc/services`, the `xinetd` daemon handles the request. Before bringing up the requested network service by the correct user, `xinetd` ensures that the client host information meets the access control rules, the number of instances of this service is under a particular threshold, and any other rules specified for that service or all `xinetd` services are followed. Once the target service is brought up for the connecting client, `xinetd` goes back to sleep, waiting for additional requests for the services it manages.

9.3.1 `xinetd` Configuration Files

The `xinet` service is controlled by the `/etc/xinetd.conf` file, as well as the various service-specific files in the `/etc/xinetd.d` directory.

/etc/xinetd.conf

The `xinetd.conf` file is the parent of all `xinetd`-controlled service configuration files, as the service-specific files are also parsed every time `xinetd` starts. By default, `xinetd.conf` contains some basic configuration settings that apply to every service:

```
defaults
{
 instances              = 60
        log_type                = SYSLOG authpriv
        log_on_success  = HOST PID
        log_on_failure  = HOST
}

includedir /etc/xinetd.d
```

These lines control various aspects of how `xinetd` does its job:

*   `instances` — Sets the maximum number of requests a particular service can handle at once.

*   `log_type` — Tells `xinetd` to use the `authpriv` log, specified in `/etc/syslog.conf` and set to `/var/log/secure` by default, rather than using another specific file. Using `FILE /var/log/xinetdlog` here instead would move `xinetd` logging to a separate `/var/log/xinetdlog` file.

- `log_on_success` — Lets `xinetd` know what to log if the connection is successful. By default, the remote host's IP address and the process ID of server processing the request are recorded.

- `log_on_failure` — Tells `xinetd` what to log if the connection fails or is not allowed. The `log_on_success` and `log_on_failure` settings in `/etc/xinetd.conf` are often added to by each of the different services, meaning that successful and failed connections by each service will usually log more than what is indicated here.

Various logging options are available for use in `/etc/xinetd.conf` and the service-specific `xinetd` configuration files:

- `ATTEMPT` — Logs the fact that a failed attempt was made. (`log_on_failure`)

- `DURATION` — Logs the length of time the service is used by a remote system. (`log_on_success`)

- `EXIT` — Logs the exit status or termination signal of the service. (`log_on_success`)

- `HOST` — Logs the remote host's IP address. (`log_on_failure` and `log_on_success`)

- `PID` — Logs the process ID of the server receiving the request. (`log_on_success`)

- `RECORD` — Records information about the remote system in the case the service cannot be started. Only particular services, such as `login` and `finger`, may use this option. (`log_on_failure`)

- `USERID` — Logs the remote user using the method defined in RFC 1413 for all multi-threaded stream services. (`log_on_failure` and `log_on_success`)

Other options for `/etc/xinetd.conf` are available, such as `per_source`, which limits the maximum number of connections from a particular IP address to a specific service.

Files in the `/etc/xinetd.d` Directory

The various files in the `/etc/xinetd.d` directory are read every time `xinetd` starts, due to the `includedir /etc/xinetd.d` statement at the bottom of `/etc/xinetd.conf`. These files, with names such as `finger`, `ipop3`, and `rlogin`, relate to the various services controlled by `xinetd`.

The files in `/etc/xinetd.d` use the same conventions and options as what is seen in `/etc/xinetd.conf`. The primary reason they are in separate configuration files, one for each service, is to make it easier to add and remove services from the domain of `xinetd` without affecting its other services.

To get an idea of how these files are structured, consider the `wu-ftp` file:

```
service ftp
{
        socket_type             = stream
```

```
            wait                    = no
            user                    = root
            server                  = /usr/sbin/in.ftpd
            server_args             = -l -a
            log_on_success          += DURATION USERID
            log_on_failure          += USERID
            nice                    = 10
            disable                 = yes
    }
```

The first line defines the service's name that is being configured. Then, the lines within the brackets contain a variety of different settings that define how this service is supposed to be started and used. The `wu-ftp` file states that the FTP service uses a `streaming` socket type (rather than `dgram`), the binary executable file to use, the arguments to pass to the binary, the information to log in addition to the `/etc/xinetd.conf` settings, the priority with which to run the service, and more.

The use of `xinetd` with a particular service also can serve as a basic level of protection from a Denial of Service (DoS) attack. The `max_load` option takes a floating point value to set a CPU usage threshold when no more connections for a particular service will be accepted, preventing certain services from overwhelming the system. The `cps` option accepts an integer value to set a rate limit on the number of connections available per second. Configuring this value to something low, such as 3, will help prevent attackers from being able to flood your system with too many simultaneous requests for a particular service.

Access Control within `xinetd`

Users of `xinetd` services can choose to use the TCP wrapper host access control files (`hosts.al-low` and `hosts.deny`), provide access control via the `xinetd` configuration files, or a mixture of both. Information concerning the use of TCP wrapper host access control files can be found in Section 9.2, *Host-Based Access Control Lists*. This section will discuss using `xinetd` to control access to the services it controls.

Note

Unlike TCP wrapper host access control files, any changes to `xinetd` configuration files require a restart of the `xinetd` service, as well as a restart of any service affected by the change, to go into affect.

The `xinetd` host access control available through its various configuration files is different from the method used by TCP wrappers. While TCP wrappers places all of the access configuration within two files, `/etc/hosts.allow` and `/etc/hosts.deny`, each service's file in `/etc/xinetd.d` can contain access control rules based on the hosts that will be allowed to use that service.

The following options are supported in the `xinetd` files to control host access:

- `only_from` — Allows the hosts specified to use the service.

- `no_access` — Blocks these hosts from using this service.

- `access_times` — Specifies the time range when a particular service may be used. The time range must be stated in a `HH:MM-HH:MM` format using 24-hour notation.

The `only_from` and `no_access` options can use a list of IP addresses or host names, or you can specify an entire network. Like TCP wrappers, combining `xinetd` access control with the proper logging configuration for that service, you can not only block the request but also record every attempt to access it.

For example, the following `/etc/xinetd.d/telnet` file can be used to block `telnet` access to a system by a particular network group and restrict the overall time range that even legitimate users can log in:

```
service telnet
{
        disable         = no
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        no_access       = 10.0.1.0/24
        log_on_success  += PID HOST EXIT
        access_times    = 09:45-16:15
}
```

In this example, when any system from the 10.0.1.0/24 subnet, such as 10.0.1.2, tries to `telnet` into the `boo` host, they will receive a message stating `Connection closed by foreign host`. In addition, their login attempt is logged in `/var/log/secure`:

```
May 15 17:35:47 boo xinetd[16188]: START: telnet pid=16191 from=10.0.1.2
May 15 17:38:49 boo xinetd[16252]: START: telnet pid=16256 from=10.0.1.2
May 15 17:38:49 boo xinetd[16256]: FAIL: telnet address from=10.0.1.2
May 15 17:38:49 booxinetd[16252]: EXIT: telnet status=0 pid=16256
```

Binding and Port Redirection

The service configuration files for `xinetd` also support binding the service to a particular IP address and redirecting incoming requests for that service to another IP address, hostname, or port.

Binding, controlled with the `bind` option in the service configuration files, specifically links the service to a particular IP address in use with the system, only allowing requests that use that IP address

to access the service. This is particularly useful for systems with multiple network adapters and using multiple IP addresses, such as machines being used as firewalls, with one network adapter facing the Internet and the other connected to an internal network. Attackers attempting to connect for a specific service, such as Telnet or FTP, via the Internet connection may be blocked from connecting to the service while internal users may connect to the service via the NIC connected to the internal network.

The `redirect` option, which accepts an IP address or hostname followed by a port number, tells the service to redirect any requests for this service to the specified location. This feature can be used to point to another port number on the same system, redirect the request to different IP address on the same machine, shift the request to a totally different system and port number, or any combination of these options. In this way, a user connecting to certain service on a system may be rerouted to another system with no disruption.

The `xinetd` daemon is able to accomplish this redirection by spawning a process that stays alive for the duration of the connection between the requesting client machine and the host actually providing the service, transferring data between the two systems.

The real strength of the `bind` and `redirect` options can be seen when they are used together. By binding a service to a particular IP address on a system and then redirecting requests for this service to a second machine that only the first machine can see, you can use an internal system to provide services for a totally different network. Alternatively, these options can be used to limit the exposure of a particular service on a multihomed machine to a known IP address, as well as redirect any requests for that service to another machine specially configured for that purpose.

For example, consider a system that is used as a firewall with this setting for its FTP service:

```
service ftp
{
        socket_type  = stream
        wait    = no
        user    = root
        server    = /usr/sbin/in.ftpd
        server_args  = -l -a
        log_on_success  += DURATION USERID
        log_on_failure  += USERID
        bind                    = 123.123.123.123
        redirect                = 10.0.1.13 21
}
```

The `bind` and `redirect` options in this file will ensure that the FTP service on the machine is bound to the external IP address (123.123.123.123), the one facing the Internet. In addition, any requests for FTP service sent to 123.123.123.123 will be redirected via a second network adapter to an internal IP address (10.0.1.13) that only the firewall and internal systems can access. The firewall will then send the communication between the two systems, and the connecting system will think it is connected to 123.123.123.123 while it is actually connected to a different machine.

This feature is particularly useful for users with broadband connections and only one fixed IP address. When using Network Address Translation (NAT), the systems behind the gateway machine, which are using internal-only IP addresses, are not available from outside the gateway system. However, when certain services controlled by xinetd are configured with the bind and redirect options, the gateway machine can act as a type of proxy between outside systems and a particular internal machine configured to provide the service. In addition, the various xinetd access control and logging options are also available for additional protection, such as limiting the number of simultaneous connections for the redirected service.

9.4 Additional Resources

Additional information concerning TCP wrappers and xinetd is available on your system and on the Web.

9.4.1 Installed Documentation

The bundled documentation on your system is a good place to start looking for additional TCP Wrappers, xinetd, and access control configuration options.

- /usr/share/doc/tcp_wrappers-<*version*> — Contains a README file that discusses how TCP wrappers work and the various hostname and host address spoofing risks that exist.

- /usr/share/doc/xinetd-<*version*> — Includes a README file that discusses aspects of access control and a sample.conf file with various ideas for modifying /etc/xinetd.d service configurations.

- For detailed information concerning the creation of TCP wrapper access control rules, read the hosts_access (section 5) and hosts_options man pages.

- The xinetd and xinetd.conf man pages contain additional information for creating xinetd configuration files and a description of how xinetd works.

9.4.2 Useful Websites

- http://www.xinetd.org — The home of xinetd, containing sample configuration files, a full listing of features, and an informative FAQ.

- http://www.macsecurity.org/resources/xinetd/tutorial.shtml — A thorough tutorial that discusses many different ways to tweak default xinetd configuration files to meet specific security goals.

10   SSH Protocol

This chapter covers the benefits of the SSH™ protocol, the sequence of events that occur when a secure connection is made to a remote system, the different layers of SSH, and methods to ensure SSH is used by users connecting to your system.

Common methods for remotely logging into another system through a shell (`telnet`, `rlogin`, or `rsh`) or copying files between hosts (`ftp` or `rcp`) do not encrypt data that is sent over the connection between the client and the server, and should be avoided. Instead, you should only connect to a remote host using a secure shell or an encrypted virtual private network. Using secure methods to remotely log in to other systems will decrease the security risks for both your system and the remote system.

10.1 Introduction

SSH (or *S*ecure *SH*ell) is a protocol for creating a secure connection between two systems. In the SSH protocol, the client machine initiates a connection with a server machine. The following safeguards are provided by SSH:

- After an initial connection, the client can verify that it is connecting to the same server during subsequent sessions.

- The client can transmit its authentication information to the server, such as a username and password, in an encrypted format.

- All data sent and received during the connection is transferred using strong encryption, making it extremely difficult to decrypt and read.

- The client has the ability to use X11[1]applications launched from the shell prompt. This technique provides a secure, graphical interface (called **X11 forwarding**).

The server benefits from SSH, as well, especially if it is running a number of services. If you use **port forwarding**, otherwise insecure protocols (for example, POP) can be encrypted for secure communication with remote machines. SSH makes it relatively simple to encrypt different types of communication normally sent insecurely over public networks.

Red Hat Linux 7.2 includes the OpenSSH server (`openssh-server`) and client (`openssh-clients`) packages, as well as the general OpenSSH package (`openssh`) which must be installed for either of them to work. Please see the *Official Red Hat Linux Customization Guide* for instructions on installing and deploying OpenSSH on your Red Hat Linux system.

---

[1]   X11 refers to the X11R6 windowing display system, traditionally referred to as X. Red Hat Linux includes XFree86, a widely used, open source X Window System, which is based on X11R6.

The OpenSSH packages require the OpenSSL package (`openssl`). OpenSSL installs several important cryptographic libraries that help OpenSSH provide encrypted communications. You must install the `openssl` package before installing any OpenSSH packages.

A large number of client and server programs can use the SSH protocol, including many open source and freely available applications. Several different SSH client versions are available for almost every major operating system in use today. Even if the users connecting to your system are not running Red Hat Linux, they can still find and use an SSH client native for their operating system.

10.1.1 Why Use SSH?

Threats to network traffic include packet sniffing, DNS and IP spoofing[2]and the promulgation of fake routing information. In general terms, these threats can be categorized as follows:

- *Interception of communication between two systems* — In this scenario, a third party exists somewhere on the network between communicating entities and makes a copy of the information being passed between them. The intercepting party may intercept and keep the information, or it may alter the information and send it on to the intended recipient.

- *Impersonation of a particular host* — Using this strategy, an intercepting system pretends to be the intended recipient of a message. If the strategy works, the client remains unaware of the deception and continues to communicate with the interceptor as if its traffic had successfully reached its destination.

Both techniques cause information to be intercepted, possibly for hostile reasons. The results can be disastrous, whether that goal is achieved by listening for all packets on a LAN or a hacked DNS server pointing to a maliciously duplicated host.

If SSH is used for remote shell logins and file copying, these security threats can be greatly diminished. A server's digital signature provides verification for its identity. The entire communication between client and server systems cannot be used if intercepted, because each of the packets is encrypted. Attempts to spoof the identity of either side of a communication will not work, since each packet is encrypted using a key known only by the local and remote systems.

10.2 Event Sequence of an SSH Connection

A certain series of events helps protect the integrity of an SSH communication between two hosts.

First, a secure **transport layer** is created so that the client knows that it is communicating with the correct server. Then, the communication is encrypted between the client and server using a symmetric cipher.

---

[2] Spoofing commonly means appearing to others to be a particular system when you are actually not that system.

Next, with a secure connection to the server in place, the client authenticates itself to the server without worrying that the authentication information may be compromised. OpenSSH on Red Hat Linux uses DSA or RSA keys and version 2.0 of the SSH protocol for authentication by default.

Finally, with the client authenticated to the server, several different services can be safely and securely used through the connection, such as an interactive shell session, X11 applications, and tunneled TCP/IP ports.

The entire connection process occurs with very little extra work required on the local system. In fact, in many respects, SSH works well because it is familiar to users who are accustomed to less secure connection methods.

In the following example, user1 on the client system is initiating an SSH connection to a server. The server's IP address is 10.0.0.2, but its domain name could be used instead. The login name of user1 on the server is user2. The ssh command is written as follows:

```
[user1@machine1 user1]$ ssh user2@10.0.0.2
```

The OpenSSH client will request the user's private key passphrase to decrypt the private key, which is used to perform authentication. However, the private key passphrase is not sent across the now secure connection between the client and server. Instead, the passphrase is used to unlock the id_dsa file and generate a signature, which it then sends to the server. If the server has a copy of the user's public key which can be used to verify the signature, the user is authenticated.

In this example, the user is using a DSA key (RSA keys, among many others, can also be used) and sees the following prompt:

```
Enter passphrase for DSA key '/home/user1/.ssh/id_dsa':
```

If the public key authentication fails for whatever reason (perhaps the passphrase is entered incorrectly or the authentication information does not already exist on the server), another type of authentication is usually attempted. In our example, the OpenSSH server allows user1 to be authenticated using user2's password because the signature sent did not match a public key stored by user2:

```
user2@machine2's password:
```

With a correctly entered password, the user is given a shell prompt. Of course, user2 must already have an account on the 10.0.0.2 machine for password authentication to work.

```
Last login: Mon Apr 15 13:27:43 2001 from machine1
[user2@machine2 user2]$
```

At this point, the user can interact with the shell in the same way as they might do with telnet or rsh, except that the communication is encrypted.

Other SSH tools, scp and sftp, work in a similar way as the insecure rcp and ftp, respectively. See the *Official Red Hat Linux Customization Guide* for instructions and examples for using these and other SSH commands.

10.3 Layers of SSH Security

The SSH protocol allows any client and server programs built to the protocol's specifications to communicate securely and be used interchangeably.

Two different varieties of SSH currently exist. SSH version 1 contains several patented encryption algorithms (however, several of these patents have expired) and a security hole that potentially allows for data to be inserted into the data stream. It is recommended that you use SSH version 2-compatible servers and clients, if at all possible.

OpenSSH includes support for both version 1 and 2. Combined with the OpenSSL encryption libraries, OpenSSH provides a full-range of security capabilities.

Both SSH protocol versions (1 and 2) use similar layers of security to strengthen the integrity of the communication from several different angles. Each layer provides its own type of protection, which when used together with the others, strengthens the overall security of the communication and makes it easier to use.

10.3.1 Transport Layer

The primary role of the transport layer is to facilitate safe and secure communication between the two hosts at the time of and after authentication. Usually running over TCP/IP, the transport layer accomplishes this by handling the encryption and decryption of data, verifying that the server is the correct machine for authentication, and providing integrity protection of data packets as they are sent and received. In addition, the transport layer can also provide compression of the data, effectively speeding the transfer of information.

Once a client contacts a server using the SSH protocol, several important points are negotiated so that the two systems can correctly construct the transport layer:

• Key exchange

• The public key algorithm to be used

• The symmetric encryption algorithm to be used

• The message authentication algorithm to be used

• The hash algorithm to be used

During the key exchange, the server identifies itself to the client with a **host key**. Of course, if this client has never communicated with this particular server before, then the server's key will be unknown to the client. OpenSSH gets around this problem by allowing the client to accept the server's host key the first time an SSH connection occurs. Then, in subsequent connections, the server's host key can be checked with a saved version on the client, providing confidence that the client is indeed communicating with the intended server.

```
                          ┌─────────┐
                          │ CAUTION │
                          └─────────┘
```

CAUTION

The host key verification method used by OpenSSH is not perfect. An at-
tacker could masquerade as the server during the initial contact, as the lo-
cal system would not necessarily know the difference between the intended
server and the attacker at that point. But, until a better host key distribution
method becomes widely available, this initially insecure method is better than
nothing.

SSH is designed to work with almost any kind of public key algorithm or encoding format. After an
initial key exchange creates two values (a hash value used for exchanges and a shared secret value),
the two systems immediately begin calculating new keys and algorithms to protect authentication and
future data sent over the connection.

10.3.2 Authentication

Once the transport layer has constructed a secure tunnel to pass information between the two systems,
the server tells the client the different authentication methods supported, such as using a private key-
encoded signature or typing a password. The client will then try to authenticate itself to the server
using any of the supported methods.

Since servers can be configured to allow different types of authentication, this method gives each side
the optimal amount of control. The server can decide which encryption methods it will support based
on its security model, and the client can choose the order of authentication methods to attempt from
among the available options. Thanks to the secure nature of the SSH transport layer, even seemingly
insecure authentication methods, such as a host-based authentication, are safe to use.

Most users requiring a secure shell will authenticate using a password. Unlike other security authen-
tication schemes, the password is transmitted to the server in cleartext. However, since the entire
password is encrypted when moving over the the transport layer, it can be safely sent across any net-
work.

10.3.3 Connection

After a successful authentication over the SSH transport layer, multiple **channels** are opened by multi-
plexing[3] the single connection between the two systems. Each of these channels handles communica-
tion for a different terminal session, forwarded X11 information, or any other separate service seeking
to use the SSH connection.

---

[3] A multiplexed connection consists of several signals being sent over a shared, common medium. With SSH,
different channels are sent over a common secure connection.

Both clients and servers can create a new channel, with each channel being assigned a different number at each end. When one side attempts to open a new channel, that side's number for the channel is sent along with the request. This information is stored by the other side and used to direct a particular type of service's communication to that channel. This is done so that different types of sessions will not affect one another and channels can be closed without disrupting the primary SSH connection between the two systems.

Channels also support flow-control, which allows them to send and receive data in an orderly fashion. In this way, data is not sent over the channel until the host receives a message that the channel is able to receive it.

Channels are particularly useful with X11 forwarding and TCP/IP port forwarding with SSH. Separate channels can be configured differently, perhaps to use a different maximum packet size or to transfer a particular type of data. This allows SSH to be flexible in handling different types of remote connections, such as dial-up over public networks or high speed LAN links, without having to change the basic infrastructure of the protocol. The client and server negotiate the configuration of each channel within the SSH connection for the user automatically.

10.4 OpenSSH Configuration Files

OpenSSH has two different sets of configuration files, one for the client programs (ssh, scp, and sftp) and the other for the server service (sshd), located in two different areas.

System-wide SSH configuration information is stored in the /etc/ssh directory:

- primes — Contains Diffie-Hellman groups used for the Diffie-Hellman key exchange. Basically, this key exchange creates a shared secret value that cannot be determined by either party alone and is used to provide host authentication. This file is critical for constructing a secure transport layer.

- ssh_config — The system-wide SSH client configuration file used to direct the SSH client. If a user has her own configuration file available in her home directory (~/.ssh/config), then its values will override the values stored in /etc/ssh/ssh_config.

- sshd_config — The configuration file for sshd.

- ssh_host_dsa_key — The DSA private key used by sshd.

- ssh_host_dsa_key.pub — The DSA public key used by sshd.

- ssh_host_key — The RSA private key used by sshd for version 1 of the SSH protocol.

- ssh_host_key.pub — The RSA public key used by sshd for version 1 of the SSH protocol.

- ssh_host_rsa_key — The RSA private key used by sshd for version 2 of the SSH protocol.

- ssh_host_rsa_key.pub — The RSA public key used by sshd for version 2 of the SSH protocol.

User-specific SSH configuration information is stored in the user's home directory within the .ssh subdirectory:

- authorized_keys2 — The file that holds a list of "authorized" public keys. If a connecting user can prove that they know the private key which corresponds to any of these, then they are authenticated. Note, this is only an optional authentication method.

- id_dsa — Contains the DSA authentication identity of the user.

- id_dsa.pub — The DSA public key of the user.

- id_rsa — The RSA public key used by sshd for version 2 of the SSH protocol.

- identity — The RSA private key used by sshd for version 1 of the SSH protocol.

- known_hosts2 — Stores the DSA host keys of the servers a user logs into via SSH when the user elects to record them. If a server has its host keys legitimately altered, perhaps on a re-installation of Red Hat Linux, the user will be notified that the host key stored in the known_hosts2 file that corresponds with this host does not match. Then, the user must delete that host's key in known_hosts in order to store the new host key for that system. The known_hosts2 file is very important for ensuring that the client is connecting the correct server. If a host's key has changed, and you are not absolutely certain why it has changed, then you should contact the host's system administrator to make sure that the host has not be compromised.

See the man pages for ssh and sshd for information concerning the various directives available in the SSH configuration files.

10.5 More Than a Secure Shell

A secure command line interface is just the beginning of the many ways SSH can be used. Given the proper amount of bandwidth, X11 sessions can be directed over an SSH channel. Or, by using TCP/IP forwarding, previously insecure port connections between systems can be mapped to specific SSH channels.

10.5.1 X11 Forwarding

Opening an X11 session over an established SSH connection is as easy as running an X program while already running an X client on your host. When an X program is run from the secure shell prompt, the SSH client and server create a new secure channel within the current SSH connection, and the X program data is sent over that channel to your client machine as if you were connected to the X server via a local terminal.

As you might imagine, X11 forwarding can be very useful. For example, you can use X11 forwarding to create a secure, interactive session with the `up2date` GUI on the server to selectively update packages (if you have the necessary Red Hat Network packages installed on the server). To do this, simply connect to the server using `ssh` and type:

```
up2date
```

You will be asked to supply the root password for the server. Then, the Red Hat Update Agent will appear and you can update your packages on the server as though you were sitting in front of the machine.

The processing overhead required to encrypt and decrypt the secure information being sent over the channel, plus the extra bandwidth necessary to send encrypted X application data, may be significant, however. Adequate testing is required to make sure that the X program is still usable, given your particular hardware and bandwidth conditions.

10.5.2 TCP/IP Forwarding

TCP/IP forwarding works with the SSH client requesting that a particular port on the client or server side be mapped over the existing SSH connection.

To map a local port on the client to a remote port on the server, you first have to know the port numbers on both machines. It is even possible to map two non-standard, different ports to each other.

To create a TCP/IP forwarding channel which listens for connections on the local host, use the following command (all on one line):

```
ssh -L <local-port>:<remote-hostname>:<remote-port>
        <username>@<hostname>
```

Note

Setting up TCP/IP forwarding to listen on ports below 1024 requires root access, just as starting services that listen on ports below 1024.

For example, if you want to check your email on a server called mail.domain.com using POP and SSH is available on that server, you can use this command to set up TCP/IP forwarding:

```
ssh -L 1100:mail.domain.com:110 mail.domain.com
```

After the TCP/IP forwarding is in place between the two machines, you can direct your POP mail client to use localhost as the POP server and 1100 as the port to check for new mail. Any requests sent to port 1100 on your system will be directed securely to the mail.domain.com server.

If mail.domain.com is not running an SSH server daemon but you can log in via SSH to a machine near it, perhaps through a firewall, you can still use SSH to secure the part of the POP connection that occurs over public networks. A slightly different command is needed:

```
ssh -L 1100:mail.domain.com:110 other.domain.com
```

In this example, you are forwarding your POP request from port 1100 on your machine through the SSH connection on port 22 to other.domain.com. Then, other.domain.com connects to port 110 on mail.domain.com to allow you to check for new mail. Only the connection between your system and other.domain.com is secure, but in many situations, this is enough to get your information safely through public networks by providing more security than you had before.

Of course, in this example and the one above it, you must be able authenticate to the SSH server to perform the TCP/IP forwarding. Be sure that you can execute normal SSH commands before attempting to set up TCP/IP forwarding.

TCP/IP forwarding can be particularly useful for getting information securely through network firewalls. If the firewall is configured to allow SSH traffic via its standard port (22) but block access through other ports, a connection between two hosts using the blocked ports is still possible by redirecting their communication over an established SSH connection between them.

Note

This can be very dangerous, however. Using TCP/IP forwarding to forward connections in this manner allows any user on the client system to connect to the service you are forwarding connections to, which can be hazardous if your client system becomes compromised.

Check with the system administrator who administers your firewall before using TCP/IP forwarding to bypass it. System administrators concerned about TCP/IP forwarding can disable this functionality on the server by specifying a No parameter for the **AllowTcpForwarding** line in /etc/ssh/sshd_config and restarting the sshd service.

10.6 Requiring SSH for Remote Connections

For SSH to be truly effective in protecting your network connections, you must stop using all insecure connection protocols, such as telnet and rsh. Otherwise, a user's password may be protected using ssh on one day only to be captured when they log in the next day using telnet.

To disable insecure connection methods to your system, use serviceconf or chkconfig to make sure that these services do not start up with the system. To use serviceconf to configure services that start at runlevels 2, 3, and 5, type the command:

```
/usr/sbin/serviceconf 235
```

Within `serviceconf`, you can disable services from starting up by deselecting them. The [Space-bar] toggles a service between being active or inactive. At a minimum, you should deselect `telnet`, `rsh`, `ftp`, and `rlogin`. When finished, select the **OK** button to save your `serviceconf` changes. See the `serviceconf` man page for additional assistance using this utility.

Changes made to with `serviceconf` will not take affect until either the system is restarted or changes runlevels. If you disabled services used with `xinetd`, you must restart `xinetd`. By default, `rlogin`, `rsh`, and `telnet` are controlled by `xinetd`. To restart `xinetd`, type:

```
/sbin/service xinetd restart
```

For services not used with `xinetd`, you must stop them manually unless you are restart your system after using `serviceconf`. To stop a service, you will probably use a command such as:

```
/sbin/service <service-name> stop
```

After restarting `xinetd` and stopping any other services you have configured not to start up automatically, disabled connection methods will no longer be accepted by your system. If you disable all remote connection methods other than the `sshd` service daemon, users will have to use an SSH client application to connect to the server.

11 Installing and Configuring Tripwire

Tripwire software can help to ensure the integrity of critical system files and directories by identifying all changes made to them. Tripwire configuration options include the ability to receive alerts via email if particular files are altered and automated integrity checking via a `cron` job. Using Tripwire for intrusion detection and damage assessment helps you keep track of system changes and can speed the recovery from a break-in by reducing the number of files you must restore to repair the system.

Tripwire compares files and directories against a baseline database of file locations, dates modified, and other data. It generates the baseline by taking a snapshot of specified files and directories in a known secure state. (For maximum security, Tripwire should be installed and the baseline created before the system is at risk from intrusion.) After creating the baseline database, Tripwire compares the current system to the baseline and reports any modifications, additions, or deletions.

WARNING

While a valuable tool when auditing the security state of your system, Tripwire is not supported by Red Hat, Inc.. Contact Tripwire, Inc., (http://www.tripwire.com) for support options.

11.1 How to Use Tripwire

The following flowchart illustrates how Tripwire should be used:

Figure 11–1   How to Use Tripwire
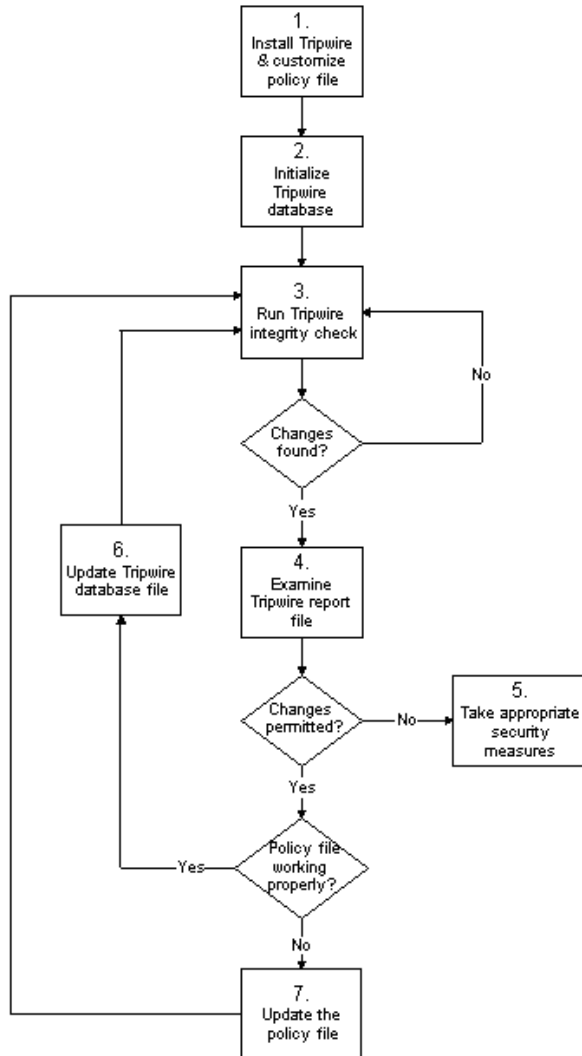
The following steps should be taken to properly install, use and maintain Tripwire:

1. *Install Tripwire and customize the policy file* — If not already done, install the `tripwire` RPM (see Section 11.2.1, *RPM Installation Instructions*). Then, customize the sample configuration (`/etc/tripwire/twcfg.txt`) and policy (`/etc/tripwire/twpol.txt`) files and run the configuration script (`/etc/tripwire/twinstall.sh`). For more information, see Section 11.2.2, *Post-Installation Instructions*.

2. *Initialize the Tripwire database* — Build a database of critical system files to monitor based on the contents of the new, signed Tripwire policy file (`/etc/tripwire/tw.pol`). For more information, see Section 11.7, *Initializing the Database*.

3. *Run a Tripwire integrity check* — Compare the newly-created Tripwire database with the actual system files, looking for missing or altered files. For more information, see Section 11.8, *Running an Integrity Check*.

4. *Examine the Tripwire report file* — View the Tripwire report file using `twprint` to note integrity violations. For more information, see Section 11.9, *Printing Reports*.

5. *Take appropriate security measures —* If monitored files have been altered inappropriately, you can either replace the originals from backups or reinstall the program.

6. *Update the Tripwire database file* — If the integrity violations are intentional and valid, such as if you intentionally edited a file or replaced a particular program, you should tell Tripwire's database file to not report them as violations in future reports. For more information, see Section 11.10, *Updating the Database after an Integrity Check*.

7. *Update the Tripwire policy file* — If you need to change the list of files Tripwire monitors or how it treats integrity violations, you should update your sample policy file (`/etc/tripwire/twpol.txt`), regenerate a signed copy (`/etc/tripwire/tw.pol`), and update your Tripwire database. For more information, see Section 11.11, *Updating the Policy File*.

Refer to the appropriate sections within this chapter for detailed instructions on these steps.

11.2 Installation Instructions

Once installed, Tripwire must also be correctly initialized to be able to keep a close watch on your files. These sections detail how to install the program, if it is not already present on your system, and then how to initialize the Tripwire database.

11.2.1 RPM Installation Instructions

The easiest way to install Tripwire is to install the `tripwire` RPM during the Red Hat Linux 7.2 installation process. However, if you've already installed Red Hat Linux 7.2, you can use RPM, Gnome-RPM, or Kpackage to install the Tripwire RPM from the Red Hat Linux 7.2 CD-ROMs. The following steps outline this process using RPM:

1. Locate the `RedHat/RPMS` directory on the Red Hat Linux 7.2 CD-ROM.

2. Locate the `tripwire` binary RPM by typing `ls -l tripwire*` in the `RedHat/RPMS` directory.

3. Type `rpm -Uvh <name>` (where `<name>` is the name of the Tripwire RPM found in step 2)

4. After installing the `tripwire` RPM, follow the post-installation instructions outlined below.

Note

The release notes and `README` file are located in `/usr/share/doc/tripwire-<version-number>`. These documents contain important information about the default policy file and other issues.

11.2.2 Post-Installation Instructions

The `tripwire` RPM installs the program files needed to run the software. After you've installed Tripwire, you must configure it for your system as outlined in the following steps:

1. If you already know of several changes that should be made to the configuration file (`/etc/tripwire/twcfg.txt`) or the policy file (`/etc/tripwire/twpol.txt`), edit those files now.

Note

While you should edit your configuration and policy files to customize Tripwire to your particular situation, editing the configuration or policy files is not required to use Tripwire. If you plan to modify the configuration or policy files, you must make these changes before running the configuration script (`/etc/tripwire/twinstall.sh`). If you modify the configuration or policy files after running the configuration script, you must re-run the configuration script before initializing the database file. Keep in mind that you *can* edit the configuration and policy files *after* initializing the database file and running an integrity check.

---

2. Type `/etc/tripwire/twinstall.sh` at the command line as root and press [Enter] to run the configuration script. The `twinstall.sh` script walks you through the processes of setting passphrases, generating the cryptographic keys that protect the Tripwire configuration and policy

files, and signing these files. See Section 11.6, *Selecting Passphrases* for more information on setting passphrases.

Note

Once encoded and signed, the configuration file (/etc/trip-wire/tw.cfg) and policy file (/etc/tripwire/tw.pol) generated by running the /etc/tripwire/twinstall.sh script should not be renamed or moved.

---

3. Initialize the Tripwire database file by issuing the /usr/sbin/tripwire --init command at the command line.

4. Run the first integrity check comparing your new Tripwire database to your system files by issuing the /usr/sbin/tripwire --check command at the command line and looking for errors in the generated report.

Once you finish these steps successfully, Tripwire has the baseline snapshot of your filesystem that it needs to check for changes to critical files. Additionally, the tripwire RPM adds a file called tripwire-check to the /etc/cron.daily directory that will automatically run an integrity check once per day.

11.3 File Locations

Before working with Tripwire, you should know where important files for the application are located. Tripwire stores its files in a variety of places depending on their role:

*   The /usr/sbin directory stores the tripwire, twadmin, and twprint programs.

*   The /etc/tripwire directory contains the local and site keys (*.key files) and the initialization script (twinstall.sh), as well as the sample and actual configuration and policy files.

*   The /var/lib/tripwire directory contains the Tripwire database of your system's files (*.twd) and a report directory where Tripwire reports are stored. The Tripwire reports, named *host_name-date_of_report-time_of_report*.twr, detail the differences between the Tripwire database and your actual system files.

11.4 Tripwire Components

The Tripwire policy file is a text file containing comments, rules, directives, and variables. This file dictates the way Tripwire checks your system. Each rule in the policy file specifies a system object to be monitored. Rules also describe which changes to the object to report and which to ignore.

System objects are the files and directories you wish to monitor. Each object is identified by an object name. A property refers to a single characteristic of an object that Tripwire software can monitor. Directives control conditional processing of sets of rules in a policy file. During installation, the text policy file (/etc/tripwire/twpol.txt) is encrypted and renamed, becoming the active policy file (/etc/tripwire/tw.pol).

When first initialized, Tripwire uses the signed policy file rules to create the database file (/var/lib/tripwire/*host_name*.twd). The database file is a baseline snapshot of the system in a known secure state. Tripwire compares this baseline against the current system to determine what changes have occurred. This comparison is called an **integrity check**.

When you perform an integrity check, Tripwire produces report files in the /var/lib/tripwire/report directory. The report files summarize any file changes that violated the policy file rules during the integrity check.

The Tripwire configuration file (/etc/tripwire/tw.cfg) stores system-specific information, such as the location of Tripwire data files. Tripwire generates the necessary configuration file information during installation, but the system administrator can change parameters in the configuration file at any time after that point. Note that the altered configuration file must be signed in the same way as the policy file in order for it to be used by default.

The configuration file variables **POLFILE**, **DBFILE**, **REPORTFILE**, **SITEKEYFILE**, and **LOCALKEY-FILE** specify the locations of the policy file, database file, report files, and site and local key files. These variables are defined by default at the time of installation. If you edit the configuration file and leave any of them undefined, the configuration file will be considered invalid by Tripwire. This causes an error on the execution of tripwire, making the program exit.

Note that the altered configuration file must be signed in the same way as the policy file in order for it to be used by Tripwire. See Section 11.11.1, *Signing the Configuration File* for instructions on signing the configuration file.

11.5 Modifying the Policy File

You can specify how Tripwire checks your system by modifying the Tripwire policy file (twpol.txt). Modifying the policy file to your particular system configuration increases the usefulness of Tripwire reports by minimizing false alerts for files or programs you aren't using but Tripwire is still reporting as altered or missing.

Locate the default policy file at /etc/tripwire/twpol.txt. An example policy file (located at /usr/share/doc/tripwire-<version-number>/policyguide.txt) is included to help you learn the policy language. Read the example policy file for instructions on how to edit the default policy file.

If you modify the policy file immediately after installing the tripwire package, be sure to type /etc/tripwire/twinstall.sh to run the configuration script. This script signs the modified policy file and renames it to tw.pol. This is the active policy file used by the tripwire program when it executes.

If you modify the sample policy file after running the configuration script, see Section 11.11, *Updating the Policy File* for instructions on signing it to make the required tw.pol file.

Note

If you modify the sample policy file, it will not be used by Tripwire until it is signed, encrypted and made into the new /etc/tripwire/tw.pol file (see Section 11.11, *Updating the Policy File*).

11.6 Selecting Passphrases

Tripwire files are signed or encrypted using site and local keys, which protect the configuration, policy, database, and report files from being viewed or altered except by users who know the site and/or local passphrases. This means that, even if an intruder can obtain root access to your system, they will not be able to alter the Tripwire files to hide their tracks unless they also know the passphrases. When selecting passphrases, you must use at least eight alphanumeric and symbolic characters for each passphrase. The maximum length of a passphrase is 1023 characters. Quotes should not be used as passphrase characters. Also, make sure that your passphrases are completely different from the root password for the system.

You should assign unique passphrases for both the site key and the local key. The site key passphrase protects the site key, which is used to sign Tripwire configuration and policy files. The local key signs Tripwire database and report files.

CAUTION

Store the passphrases in a secure location. *There is no way to decrypt a signed file if you forget your passphrase.* If you forget the passphrases, the files are unusable and you will have to run the configuration script again, which also reinitializes the Tripwire database.

11.7 Initializing the Database

When initializing its database, Tripwire builds a collection of filesystem objects based on the rules in the policy file. This database serves as the baseline for integrity checks.

To initialize the Tripwire database, use the following command:

```
/usr/sbin/tripwire --init
```

This command can take several minutes to run.

11.8 Running an Integrity Check

When running an integrity check, Tripwire compares the current, actual filesystem objects with their properties as recorded in its database. Violations are printed to standard output and saved in a report file that can be accessed later by `twprint`. For more information on viewing Tripwire reports, see Section 11.9, *Printing Reports*.

An email configuration option in the policy file even allows particular email addresses to be sent notices when certain integrity violations occur. See Section 11.12, *Tripwire and Email* for instructions on how to set this up.

To run an integrity check, use the following command:

```
/usr/sbin/tripwire --check
```

This command requires some time to run in most situations, depending upon the number of files to be checked.

11.9 Printing Reports

The `twprint -m r` command will display the contents of a Tripwire report in clear text. You must tell `twprint` which report file to display.

A `twprint` command for printing Tripwire reports looks similar to the following (all on one line):

```
/usr/sbin/twprint -m r --twrfile
        /var/lib/tripwire/report/<name>.twr
```

The -m r option in the command tells twprint to decode a Tripwire report. The --twrfile option tells twprint to use a specific Tripwire report file.

The name of the Tripwire report that you want to see includes the name of the host that Tripwire checked to generate the report, plus the creation date and time. You can review previously saved reports at any time. Simply type ls /var/lib/tripwire/report to see a list of Tripwire reports.

Tripwire reports can be rather lengthy, depending upon the number of violations found or errors generated. A sample report starts off like this:

```
Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:        root
Report created on:          Fri Jan 12 04:04:42 2001
Database last updated on:   Tue Jan  9 16:19:34 2001


=====================================================================
Report Summary:
=====================================================================
Host name:                  some.host.com
Host IP address:            10.0.0.1
Host ID:                    None
Policy file used:           /etc/tripwire/tw.pol
Configuration file used:    /etc/tripwire/tw.cfg
Database file used:         /var/lib/tripwire/some.host.com.twd
Command line used:          /usr/sbin/tripwire --check


=====================================================================
Rule Summary:
=====================================================================
---------------------------------------------------------------------
Section: Unix File System
---------------------------------------------------------------------
  Rule Name                 Severity Level    Added   Removed  Modified
  ---------                 --------------    -----   -------  --------
  Invariant Directories     69                0       0        0
  Temporary directories     33                0       0        0
* Tripwire Data Files       100               1       0        0
  Critical devices          100               0       0        0
  User binaries             69                0       0        0
  Tripwire Binaries         100               0       0        0
```

11.9.1 Using `twprint` to View the Tripwire Database

You can also use twprint to view the entire database or information about selected files in the
Tripwire database. This is useful for seeing just how much information Tripwire is tracking on your
system.

To view the entire Tripwire database, type this command:

```
/usr/sbin/twprint -m d --print-dbfile | less
```

This command will generate a large amount of output, with the first few lines appearing similar to this:

```
Tripwire(R) 2.3.0 Database

Database generated by:        root
Database generated on:        Tue Jan  9 13:56:42 2001
Database last updated on:     Tue Jan  9 16:19:34 2001


=================================================================
Database Summary:
=================================================================
Host name:                    some.host.com
Host IP address:              10.0.0.1
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/some.host.com.twd
Command line used:            /usr/sbin/tripwire --init


=================================================================
Object Summary:
=================================================================
-----------------------------------------------------------------
# Section: Unix File System
-----------------------------------------------------------------
     Mode          UID          Size          Modify Time
     ------        ----------   ----------    ----------
 /
     drwxr-xr-x  root (0)      XXX           XXXXXXXXXXXXXXXXX
 /bin
     drwxr-xr-x  root (0)      4096          Mon Jan  8 08:20:45 2001
 /bin/arch
     -rwxr-xr-x  root (0)      2844          Tue Dec 12 05:51:35 2000
 /bin/ash
     -rwxr-xr-x  root (0)      64860         Thu Dec  7 22:35:05 2000
 /bin/ash.static
```

```
          -rwxr-xr-x  root (0)      405576     Thu Dec  7 22:35:05 2000
```

To see information about a particular file that Tripwire is tracking, such as /etc/hosts, type a
different twprint command:

```
    /usr/sbin/twprint -m d --print-dbfile /etc/hosts
```

The result will look similar to this:

```
    Object name:  /etc/hosts

    Property:             Value:
    -------------         -----------
    Object Type           Regular File
    Device Number         773
    Inode Number          216991
    Mode                  -rw-r--r--
    Num Links             1
    UID                   root (0)
    GID                   root (0)
```

See the twprint man page for other options.

11.10 Updating the Database after an Integrity Check

If you run an integrity check and Tripwire finds violations, you will first need to determine whether
the violations discovered are actual security breaches or the product of authorized modifications. If
you recently installed an application or edited critical system files, Tripwire will (correctly) report
integrity check violations. In this case, you should update your Tripwire database so those changes
are no longer reported as violations. However, if unauthorized changes are made to system files that
generate integrity check violations, then you should restore the original file from a backup or reinstall
the program.

To update your Tripwire database to accept the violations found in a report, you must specify the report
you wish to use to update the database. When issuing the command to integrate those valid violations
into your database, be sure to use the most recent report. Type the following command (all on one
line), where *name* is the name of the report to be used:

```
    /usr/sbin/tripwire --update --twrfile
            /var/lib/tripwire/report/<name>.twr
```

Tripwire will show you the particular report using the default text editor (specified in the Tripwire
configuration file on the **EDITOR** line). This is your chance to deselect files that you do not wish to be
updated in the Tripwire database. It is important that you only allow authorized integrity violations to
be changed in the database.

All proposed updates to the Tripwire database start with a `[x]` before the file name. If you want to specifically exclude a valid violation from being added to the Tripwire database, remove the `x` from the box. To accept any files with an `x` beside them as changes, write the file in the editor and quit the text editor. This signals to Tripwire to alter its database and not report these files as violations.

For example, the default text editor for Tripwire is `vi`. To write the file with `vi` and make the changes to the Tripwire database when updating with a specific report, type `:wq` in `vi`'s command mode and press [Enter]. You will be asked to enter your local passphrase. Then, a new database file will be written to include the valid violations.

After a new Tripwire database is written, the newly authorized integrity violations will no longer show up as warnings when the next integrity check is run.

11.11 Updating the Policy File

If you want to actually change the files Tripwire records in its database or modify the severity in which violations are reported, you need to edit your Tripwire policy file.

First, make whatever changes are necessary to the sample policy file (`/etc/tripwire/tw-pol.txt`). A common change to this policy file is to comment out any files that do not exist on your system so that they will not generate a `file not found` error in your Tripwire reports. For example, if your system does not have a `/etc/smb.conf` file, you can tell Tripwire not to try to look for it by commenting out its line in `twpol.txt`:

```
#      /etc/smb.conf                       -> $(SEC_CONFIG) ;
```

Next, you must tell Tripwire to generate a new `/etc/tripwire/tw.pol` signed file and then generate an updated database file based on this policy information. Assuming `/etc/tripwire/tw-pol.txt` is the edited policy file, use this command:

```
/usr/sbin/twadmin --create-polfile -S site.key /etc/tripwire/twpol.txt
```

You will be asked for the site passphrase. Then, the `twpol.txt` file will be parsed and signed.

It is important that you update the Tripwire database after creating a new `/etc/tripwire/tw.pol` file. The most reliable way to accomplish this is to delete your current Tripwire database and create a new database using the new policy file.

If your Tripwire database file is named `wilbur.domain.com.twd`, type this command:

```
rm /var/lib/tripwire/wilbur.domain.com.twd
```

Then type the command to create a new database:

```
/usr/sbin/tripwire --init
```

A new database will be created according to the instructions in the new policy file. To make sure the database was correctly changed, run the first integrity check manually and view the contents of the

resulting report. See Section 11.8, *Running an Integrity Check* and Section 11.9, *Printing Reports* for specific instructions on these points.

11.11.1 Signing the Configuration File

The text file with the configuration file changes (commonly `/etc/tripwire/twcfg.txt`) must be signed to replace the `/etc/tripwire/tw.cfg` and be used by Tripwire when it runs its integrity check. Tripwire will not recognize any configuration changes until the configuration text file is correctly signed and used to replace the `/etc/tripwire/tw.pol` file.

If your altered configuration text file is `/etc/tripwire/twcfg.txt`, type this command to sign it, replacing the current `/etc/tripwire/tw.cfg` file:

```
/usr/sbin/twadmin --create-cfgfile -S site.key /etc/tripwire/twcfg.txt
```

Since the configuration file does not not alter any Tripwire policies or files tracked by the application, it is not necessary to regenerate the database of monitored system files.

11.12 Tripwire and Email

Tripwire can email someone if a specific type of rule in the policy file is violated. To configure Tripwire to do this, you first have to know the email address of the person to be contacted if a particular integrity violation occurs, plus the name of the rule you would like to monitor. Note that on large systems with multiple administrators, you can have different sets of people notified for certain violations and no one notified for minor violations.

Once you know who to notify and what to notify them about, add an **emailto=** line to the rule directive section of each rule. Do this by adding a comma after the **severity=** line and putting **emailto=** on the next line, followed by the email addresses to send the violation reports for that rule. Multiple emails will be sent if more than one email address is specified and they are separated by a semi-colon.

For example, if you would like two administrators, Sam and Bob, notified if a networking program is modified, change the Networking Programs rule directive in the policy file to look like this:

```
(
  rulename = "Networking Programs",
  severity = $(SIG_HI),
  emailto = bob@domain.com;sam@domain.com
)
```

Once a new signed policy file is generated from the `/etc/tripwire/twpol.txt` file, the specified email addresses will be notified upon violations of that particular rule. For instructions on signing your policy file, see Section 11.11, *Updating the Policy File*.

11.12.1 Sending Test Email Messages

To make sure that Tripwire's email notification configuration can actually send email correctly, use the following command:

```
/usr/sbin/tripwire --test --email your@email.address
```

A test email will immediately be sent to the email address by the `tripwire` program.

11.13 Additional Resources

Tripwire can do more than what is covered in this chapter. Refer to these additional sources of information to learn more about Tripwire.

11.13.1 Installed Documentation

- `/usr/share/doc/tripwire-<version-number>` — An excellent starting point for learning about how to customize the configuration and policy files in the `/etc/tripwire` directory.

- Also, refer to the man pages for `tripwire`, `twadmin` and `twprint` for help using those utilities.

11.13.2 Useful Websites

- http://www.tripwire.org — The home of the Tripwire Open Source Project, where you can find the latest news on the application, including an FAQ list.

Part III   Network Services Reference

12   Network Scripts

Using Red Hat Linux, all network communications occur between **interfaces**, which are networking devices connected to the system, configured in a particular way, and utilizing at least one protocol to exchange data with other systems. The different types of interfaces that exist are as varied as the devices that support them, such as **network interface cards** (NICs — commonly called Ethernet cards), wireless networking PC cards, infrared ports, and more.

The configuration files for the various network interfaces and the scripts to make them active and inactive are located in the /etc/sysconfig/network-scripts directory. While the existence of particular interface files can differ from system to system depending on use, the three different types of files that exist in this directory, **interface configuration files**, **interface control scripts**, and **network function files**, work together to enable Red Hat Linux to use the various network devices available.

This chapter will explore the relationship between these files and different options for their use.

12.1 Interface Configuration Files

Interface configuration files control the operation of a particular network interface device. As your Red Hat Linux system boots, it uses these files to know what interfaces to bring up automatically and how to configure them to operate correctly. These files are usually named ifcfg-<device>, where <device> refers to the name of the device that the configuration file controls.

12.1.1 Ethernet Interfaces

One of the most common interface files is ifcfg-eth0, which controls the first NIC in a system. In a system with many NICs, you will have multiple ifcfg-eth files, each one with number at the end of the file name. Because each device has its own configuration file, you have a great deal of control over how each interface functions.

A sample ifcfg-eth0 for a system using a fixed IP address looks similar to this:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
BROADCAST=10.0.1.255
NETWORK=10.0.1.0
NETMASK=255.255.255.0
IPADDR=10.0.1.27
USERCTL=no
```

The values required in an interface configuration file can change based on other values. For example, the `ifcfg-eth0` file for an interface utilizing DHCP looks quite a bit different, due to the fact that IP information is now provided by the DHCP server:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

Most of the time, you will probably want to use a GUI utility, such as Network Configurator (`red-hat-config-network`) or `netconfig` to make changes to the various interface configuration files. See the *Official Red Hat Linux Customization Guide* for instructions on using these tools.

Within each of the interface configuration files, the following values are common:

- `BOOTPROTO=<protocol>`, where `<protocol>` is one of the following:

  - `none` — No boot-time protocol should be used.

  - `bootp` — The BOOTP protocol should be used.

  - `dhcp` — The DHCP protocol should be used.

- `BROADCAST=<address>`, where `<address>` is the broadcast address.

- `DEVICE=<name>`, where `<name>` is the name of the physical device (except dynamically-allocated PPP devices where it is the logical name).

- `IPADDR=<address>`, where `<address>` is the IP address.

- `NETMASK=<mask>`, where `<mask>` is the netmask value.

- `NETWORK=<address>`, where `<address>` is the network address.

- `ONBOOT=<answer>`, where `<answer>` is one of the following:

  - `yes` — This device should be activated at boot-time.

  - `no` — This device should not be activated at boot-time.

- `USERCTL=<answer>`, where `<answer>` is one of the following:

  - `true` — Non-root users are allowed to control this device.

  - `false` — Non-root users are not allowed to control this device.

Other common interface configuration files that use these options include `ifcfg-lo`, which controls the IP protocol's local loopback device, `ifcfg-irlan0`, which arranges settings for the first infrared device, `ifcfg-plip0`, which controls the first PLIP device, and `ifcfg-tr0`, used with the first Token Ring device.

A **local loopback interface** is often used in testing, as well as a variety of applications that require an IP address pointing back to the same system. Any data sent to the loopback device is immediately returned to the host's network layer. The infrared interface allows information between devices, such as a laptop and a printer, to flow over an infrared link, which works in a similar way to an Ethernet device except that it commonly occurs over a peer-to-peer connection. A **Parallel Line Interface Protocol (PLIP)** connection works much the same way, except that it utilizes a parallel port. **Token Ring** topologies are not as common on Local Area Networks as they once were, having been eclipsed by Ethernet.

12.1.2 Dialup Interfaces

If you connect to a network, such as the Internet, via PPP dialup connection, you need a configuration file for that interface. You can have multiple PPP interface configuration files for a particular device, such as a modem. A sample `ifcfg-ppp0` files looks like this:

```
DEVICE=ppp0
NAME=test
WVDIALSECT=test
MODEMPORT=/dev/modem
LINESPEED=115200
PAPNAME=test
USERCTL=true
ONBOOT=no
PERSIST=no
DEFROUTE=yes
PEERDNS=yes
DEMAND=no
IDLETIMEOUT=600
```

This file is created automatically for you when you use RP3 or Kppp to create a dialup account. In addition, any changes to dialup account settings are reflected in these interface configuration files. The *Official Red Hat Linux Getting Started Guide* contains instructions for using these GUI-based dialup connection tools. You can also create and edit this file manually.

Serial Line Internet Protocol (SLIP) is another dialup interface, although it is less commonly used. SLIP files have interface configuration file names such as `ifcfg-sl0`.

Among the options not already discussed that may be used in these files:

- `DEFROUTE=<answer>`, where `<answer>` is one of the following:

    - `yes` — Set this interface as the default route.

    - `no` — Do not set the interface as the default route.

- `DEMAND=<answer>`, where `<answer>` is one of the following:

- *yes* — This interface will allow `pppd` to initiate a connection when someone attempts to use it.

- *no* — A connection must be manually established for this interface.

- `IDLETIMEOUT=`*`<value>`*, where *`<value>`* is number of seconds of idle activity before the interface will disconnect itself.

- `INITSTRING=`*`<string>`*, where *`<string>`* is the init string passed to the modem device. This option is primarily used with SLIP interfaces.

- `LINESPEED=`*`<value>`*, where *`<value>`* is the baud rate of the device. Possible standard values here include `57600`, `38400`, `19200`, and `9600`, among others.

- `MODEMPORT=`*`<device>`*, where *`<device>`* is the name of the device (usually a modem) that is used to establish the connection for the interface.

- `MTU=`*`<value>`*, where *`<value>`* is the **Maximum Transfer Unit (MTU)** setting for the interface. The MTU refers to the largest number of bytes of data a frame can carry, not counting its header and trailer information. In some dialup situations, setting this to a value of `576` will result in fewer dropped packets and slightly improve the throughput for a connection.

- `NAME=`*`<name>`*, where *`<name>`* is the reference to the title given to a collection of dialup connection configurations.

- `PAPNAME=`*`<name>`*, where *`<name>`* is the username given during the **Password Authentication Protocol (PAP)** exchange that occurs to allow you to connect to a remote system.

- `PEERDNS=`*`<answer>`*, where *`<answer>`* is one of the following:

  - *yes* — This interface will modify your system's `/etc/resolv.conf` file entries to use the DNS servers provided by the remote system when a connection is established.

  - *no* — The `/etc/resolv.conf` file will not be changed.

- `PERSIST=`*`<answer>`*, where *`<answer>`* is one of the following:

  - *yes* — This interface should be kept active at all times, even if deactivated after a modem hang up.

  - *no* — This interface should not be kept active at all times.

- `REMIP=`*`<address>`*, where *`<address>`* is the remote system's IP address. This is usually left unspecified.

- `WVDIALSECT=`*`<name>`*, where *`<name>`* associates this interface with a dialer configuration in `/etc/wvdial.conf`, which contains the phone number to be dialed and other important information for the interface.

12.1.3 Alias and Clone Files

Two lesser used types of interface configuration files found in `/etc/sysconfig/net-work-scripts` are **alias** and **clone** files, which include an additional component in the name of the file beyond the interface name.

Alias interface configuration files take names in the format of `ifcfg-<if-name>:<alias-value>`, and they allow an alias to point to an interface. For example, a `ifcfg-eth0:0` file could be configured to specify `DEVICE=eth0:0` and a static IP address of 10.0.0.2, serving as an alias of an Ethernet interface already configured to receive its IP information via DHCP in `ifcfg-eth0`. At that point, the `eth0` device is bound to a dynamic IP address, but it can always be referred to on that system via the fixed 10.0.0.2 IP address.

A clone interface configuration file has a name similar to `ifcfg-<if-name>-<clone-name>`. While an alias file is another way to refer to an existing interface configuration file, a clone file is used to specify additional options when specifying an interface. For example, if you have a standard DHCP Ethernet interface called `eth0`, it may look similar to this:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Since `USERCTL` is not set to `yes`, users cannot bring this interface up and down. To give users this ability, create a clone called `user` from `ifcfg-eth0` which allows a user to bring up and down the `eth0` interface. The resulting name of the clone would be `ifcfg-eth0-user` and would only need one line:

```
USERCTL=yes
```

When a user moves to bring up the `eth0` interface with the `ifup eth0-user` command, the configuration options from `ifcfg-eth0` and `ifcfg-eth0-user` are used together. While this is a very basic example, this method can be used with a variety of options and interfaces.

The easiest way to create alias and clone interface configuration files is to use the GUI-based Network Configurator (`redhat-config-network`) tool.

12.2 Interface Control Scripts

The interface control scripts control bringing up (activating) and down (deactivating) interface connections. There are two primary interface control scripts, `/sbin/ifdown` and `/sbin/ifup`, that use assorted other control scripts located in the `/etc/sysconfig/network-scripts` directory to actually bring the network interfaces up and down.

The two primary interface control scripts in the `/etc/sysconfig/network-scripts` directory, `ifdown` and `ifup`, are symbolic links to the scripts in the `/sbin` directory. When either of these scripts are called, they accept a value of the interface to be used, such as:

```
[root@bleach network-scripts]# ifup eth0
Determining IP information for eth0... done.
[root@bleach network-scripts]#
```

At that point, the `/etc/sysconfig/network-scripts/network-functions` and `/etc/rc.d/init.d/functions` files are sourced and functions from those files are used to perform a variety of tasks. See Section 12.3, *Network Functions* for more information about these tasks.

After verifying that an interface has been specified and that the user executing the request is allowed to bring the interface up or down, the correct script for the type of interface device is called into action. This is the script that actually brings the interface up and down. The following interface control scripts are the most common of this type:

- `ifup-aliases` — Configures IP aliases from interface configuration files when more than one IP address is associated with an interface.

- `ifdown-cipcb` and `ifup-cipcb` — Used to bring **Crypto IP Encapsulation (CIPE)** connections up and down.

- `ifdown-ipv6` and `ifup-ipv6` — Contains IPv6-related function calls that utilize the environment variables in various interface configuration files and `/etc/sysconfig/network`.

- `ifup-ipx` — Used to bring up an IPX interface.

- `ifup-plip` — Used to bring up a PLIP interface.

- `ifup-plusb` — Used to bring up a USB interface for network connections.

- `ifdown-post` and `ifup-post` — Contains commands to be executed after a particular interface is brought up or down, such as configuring IP aliases or routes by calling certain scripts, such as `ifup-aliases` or `ifup-routes`.

- `ifdown-ppp` and `ifup-ppp` — Used to bring a PPP interface up or down using a particular device.

- `ifup-routes` — Adds static routes for a particular device as its interface is brought up.

- `ifdown-sit` and `ifup-sit` — Contains function calls related to bringing up and down an IPv6 tunnel within an IPv4 connection.

- `ifdown-sl` and `ifup-sl` — Used to bring an SLIP interface up or down.

In order to bring a particular interface up or down properly, these scripts may call one another to provide a certain type of functionality. In addition, other scripts will find their way into the /etc/syscon-fig/network-scripts directory as software is installed that requires a new type of interface. Be aware that removing or modifying these scripts can cause various interface connections to act strangely or fail, as these scripts tend to rely on each other. However, advanced users can modify scripts linked with a specific interface to make additional steps occur when that interface is brought up or down.

You can also use the init script /etc/rc.d/init.d/network to activate and deactive all network interfaces configured to start at boot time with the command:

```
/sbin/service network action
```

where *action* is either start to start the network interfaces, stop to stop the network interfaces, or restart to restart the network interfaces. You can also use the command /sbin/service/net-work status to view a list of configured devices and currently active devices.

Instructions for making interfaces active and inactive, through the use of the command line or GUI-based administration tools, can be found in the *Official Red Hat Linux Customization Guide*.

12.3 Network Functions

Red Hat Linux makes use of several files that contain important functions that are used in various ways to bring interfaces up and down. Rather than forcing each interface control file to contain the same functions as another, these functions are conveniently grouped together in a few files that can be sourced when needed. If you are altering your interface control files for particular reasons, it may be well worth it to also group common functions in these files.

The most common network functions file is network-functions, located in the /etc/sysconfig/network-scripts directory. This file contains a variety of com-mon IPv4 functions useful to many interface control scripts, such as contacting running programs that have requested information about changes in an interface's status, setting host names, finding a gateway device, seeing if a particular device is down or not, and adding a default route.

As the functions required for IPv6 interfaces are different than IPv4 interfaces, a network-func-tions-ipv6 file exists specifically to hold this information. IPv6 support must be enabled in the kernel in order to communicate via that protocol. A function is present in this file that checks for the presence of IPv6 support. Additionally, functions that configure and delete static IPv6 routes, create and remove tunnels, add and remove IPv6 addresses to an interface, and test for the existence of an IPv6 address on an interface can also be found in this file.

13 Apache

The Apache product includes software developed by the Apache Software Foundation (http://www.apache.org).

The Apache HTTP server is a robust and commercial-grade open source Web server used by the majority of websites on the Internet. The Red Hat Linux distribution includes Apache, as well as a number of additional modules which are designed to enhance the functionality of the server and add strong encryption capabilities. Apache's default configuration should work for most users. You may never need to change any of Apache's configuration directives. If you do want to change any of the default configuration options, you will need to know what some of the options are and where to find them. This chapter covers how to use and configure the Apache Web server.

WARNING

If you plan to use the **Apache Configuration Tool** (`apacheconf`), a GUI utility provided with Red Hat Linux, *do not* edit your Apache Web server's `httpd.conf` configuration file. Conversely, if you want to edit `httpd.conf` by hand, *do not* use the **Apache Configuration Tool**.

If you need more information on the **Apache Configuration Tool**, please see the *Official Red Hat Linux Customization Guide*.

After you have installed the `apache` package, the Apache Web server's documentation is available at http://your_domain/manual/ or you can use the Apache documentation available on the Web at http://httpd.apache.org/docs/. The Apache Web server's documentation contains a full list and complete descriptions of all of Apache's configuration options. For your convenience, this chapter provides short descriptions of the configuration directives used in the version of Apache provided with Red Hat Linux.

The version of Apache included with Red Hat Linux includes the ability to set up secure Web servers using the strong SSL encryption provided by the mod_ssl and OpenSSL packages. When you are reading your Web server's configuration file, be aware that it includes both a non-secure and a secure Web server. The secure Web server runs as a virtual host, which is also configured in the `httpd.conf` file. For more information about virtual hosts, see Section 13.5, *Using Virtual Hosts*.

Note

We do not include FrontPage extensions. The Microsoft™ license prohibits
the inclusion of the extensions in a third party product. To find out more
about FrontPage extensions, refer to http://www.rtr.com/fpsupport/.

13.1 Default Modules

Apache is distributed with a number of modules. By default the following modules are installed and
enabled with the Apache package on Red Hat Linux:

```
mod_vhost_alias
mod_env
mod_log_config
mod_log_agent
mod_log_referer
mod_mime
mod_negotiation
mod_status
mod_info
mod_include
mod_autoindex
mod_dir
mod_cgi
mod_asis
mod_imap
mod_actions
mod_userdir
mod_alias
mod_rewrite
mod_access
mod_auth
mod_auth_db
mod_expires
mod_headers
mod_setenvif
mod_so
```

The following modules are installed but disabled (see Section 13.4, *Adding Modules to Your Server*):

```
mod_mmap_static
mod_mime_magic
mod_speling
mod_auth_anon
```

```
mod_define
mod_auth_dbm
mod_auth_db
mod_digest
mod_proxy
mod_cern_meta
mod_usertrack
mod_example
mod_unique_id
```

The following modules are available by installing additional packages:

```
mod_ssl
mod_auth_any
mod_auth_mysql
mod_auth_pgsql
mod_bandwidth
mod_dav
mod_perl
mod_php4
mod_put
mod_python
mod_roaming
mod_throttle
```

13.2 Starting and Stopping `httpd`

During the installation process, a Bourne shell script named `httpd` was saved in `/etc/rc.d/init.d`. To manually stop and start your server, run `httpd` with either `stop` or `start` as an argument.

To start your server, type the command:

```
/etc/rc.d/init.d/httpd start
```

If you are running Apache as a secure server, you will be prompted to fill in your password. After you type it in, your server will start.

To stop your server, type the command:

```
/etc/rc.d/init.d/httpd stop
```

The command `restart` is a shorthand way of stopping and then starting your server. The `restart` command explicitly stops and then starts your server. You will be prompted for your password if you are running Apache as a secure server. The `restart` command looks like the following:

```
/etc/rc.d/init.d/httpd restart
```

If you just finished editing something in your `httpd.conf` file, you do not need to explicitly stop and start your server. Instead, you may use the `reload` command. When you use `reload`, you will not need to type in your password. Your password will remain cached across reloads, but it will not be cached between stops and starts. The `reload` command looks like the following:

```
/etc/rc.d/init.d/httpd reload
```

By default, the `httpd` process will start automatically when your machine boots. If you are running Apache as a secure server, you will be prompted for the secure server's password after the machine boots, unless you generated a key for your secure server without password protection.

13.3 Configuration Directives in `httpd.conf`

The Apache Web server configuration file is `/etc/httpd/conf/httpd.conf`. The `httpd.conf` file is well-commented and somewhat self-explanatory. Its default configuration will work for most people, so you should not need to change the directives in `httpd.conf`. However, you may want to be familiar with the most important configuration options.

The empty `srm.conf` and `access.conf` files are also in the `/etc/httpd/conf` directory. The `srm.conf` and `access.conf` files were formerly used, along with `httpd.conf`, as configuration files for Apache.

If you need to configure Apache, edit `httpd.conf` and then either reload, or stop and start the `httpd` process. How to reload, stop and start Apache is covered in Section 13.2, *Starting and Stopping `httpd`*.

Before you edit `httpd.conf`, you should first copy the original file to something like `httpd.confold`, for example. By creating a backup, you can recover potential mistakes made while editing the new configuration file.

If you do make a mistake, and your Web server does not work correctly, first review what you have recently edited in `httpd.conf`. Make sure that you did not make a typo. The next place to look is your Web server's error log (`/var/log/httpd/error_log`). The error log may not be easy to interpret, depending on your level of experience. If you have just experienced a problem, however, the last entries in the error log should provide information about what has happened.

The next sections provide short descriptions of the directives included in `httpd.conf`. These descriptions are not exhaustive. If you need more information, refer to the Apache documentation provided in HTML format at http://*your_domain*/manual/ or to the Apache group documentation at http://httpd.apache.org/docs/. For more information about mod_ssl directives, refer to the documentation included in HTML format at http://*your_domain*/manual/mod/mod_ssl/, or see the mod_ssl *User Manual* at http://www.modssl.org/docs/2.7/.

13.3.1 `ServerType`

Your `ServerType` must be set to `standalone`. By default, your Web server is set to `Server-Type standalone`.

`ServerType standalone` means that the server is started once and then that server handles all of the connections.

13.3.2 `ServerRoot`

The `ServerRoot` is the top-level directory which contains the server's files. Both your secure and non-secure servers are set to use a `ServerRoot` of `/etc/httpd`.

13.3.3 `LockFile`

`LockFile` sets the path to the lockfile used when the Apache server is compiled with either `USE_FCNTL_SERIALIZED_ACCEPT` or `USE_FLOCK_SERIALIZED_ACCEPT`. `LockFile` should be left at its default value.

13.3.4 `PidFile`

`PidFile` names the file where the server records its process ID (pid). Your Web server is set to record its pid in `/var/run/httpd.pid`.

13.3.5 `ScoreBoardFile`

The `ScoreBoardFile` stores internal server process information, which is used for communication between the parent server process and its child processes. Red Hat Linux uses shared memory to store the `ScoreBoardFile`, the default of `/var/run/httpd.scoreboard` is only used as a fallback.

13.3.6 `ResourceConfig`

The `ResourceConfig` directive instructs the server to read the file named after `ResourceConfig` for more directives. The `ResourceConfig` directive is commented out, because your Web server only uses `httpd.conf` for configuration directives.

13.3.7 `AccessConfig`

The `AccessConfig` directive instructs the server to read the file named after `AccessConfig` for more directives, after it has read the file named by `ResourceConfig`. The `AccessConfig` directive is commented out, because your Web server only uses `httpd.conf` for configuration directives.

13.3.8 `Timeout`

`Timeout` defines, in seconds, the amount of time that your server will wait for receipts and transmissions during communications. Specifically, `Timeout` defines how long your server will wait to receive a GET request, how long it will wait to receive TCP packets on a POST or PUT request and how long it will wait between ACKs responding to TCP packets. `Timeout` is set to 300 seconds, which is appropriate for most situations.

13.3.9 `KeepAlive`

`KeepAlive` sets whether your server will allow more than one request per connection (in other words, it will allow persistent connections). `KeepAlive` can be used to prevent any one client from consuming too much of the server's resources.

By default Keepalive is set to off, which means that your server does not allow persistant connections. You could set it to on, which would enable persistant connections. In most cases you want Keepalive set to off because Apache can only have a limited number of child processes and these processes would be hanging around doing nothing waiting for a possible future connection from the same client. If you do enable Keepalives you should set the the KeepAliveTimeout (see Section 13.3.11, *KeepAlive-Timeout*) low and monitor your servers error_log which will warn you if you are running out of child processes to handle requests.

13.3.10 `MaxKeepAliveRequests`

This directive sets the maximum number of requests allowed per persistent connection. The Apache Group recommends a high setting, which will improve your server's performance. `MaxKeepAliveRequests` is set to `100` by default, which should be appropriate for most situations.

13.3.11 `KeepAliveTimeout`

`KeepAliveTimeout` sets the number of seconds your server will wait for a subsequent request, after a request has been served, before it closes the connection. Once a request has been received, the `Timeout` directive applies instead.

13.3.12 `MinSpareServers` and `MaxSpareServers`

The Apache Web server dynamically adapts to the perceived load by maintaining an appropriate number of spare server processes based on the traffic. The server checks the number of servers waiting for a request and kills some if there are more than `MaxSpareServers` or creates some if the number of servers is less than `MinSpareServers`.

Your server's default `MinSpareServers` is 5; your server's default `MaxSpareServers` is 20. These default settings should be appropriate in most situations. You should not increase the `MinS-pareServers` to a large number. Doing so will create a heavy processing load on your server even when traffic is light.

13.3.13 `StartServers`

`StartServers` sets how many server processes are created upon startup. Since your Web server dynamically kills and creates server processes based on traffic load, you will not need to change this parameter. Your Web server is set to start eight server processes at startup.

13.3.14 `MaxClients`

`MaxClients` sets a limit on the total number of server processes (i.e., simultaneously connected clients) that can run at one time. You should keep `MaxClients` at a high number (your server's default is set to 150), because no one else will be allowed to connect once that number of simultaneously connected clients is reached. You can not set `MaxClients` to higher than 256 without recompiling Apache. The main reason for having `MaxClients` is to keep a runaway Web server from crashing your operating system.

13.3.15 `MaxRequestsPerChild`

`MaxRequestsPerChild` sets the total number of requests each child server process serves before the child dies. The main reason for setting `MaxRequestsPerChild` is to avoid long-lived process induced memory leaks. The default `MaxRequestsPerChild` for your server is `1000`.

13.3.16 `Listen`

The `Listen` command identifies the ports on which your Web server will accept incoming requests. Your Web server is set to listen to port 80 for non-secure Web communications and (in the virtual host tags that define the secure server) to port 443 for secure Web communications.

If you set Apache to listen to a port under 1024, you must be root to start it. For port 1024 and above, `httpd` can be started as a regular user.

`Listen` can also be used to specify particular IP addresses over which the server will accept connections.

13.3.17 `BindAddress`

`BindAddress` is a way of specifying which IP addresses your server will listen to. You should use the `Listen` directive instead if you need this functionality. `BindAddress` is not used by your Web server; by default it is commented out in `httpd.conf`.

13.3.18 `LoadModule`

`LoadModule` is used to load in Dynamic Shared Object (DSO) modules. More information on the Apache's DSO support, including exactly how to use the `LoadModule` directive, can be found in Section 13.4, *Adding Modules to Your Server*. Note, the order of the modules is important, so do not move them around.

13.3.19 `IfDefine`

The `<IfDefine>` and `</IfDefine>` tags surround configuration directives that are applied if the "test" stated in the `<IfDefine>` tag is true. The directives are ignored if the test is false.

The test in the `<IfDefine>` tags is a parameter name (for example, HAVE_PERL). If the parameter is defined, meaning that it is provided as an argument to the server's start-up command, then the test is true. In this case, when your Web server is started, the test is true and the directives contained in the `IfDefine` tags are applied.

By default, `<IfDefine HAVE_SSL>` tags surround the virtual host tags for your secure server. `<IfDefine HAVE_SSL>` tags also surround the `LoadModule` and `AddModule` directives for the `ssl_module`.

13.3.20 `ClearModuleList`

The `ClearModuleList` directive is located immediately before the long list of `AddModule` directives. `ClearModuleList` erases the server's built-in list of active modules. Then the list of `AddModule` directives re-creates the list, immediately after `ClearModuleList`.

13.3.21 `AddModule`

`AddModule` is the directive used to create a complete list of all available modules. You will use the `AddModule` directive if you add your own module in as a DSO. For more information on how `AddModule` is used for DSO support, see Section 13.4, *Adding Modules to Your Server*.

13.3.22 `ExtendedStatus`

The `ExtendedStatus` directives controls whether Apache generates basic (`off`) or detailed server status information (`on`), when the `server-status` handler is called. `Server-status` is called using `Location` tags. More information on calling `server-status` is included in Section 13.3.71, *Location* .

13.3.23 `Port`

Normally, `Port` defines the port that your server is listening to. Your Web server, however, is listening to more than one port by default, since the `Listen` directive is also being used. When `Listen`

directives are in effect, your server listens at all of those ports. See the description of the `Listen` directive for more information about `Listen`.

The `Port` command is also used to specify the port number used to construct a canonical name for your server. See Section 13.3.39, *UseCanonicalName* for more information about your server's canonical name.

13.3.24 `User`

The `User` directive sets the userid used by the server to answer requests. `User`'s setting determines the server's access. Any files inaccessible to this user will also be inaccessible to your website's visitors. The default for `User` is `apache`.

The `User` should only have privileges so that it can access files which are supposed to be visible to the outside world. The `User` is also the owner of any CGI processes spawned by the server. The `User` should not be allowed to execute any code which is not intended to be in response to HTTP requests.

Note

For security reasons, Apache will refuse to run as `User  root`. Using root as the `User` will create large security holes for your Web server.

The parent `httpd` process first runs as root during normal operations but is then immediately handed off to the apache user. The server must start as root because it needs to bind to a port below 1024 (the default port for secure Web communications is port 443; the default port for non-secure Web communications is port 80). Ports below 1024 are reserved for system use, so they can not be used by anyone but root. Once the server has attached itself to its port, however, it hands the process off to the `User` before it accepts any connection requests.

13.3.25 `Group`

The `Group` directive is similar to the `User`. The `Group` sets the group under which the server will answer requests. The default `Group` is `apache`.

13.3.26 `ServerAdmin`

`ServerAdmin` should be the email address of the Web server's administrator. This email address will show up in error messages on server-generated Web pages, so users can report a problem by sending email to the server administrator. `ServerAdmin` is set by default to `root@localhost`.

Typically, a good way to set up `ServerAdmin` is to set it to webmaster@your_domain.com. Then alias `webmaster` to the person responsible for the Web server in `/etc/aliases`. Finally, run `/usr/bin/newaliases` to add the new alias.

13.3.27 `ServerName`

You can use `ServerName` to set a hostname for your server which is different from your host's real name. For example, you might want to use www.your_domain.com when your server's real name is actually foo.your_domain.com. Note that the `ServerName` must be a valid Domain Name Service (DNS) name that you have the right to use (do not just make something up).

If you do specify a `ServerName`, be sure its IP address and server name pair are included in your `/etc/hosts` file.

13.3.28 `DocumentRoot`

The `DocumentRoot` is the directory which contains most of the HTML files which will be served in response to requests. The default `DocumentRoot` for both the non-secure and secure Web servers is `/var/www/html`. For example, the server might receive a request for the following document:

```
http://your_domain/foo.html
```

The server will look for the following file in the default directory:

```
/var/www/html/foo.html
```

If you want to change the `DocumentRoot` so that it is not shared by the secure and the non-secure Web servers, see Section 13.5, *Using Virtual Hosts*.

13.3.29 `Directory`

`<Directory /path/to/directory>` and `</Directory>` tags are used to enclose a group of configuration directives that are meant to apply only to that directory and all of its subdirectories. Any directive which is applicable to a directory may be used within `<Directory>` tags. `<File>` tags can be used in the same way, to apply to a specific file or files.

By default, very restrictive parameters are applied to the root directory, using the `Options` (see Section 13.3.30, *Options*) and `AllowOverride` (see Section 13.3.31, *AllowOverride*) directives. Under this configuration, any directory on your system which needs more permissive settings has to be explicitly given those settings.

Using `Directory` tags, the `DocumentRoot` is defined to have less rigid parameters, so that HTTP requests can be served from it.

The `cgi-bin` directory is set up to allow the execution of CGI scripts, with the `ExecCGI` option. If you need to execute a CGI script in another directory, you will need to set `ExecCGI` for that directory. For example, if your `cgi-bin` is `/var/www/cgi-bin`, but you want to execute CGI scripts from within `/home/my_cgi_directory`, add an `ExecCGI` directive to a set of `Directory` directives like the following to your `httpd.conf` file:

```
<Directory /home/my_cgi_directory>
    Options +ExecCGI
</Directory>
```

To allow CGI script execution in `/home/my_cgi_directory`, you will need to take a few extra steps besides setting `ExecCGI`. You will also need to have the `AddHandler` directive uncommented to identify files with the `.cgi` extension as CGI scripts. See Section 13.3.65, *AddHandler* for instructions on setting `AddHandler`. Permissions for CGI scripts, and the entire path to the scripts, must be set to 0755.

13.3.30 `Options`

The `Options` directive controls which server features are available in a particular directory. For example, under the restrictive parameters specified for the root directory, `Options` is set to only `FollowSymLinks`. No features are enabled, except that the server is allowed to follow symbolic links in the root directory.

By default, in your `DocumentRoot` directory, `Options` is set to include `Indexes`, `Includes` and `FollowSymLinks`. `Indexes` permits the server to generate a directory listing for a directory if no `DirectoryIndex` (for example, `index.html`) is specified. `Includes` means that server-side includes are permitted. `FollowSymLinks` allows the server to follow symbolic links in that directory.

You will also need to include `Options` statements for directories within virtual hosts directives, if you want your virtual hosts to recognize those `Options`.

For example, server side includes are already enabled inside the `/var/www/html` directory, because of the `Options Includes` line within the `<Directory "/var/www/html">` directives section. However, if you want a virtual host to recognize that server side includes are allowed within `/var/www/html`, you will need to include a section like the following within your virtual host's tags:

```
<Directory /var/www/html>
Options Includes
</Directory>
```

13.3.31 `AllowOverride`

The `AllowOverride` directive sets whether or not any `Options` can be overridden by the declarations in an `.htaccess` file. By default, both the root directory and the `DocumentRoot` are set to allow no `.htaccess` overrides.

13.3.32 `Order`

The `Order` directive simply controls the order in which `allow` and `deny` directives are evaluated. Your server is configured to evaluate the `Allow` directives before the `deny` directives for your `Doc-umentRoot` directory.

13.3.33 `Allow`

`Allow` specifies which requester can access a given directory. The requester can be `all`, a domain name, an IP address, a partial IP address, a network/netmask pair, etc. Your `DocumentRoot` directory is configured to `Allow` requests from `all` (i.e., anyone).

13.3.34 `Deny`

`Deny` works just like `Allow`, but you are specifying who is denied access. Your `DocumentRoot` is not configured to `Deny` requests from anyone.

13.3.35 `UserDir`

`UserDir` is the name of the subdirectory within each user's home directory where they should place personal HTML files which are to be served by the Web server.

By default, the subdirectory is `public_html`. For example, the server might receive the following request:

        http://*your_domain*/~*username*/foo.html

The server would look for the file:

        /home/username/public_html/foo.html

In the above example, `/home/username` is the user's home directory (note that the default path to users' home directories may be different on your system).

Make sure that the permissions on the users' home directories are set correctly. Users' home directories must be set to 0711. The read (r) and execute (x) bits must be set on the users' `public_html` directories (0755 will work). Files that will be served in users' `public_html` directories must be set to at least 0644.

13.3.36 `DirectoryIndex`

The `DirectoryIndex` is the default page served by the server when a user requests an index of a directory by specifying a forward slash (/) at the end of the directory name.

When a user requests the page http://*your_domain*/*this_directory*/, they will get either the `Directo-ryIndex` page if it exists, or a server-generated directory list. The default for `DirectoryIndex`

is index.html index.htm index.shtml index.php index.php4 index.php3 in-dex.cgi. The server will try to find any one of these files, and will return the first one it finds. If it does not find any of these files and `Options Indexes` is set for that directory, the server will generate and return a listing, in HTML format, of the subdirectories and files in the directory.

13.3.37 `AccessFileName`

`AccessFileName` names the file which the server should use for access control information in each directory. By default, your Web server is set to use `.htaccess`, if it exists, for access control information in each directory.

Immediately after the `AccessFileName` directive, a set of `Files` tags apply access control to any file beginning with a `.ht`. These directives deny Web access to any `.htaccess` files (or other files which begin with `.ht`) for security reasons.

13.3.38 `CacheNegotiatedDocs`

By default, your Web server asks proxy servers not to cache any documents which were negotiated on the basis of content (that is, they may change over time or because of the input from the requester). If you uncomment `CacheNegotiatedDocs`, you are disabling that function and proxy servers will be allowed to cache the documents from then on.

13.3.39 `UseCanonicalName`

`UseCanonicalName` is set by default to `on`. `UseCanonicalName` allows the server to construct an URL that references itself, using `ServerName` and `Port`. When the server refers to itself in response to requests from clients, it uses this URL. If you set `UseCanonicalName` to `off`, the server will instead use the value that came in the request from the client to refer to itself.

13.3.40 `TypesConfig`

`TypesConfig` names the file which sets the default list of MIME type mappings (filename exten-sions to content types). The default `TypesConfig` file is `/etc/mime.types`. Instead of editing `/etc/mime.types`, the recommended way to add MIME type mappings is to use the `AddType` directive.

For more information about `AddType`, refer to Section 13.3.64, *AddType*.

13.3.41 `DefaultType`

`DefaultType` sets a default content type for the Web server to use for documents whose MIME types can not be determined. Your Web server defaults to assume a plain text content type for any file with an indeterminate content type.

13.3.42 `IfModule`

`<IfModule>` and `</IfModule>` tags surround directives that are conditional. The directives contained within the `IfModule` tags are processed under one of two conditions. The directives are processed if the module contained within the starting `<IfModule>` tag is loaded in to the Apache server. Or, if an "!" (an exclamation point) is included before the module name, the directives are processed only if the module in the starting `<IfModule>` tag is *not* compiled in.

The `mod_mime_magic.c` file is included in these `IfModule` tags. The mod_mime_magic module can be compared to the UNIX `file` command, which looks at a few bytes of a file's contents, then uses "magic numbers" and other hints in order to figure out the MIME type of the file.

If the mod_mime_magic module is compiled in to Apache, these `IfModule` tags tell the mod_mime_magic module where the hints definition file is: `/usr/share/magic` in this case.

The mod_mime_magic module is not compiled in by default. If you would like to use it, see Section 13.4, *Adding Modules to Your Server*, for instructions on how to add modules to your server.

13.3.43 `HostnameLookups`

`HostnameLookups` can be set to `on`, `off` or `double`. If you allow `HostnameLookups` (by setting it to `on`), your server will automatically resolve the IP address for each connection which requests a document from your Web server. Resolving the IP address means that your server will make one or more connections to the DNS in order to find out the hostname that corresponds to a particular IP address. If you set `HostnameLookups` to `double`, your server will perform a double-reverse DNS. In other words, after a reverse lookup is performed, a forward lookup is performed on the result. At least one of the IP addresses in the forward lookup must match the address from the first reverse lookup.

Generally, you should leave `HostnameLookups` set to `off`, because the DNS requests add a load to your server and may slow it down. If your server is busy, the effects of `HostnameLookups` will be quite noticeable.

`HostnameLookups` are also an issue for the Internet as a whole. All of the individual connections made to look up each hostname add up. Therefore, for your own Web server's benefit, as well as for the good of the Internet as a whole, you should leave `HostnameLookups` set to `off`.

If you like to see the hostnames in your log files, consider running one of the many log analyser tools that perform the DNS lookups more efficiently and in bulk when you come to rotate your log files.

13.3.44 `ErrorLog`

`ErrorLog` names the file where server errors are logged. As this directive indicates, the error log file for your Web server is `/var/log/httpd/error_log`.

The error log is a good place to look if your Web server generates any errors or fails, and you are not sure what happened.

13.3.45 `LogLevel`

`LogLevel` sets how verbose the error messages in the error logs will be. `LogLevel` can be set (from least verbose to most verbose) to `emerg`, `alert`, `crit`, `error`, `warn`, `notice`, `info` or `debug`. Your Web server's `LogLevel` is set to `warn`.

13.3.46 `LogFormat`

The `LogFormat` directives in your `httpd.conf` file set up a format for the messages in your access log. The actual `LogFormat` used will depend on the settings given in the `CustomLog` directive (see Section 13.3.47, *CustomLog*).

13.3.47 `CustomLog`

`CustomLog` identifies the log file and the log file format. In your Web server's default configuration, `CustomLog` defines the log file in which accesses to your Web server are recorded: `/var/log/httpd/access_log`. You will need to know the location of this file if you want to generate any access-based server performance statistics for your Web server.

`CustomLog` also sets the log file format to combined. The common logfile format looks like this:

```
remotehost rfc931 authuser [date] "request" status bytes referer user-agent
```

remotehost

> The remote hostname. If the hostname is not available from DNS, or if `HostnameLookups` is set to `Off`, then *remotehost* will be the IP address of the remote host.

rfc931

> Not used. You will see a – in the log file in its place.

authuser

> If authentication was required, this is the username with which the user identified him or herself. Usually, this is not used, so you will see a – in its place.

[date]

> The date and time of the request.

"request"

> The request string exactly as it came from the browser or client.

status

The HTTP status code which was returned to the browser or client.

bytes

The size of the document.

referer

This can give the URL of the Web page which linked to the the current request.

user-agent

This gives the name of the browser or client making the request.

13.3.48 `ServerSignature`

The `ServerSignature` directive adds a line containing the Apache server version and the `ServerName` of the serving host to any server-generated documents (for example, error messages sent back to clients). `ServerSignature` is set to `on` by default. You can change it to `off`, so no signature line will be added, or you can change it to `EMail`. `EMail` will add a `mailto:ServerAdmin` HTML tag to the signature line.

13.3.49 `Alias`

The `Alias` setting allows directories to be outside the `DocumentRoot` directory and yet still accessible to the Web server. Any URL ending in the alias will automatically resolve to the alias' path. By default, one alias is already set up. An `icons` directory can be accessed by the Web server, but the directory is not in the `DocumentRoot`. The `icons` directory, an alias, is actually `/var/www/icons/`, not `/var/www/html/icons/`.

13.3.50 `ScriptAlias`

The `ScriptAlias` setting defines where CGI scripts (or other types of scripts) can be found. Generally, you do not want to leave CGI scripts within the `DocumentRoot`. If CGI scripts are in `DocumentRoot`, they could potentially be viewed as text documents. Even if you do not care if people can see (and then use) your CGI scripts, revealing how they work creates opportunities for unscrupulous people to exploit any security holes in the script, and may create a security risk for your server. By default, the `cgi-bin` directory is a `ScriptAlias` of `/cgi-bin/`, and is actually located in `/var/www/cgi-bin/`.

Your `/var/www/cgi-bin` directory has `Options ExecCGI` set, meaning that execution of CGI scripts is permitted within that directory.

See Section 13.3.65, *AddHandler* and Section 13.3.29, *Directory* for instructions on how to execute CGI scripts in directories other than the `cgi-bin`.

13.3.51 `Redirect`

When a Web page is moved, `Redirect` can be used to map the old URL to a new URL. The format is as follows:

    Redirect /*path*/foo.html http://*new_domain*/*path*/foo.html

So, if an HTTP request is received for a page which used to be found at http://*your_domain*/*path*/foo.html, the server will send back the new URL (http://*new_domain*/*path*/foo.html) to the client, which should attempt to fetch the document from the new URL.

For more advanced Redirection you can use the `mod_rewrite` module included with the server.

13.3.52 `IndexOptions`

`IndexOptions` controls the appearance of server generated directing listings, by adding icons and file descriptions, etc. If `Options Indexes` is set (see Section 13.3.30, `Options`), your Web server may generate a directory listening when your Web server receives an HTTP request like the following:

http://*your_domain*/*this_directory*/

First, your Web server looks in that directory for a file from the list after the `DirectoryIndex` directive (usually, `index.html`). If your Web server does not find one of those files, it creates an HTML directory listing of the subdirectories and files in the directory. You can modify the appearance of this directory listing using certain directives in `httpd.conf`, including `IndexOptions`.

Your default configuration sets `FancyIndexing` on. If `FancyIndexing` is turned on, clicking on the column headers in the directory listing will sort the order of the display by that header. Another click on the same header will switch from ascending to descending order and back. `FancyIndexing` also shows different icons for different files, depending upon file extensions. If you use the `AddDescription` directive and turn `FancyIndexing` on, then a short description of a file will be included in the server generated directory listing.

`IndexOptions` has a number of other parameters which can be set to control the appearance of server generated directories. Parameters include `IconHeight` and `IconWidth`, to make the server include HTML `HEIGHT` and `WIDTH` tags for the icons in server generated Web pages; `IconsAre-Links`, for making the icons act as part of the HTML link anchor along with the filename, and others.

13.3.53 `AddIconByEncoding`

This directive names icons which will be displayed by files with MIME encoding, in server generated directory listings. For example, by default, your Web server shows the `compressed.gif` icon next to MIME encoded x-compress and x-gzip files in server generated directory listings.

13.3.54 `AddIconByType`

This directive names icons which will be displayed next to files with MIME types in server generated directory listings. For example, your server is set to show the icon `text.gif` next to files with a mime-type of "text," in server generated directory listings.

13.3.55 `AddIcon`

`AddIcon` tells the server which icon to show in server generated directory listings for certain file types or for files with certain extensions. For example, your Web server is set to show the icon `binary.gif` for files with `.bin` or `.exe` extensions.

13.3.56 `DefaultIcon`

`DefaultIcon` names the icon to show in server generated directory listings for files which have no other icon specified. The `unknown.gif` image file is the `DefaultIcon` for those files by default.

13.3.57 `AddDescription`

You can use `AddDescription` to show text that you specify for certain files, in server generated directory listings (you will also need to enable `FancyIndexing` as an `IndexOptions`). You can name specific files, wildcard expressions or file extensions to specify the files which this directive should apply to. For example, you could use the following line:

```
AddDescription "A file that ends in .ni" .ni
```

In server generated directory listings, all files with extensions of `.ni` would have the description `A file that ends in .ni` after the filename. Note that you will also need `FancyIndexing` turned on.

13.3.58 `ReadmeName`

`ReadmeName` names the file which (if it exists in the directory) will be appended to the end of server generated directory listings. The Web server will first try to include the file as an HTML document and then try to include it as plaintext. By default, `ReadmeName` is set to `README`.

13.3.59 `HeaderName`

`HeaderName` names the file which (if it exists in the directory) will be prepended to the start of server generated directory listings. Like `ReadmeName`, the server will try to include it as an HTML document if possible, or in plaintext if not.

13.3.60 `IndexIgnore`

`IndexIgnore` lists file extensions, partial filenames, wildcard expressions or full filenames. The Web server will not include any files which match any of those parameters in server generated directory listings.

13.3.61 `AddEncoding`

`AddEncoding` names filename extensions which should specify a particular encoding type. `AddEncoding` can also be used to instruct some browsers (not all) to uncompress certain files as they are downloaded.

13.3.62 `AddLanguage`

`AddLanguage` associates filename extensions with specific content languages. This directive is mostly useful for content negotiation, when the server returns one of several documents based on the client's language preference as set in their browser.

13.3.63 `LanguagePriority`

`LanguagePriority` allows you to set precedence for different languages in which to serve files, which will be in effect if the client expressed no preference for language in their browser.

13.3.64 `AddType`

Use the `AddType` directive to define MIME type and file extension pairs. For example, if you are using PHP4, your Web server is using the `AddType` directive to make your Web server recognize files with PHP extensions (`.php4`, `.php3` `.phtml` `.php`) as PHP MIME types.

The following `AddType` line tells your server to recognize the `.shtml` file extension (for server side includes):

```
AddType text/html .shtml
```

You will need to include the above line within the virtual host tags for any virtual hosts which should allow server side includes.

13.3.65 `AddHandler`

`AddHandler` maps file extensions to specific handlers. For example, the `cgi-script` handler can be used matched with the extension `.cgi` to automatically treat a file ending with `.cgi` as a CGI script. This will work, even for files outside of the `ScriptAlias` directory, as long as you follow the instructions provided here.

You have a CGI `AddHandler` line in your `httpd.conf` file:

```
AddHandler cgi-script .cgi
```

You will have to uncomment the line. Then Apache will execute CGI scripts for files ending in `.cgi`, even if they are outside of the `ScriptAlias`, which is set by default to locate your `/cgi-bin/` directory in `/var/www/cgi-bin/`.

You will also need to set `ExecCGI` as an `Options` for any directory containing a CGI script. See Section 13.3.29, *Directory* for more information about setting `ExecCGI` for a directory. Additionally, you will need to make sure the permissions are set correctly for the CGI scripts and the directories containing CGI scripts. CGI scripts and the entire directory path to the scripts must be set to 0755.

You will need to add the same `AddHandler` line to your `VirtualHost` setup, if you are using virtual hosts and you want them to also recognize CGI scripts outside the `ScriptAlias`.

In addition to CGI scripts, your Web server also uses `AddHandler` to process server-parsed HTML and imagemap files.

13.3.66 `Action`

`Action` allows you to specify a MIME content type and CGI script pair, so that whenever a file of that media type is requested, a particular CGI script will be executed.

13.3.67 `MetaDir`

`MetaDir` specifies the name of a directory where your Web server should look for files containing meta information (extra HTTP headers) to include when serving documents.

13.3.68 `MetaSuffix`

`MetaSuffix` specifies the filename suffix for the file that contains meta information (extra HTTP headers), which should be located in the `MetaDir` directory.

13.3.69 `ErrorDocument`

By default, in the event of a problem or error, your Web server outputs a simple (and usually cryptic) error message back to the requesting client. Instead of using the default, you can use `ErrorDocument` to configure your Web server so that it outputs a customized message or redirects the client to a local or external URL. The `ErrorDocument` directive simply associates a HTTP response code with a message or a URL which will be sent back to the client.

13.3.70 `BrowserMatch`

The `BrowserMatch` directive allows your server to define environment variables and/or take appropriate actions based on the User-Agent HTTP header field, which identifies the client's browser. By default, your Web server uses `BrowserMatch` to deny connections to specific browsers with known

problems and also to disable keepalives and HTTP header flushes for browsers that are known to have problems with those actions.

13.3.71 `Location`

`<Location>` and `</Location>` tags allow you to specify access control based on the URL.

The next use of `Location` tags is located within `IfModule mod_perl.c` tags. These configuration directives are in effect if the `mod_perl.so` DSO is loaded. See Section 13.4, *Adding Modules to Your Server* for more information about adding modules to Apache.

The `Location` tags name the `/var/www/perl` directory (an `Alias` for `/perl`) as the directory from which Perl scripts will be served. If a document is requested with an URL containing `/perl` in the path, your Web server will look in `/var/www/perl/` for the appropriate Perl script.

Several other `<Location>` options are commented out in your `httpd.conf` file. If you want to enable the functionality they provide, you will need to uncomment the appropriate section of directives.

Note

The put module is no longer distributed as part of the Apache package. You will have to load the `mod_put` package separately.

Immediately after the Perl directives discussed previously, your `httpd.conf` file includes a section of directives for enabling HTTP PUT (used by Netscape Gold's publish feature, which can post Web pages to a Web server). If you want to allow HTTP PUT, you will need to uncomment this entire section:

```
#Alias /upload /tmp
#<Location /upload>
#     EnablePut On
#     AuthType Basic
#     AuthName Temporary
#     AuthUserFile /etc/httpd/conf/passwd
#     EnableDelete Off
#     umask 007
#     <Limit PUT>
#        require valid-user
#     </Limit>
#</Location>
```

You will also need to uncomment the following lines at the beginning of `httpd.conf` so that the mod_put module is loaded in to Apache:

```
#LoadModule put_module          modules/mod_put.so
#AddModule mod_put.c
```

If you want to allow people connecting from your domain to see server status reports, you should uncomment the next section of directives:

```
#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .your_domain.com
#</Location>
```

You must replace `.your_domain.com` with your second level domain name.

If you want to provide server configuration reports (including installed modules and configuration directives) to requests from inside your domain, you will need to uncomment the following lines:

```
#<Location /server-info>
#    SetHandler server-info
#    Order deny,allow
#    Deny from all
#    Allow from .your_domain.com
#</Location>
```

Again, you must fill in `.your_domain.com`.

The next section of directives use `Location` tags to allow access to the documentation in `/usr/share/doc` (for example, with a URL like http://*your_domain*/doc/whatever.html). These directives only allow this access to requests made from the localhost.

Another use of the `Location` tags is a commented-out section which is intended to track attacks on your Web server which exploit an old bug from pre-Apache 1.1 days. If you want to track these requests, uncomment the following lines:

```
#<Location /cgi-bin/phf*>
#    Deny from all
#    ErrorDocument 403 http://phf.apache.org/phf_abuse_log.cgi
#</Location>
```

If these lines are uncommented, your Web server will redirect any requests which end in `/cgi-bin/phf*` to a logging CGI script run by the Apache Group.

13.3.72 `ProxyRequests`

If you uncomment the `IfModule` tags surrounding the `ProxyRequests` directives, your Apache server will also function as a proxy server. You will also need to load the `mod_proxy` module. For instructions on how to load in modules, see Section 13.4, *Adding Modules to Your Server*.

13.3.73 `ProxyVia`

The `ProxyVia` command controls whether or not an HTTP Via: header line is sent along with requests or replies which go through the Apache proxy server. The Via: header will show the hostname if `ProxyVia` is set to `On`, the hostname and Apache version for `Full`, any Via: lines will be passed along unchanged for `Off`, and Via: lines will be removed for `Block`.

13.3.74 Cache Directives

A number of cache directives are commented out in the proxy `IfModule` tags mentioned above. If you are using the proxy server functionality and you want to also enable the proxy cache, you should uncomment the cache directives as described. The default settings for your cache directives should be appropriate for most configurations.

`CacheRoot` sets the name of the directory which will contain cached files. The default `CacheRoot` is `/var/cache/httpd`.

`CacheSize` sets how much space the cache can use, in KB. The default `CacheSize` is 5 KB.

`CacheGcInterval` sets a number of hours. After that number of hours, files in the cache will be deleted if the cache is using more space than allowed by `CacheSize`. The default for `CacheGcInterval` is four hours.

Cached HTML documents will be retained (without a reload from the originating Web server) in the cache for a maximum number of hours set by `CacheMaxExpire`. The default is 24 hours.

The `CacheLastModifiedFactor` affects the creation of an expiry (expiration) date for a document which did not come from its originating server with its own expiry set. The default `CacheLast-ModifiedFactor` is set to `0.1`, meaning that the expiry date for such documents equals one-tenth of the amount of time since the document was last modified.

`CacheDefaultExpire` is the expiry time in hours for a document that was received using a protocol that does not support expiry times. The default is set to one hour.

Any document that is retrieved from a host and/or domain that matches one set in `NoCache` will not be cached. If you know of hosts or domains from which you do not want to cache documents, uncomment `NoCache` and set their domains or hostnames here.

13.3.75 `NameVirtualHost`

You will need to use the `NameVirtualHost` directive for the IP address (and port number if necessary) of any name-based virtual hosts you are setting up. The name-based virtual hosts configuration is used when you want to set up different virtual hosts for different domains, but you do not have (or do not want to use) different IP addresses for all of the different domain names for which your Web server serves documents.

Note

You cannot use name-based virtual hosts with your secure server. Any name-based virtual hosts you set up will only work with non-secure HTTP connections and not with SSL connections.

You cannot use name-based virtual hosts with your secure server because the SSL handshake (when the browser accepts the secure Web server's authenticating certificate) occurs before the HTTP request which identifies the correct name-based virtual host. In other words, authentication occurs before there is any identification of different name-based virtual hosts. If you want to use virtual hosts with your secure server, you will need to use IP address-based virtual hosts.

If you are using name-based virtual hosts, uncomment the `NameVirtualHost` configuration directive and add the correct IP address for your server after `NameVirtualHost`. Then add more information about the different domains using the `VirtualHost` tags which surround the `ServerName` for each virtual host, plus any other configuration directives which are only applicable to that virtual host.

13.3.76 `VirtualHost`

`<VirtualHost>` and `</VirtualHost>` tags surround any configuration directives which are intended to apply to a virtual host. Most configuration directives can be used within virtual host tags, and then they only apply to that particular virtual host.

A set of commented out `VirtualHost` tags surround some example configuration directives and placeholders for the information you would need to fill in to set up a virtual host. Please see Section 13.5, *Using Virtual Hosts*, for more information about virtual hosts.

13.3.77 `SetEnvIf`

The Apache configuration directive `SetEnvIf` can be used to set environment variables based on headers in the request. In the supplied `httpd.conf` file, it is used to disable HTTP keepalive and to allow SSL to close the connection without a close notify alert from the client browser. This setting is necessary for certain browsers that do not reliably shut down the SSL connection.

13.3.78 SSL Configuration Directives

The SSL directives in your server's `httpd.conf` file are included to enable secure Web communications using SSL and TLS.

For more information on SSL directives, please point your browser to http://*your_domain*/manual/mod/mod_ssl/. More information on SSL directives is also available at http://www.modssl.org/docs/2.7/ssl_reference.html, a chapter in a Web document about mod_ssl by Ralf Engelschall. The same document, the mod_ssl *User Manual*, begins at http://www.modssl.org/docs/2.7/ and is a great reference source for mod_ssl and for Web cryptography in general. This manual provides general information about securing your Web server in Chapter 13, *Apache*.

Note

Do not modify your SSL directives unless you are absolutely sure about what you are doing. In most cases, the SSL directives are configured appropriately as installed.

13.4 Adding Modules to Your Server

Since Apache 1.3 supports DSOs, you can easily load Apache modules or compile in your own modules to your Web server. DSO support means that modules may be loaded at runtime. Since the modules are only loaded as necessary, they will not use any memory unless they are loaded and less memory will be needed overall.

The Apache Group provides complete DSO Documentation at http://httpd.apache.org/docs/dso.html. After installation of your server, you can also check http://*your_domain*/manual/mod/ for documentation on Apache modules in HTML format (if you installed the `apache-manual` package). A short description of how to load modules is provided next. If you need more details, check the URLs provided.

For Apache to use a dynamically shared module, that module must have a `LoadModule` line and an `AddModule` line in `httpd.conf`. By default, many modules have these two lines already included in `httpd.conf`, but a few of the less commonly used modules are commented out. The commented out modules were included during compilation, but they are not loaded by default.

If you need to use one of those non-loaded modules, look in the `httpd.conf` file to see all the available modules. Each of the available modules has a corresponding `LoadModule` line. To show you an example, the `LoadModule` section begins with these seven lines:

```
#LoadModule mmap_static_module modules/mod_mmap_static.so
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule env_module         modules/mod_env.so
LoadModule config_log_module  modules/mod_log_config.so
LoadModule agent_log_module   modules/mod_log_agent.so
LoadModule referer_log_module modules/mod_log_referer.so
```

```
#LoadModule mime_magic_module  modules/mod_mime_magic.so
```

Most of the lines are not commented out, indicating that each associated module was compiled in and
is loaded in by default. The first line is commented out, which means that the corresponding module
(`mmap_static_module`) was compiled in but not loaded.

To make Apache load an unloaded module, first uncomment the corresponding `LoadModule` line.
For example, if you wanted to make Apache load in the `mime_magic_module`, uncomment this
line:

```
#LoadModule mime_magic_module modules/mod_mime_magic.so
```

Next, you need to uncomment the corresponding line from the `AddModule` section in `httpd.conf`.
To continue with our previous example, uncomment the `mod_mime_magic` line, which looks like
the following:

```
#AddModule mod_mime_magic.c
```

Once you have uncommented the `LoadModule` and `AddModule` lines for the module that you want
to load in, stop and start Apache, as covered in Section 13.2, *Starting and Stopping* `httpd`. After
starting, the module should be loaded in to Apache.

If you have your own module, you can add it to the `httpd.conf` file so that it is compiled in and
loaded as a DSO. If you want to do this, you need to install the `apache-devel` package, as covered
in Chapter 13, *Apache*. You need the `apache-devel` package because it installs the include files,
the header files and the APache eXtenSion (APXS) support tool. APXS uses the include files and the
header files to compile your module so that it will work with Apache.

> **WARNING**
>
> **If you plan to use the Apache Configuration Tool, a GUI utility
> provided with Red Hat Linux, you must not compile your own mod-
> ules to your Apache Web server or edit your Apache Web server's
> `httpd.conf` configuration file. Conversely, if you want to add mod-
> ules to Apache or edit `httpd.conf` by hand, do not use the Apache
> Configuration Tool.**
>
> **If you need more information on the Apache Configuration Tool,
> please see the *Official Red Hat Linux Customization Guide*.**

If you have written your own module or are using a third party module, you should be able to use
APXS to compile your module sources outside the Apache source tree, without needing to tweak

any compiler and/or linker flags. If you need more information on APXS, please see the Apache documentation at http://httpd.apache.org/docs/dso.html.

Once you have compiled your module using APXS, put your module into `/usr/lib/apache`. Then your module needs both a `LoadModule` line and an `AddModule` line in the `httpd.conf` file, just as described previously for Apache's own modules. After the `LoadModule` list in `httpd.conf`, add a line for the shared object file for your module like the following:

```
LoadModule foo_module  modules/mod_foo.so
```

Note that you will need to change the name of the module and the name of your shared object file as appropriate.

At the end of the `AddModule` list in `httpd.conf`, add a line for the source code file for your module like the following:

```
AddModule mod_foo.c
```

Note that you will need to change the name of the source code file as appropriate.

Once you have completed the previous steps, stop and start your Web server as outlined in Section 13.2, *Starting and Stopping `httpd`*. If you have done everything correctly and your module is correctly coded, your Web server should find your module and load it in as it starts.

13.4.1 The mod_ssl Security Module

The mod_ssl security portion of the Web server is provided as a Dynamic Shared Object (DSO). This means that if you recompile the Apache Web server you need to ensure that the EAPI extension patch from mod_ssl is applied to your server. Follow the instructions for building mod_ssl into Apache included with the mod_ssl documentation, but add the following flag:

```
--with-eapi-only
```

The complete command line should look like the following:

```
./configure [userflags] --with-eapi-only
```

Then build and install Apache.

Note

Red Hat cannot support re-compiled versions of the Apache Web server. Installation of the shipped version is supported, but if you re-compile Apache, you are on your own. Please do not re-compile Apache unless you know exactly what you are doing.

13.5 Using Virtual Hosts

> **WARNING**
>
> **If you plan to use the Apache Configuration Tool, a GUI utility provided with Red Hat Linux, you may not edit your Apache Web server's `httpd.conf` configuration file. Conversely, if you want to edit `httpd.conf` by hand, do not use the Apache Configuration Tool.**
>
> **If you need more information on the Apache Configuration Tool, please see the *Official Red Hat Linux Customization Guide*.**

You can use Apache's virtual hosts capability to run different servers for different IP addresses, different host names or different ports on the same machine. If you are interested in using virtual hosts, complete information is provided in the Apache documentation on your machine or on the Web at http://httpd.apache.org/docs/vhosts/.

> **Note**
>
> You cannot use name-based virtual hosts with your secure Web server, because the SSL handshake (when the browser accepts the secure Web server's certificate) occurs before the HTTP request which identifies the appropriate name-based virtual host. If you want to use name-based virtual hosts, they will only work with your non-secure Web server.

Virtual hosts are configured within the `httpd.conf` file, as described in Section 13.3, *Configuration Directives in `httpd.conf`*. Please review that section before you start to change the virtual hosts configuration on your machine.

13.5.1 The Secure Web Server Virtual Host

The default configuration of your Web server runs a non-secure and a secure server. Both servers use the same IP address and host name, but they listen on different ports, and the secure server is a virtual host. This configuration enables you to serve both secure and non-secure documents in the most efficient manner possible. Setting up the secure HTTP transmission is very resource intensive, so generally you will be able to serve far fewer pages per second with a secure server. You need to consider this when you decide what information to include on the secure server, and which you can use the non-secure server for.

The configuration directives for your secure server are contained within virtual host tags in the `httpd.conf` file. If you need to change something about the configuration of your secure server, you will need to change the configuration directives inside virtual host tags in the `httpd.conf` file. If you want to enable certain features (for example, server side includes) for your secure server, they will need to be enabled within the virtual host tags that define your secure server.

The non-secure Web server is configured as the "non-virtual" host in the `httpd.conf` file. In other words, the non-secure Web server's configuration options are outside of the virtual host tags in `httpd.conf`. If you want to change something about your non-secure Web server, you will need to change the configuration directives in `httpd.conf` outside of the virtual host tags.

By default, both the secure and the non-secure Web servers share the same `DocumentRoot`, a configuration directive specified in `httpd.conf`. In other words, the secure and the non-secure Web server look in the same place for the HTML files that they provide in response to requests. By default, the `DocumentRoot` is set to `/var/www/html`.

To change the `DocumentRoot` so that it is no longer shared by both the secure server and the non-secure server, change one of the `DocumentRoot` directives in `httpd.conf`. The `DocumentRoot` outside the virtual host tags defines the `DocumentRoot` for your non-secure Web server. The `DocumentRoot` within the virtual host tags that define your secure server is for your secure server.

If for some reason you want to disable the non-secure Web server on your machine, you can. Your secure server listens on port 443, the default port for secure Web communications, while your non-secure Web server listens on 80, the default port for non-secure Web communications. To stop the non-secure Web server from accepting connections, in `httpd.conf`, find the line which reads:

```
Port 80
```

Change the above line so that it reads:

```
Port 443
```

Then comment out the `Listen 80` line.

After these two steps, your Web server will be accepting connections on port 443, the default port for secure Web communications. However, your server will not accept connections on port 80, the default port for non-secure communications, so the non-secure Web server will be effectively disabled.

13.5.2 Setting Up Virtual Hosts

Most people will probably use their Web server as it is configured. Therefore, they will be using the built-in virtual hosts capability, but they will not have to do any manipulation of the virtual hosts directives in `httpd.conf`. However, if you would like to use the virtual hosts capability for some other reason, you can.

To create a virtual host, you will need to alter the virtual host lines, provided as an example, in `httpd.conf`, or create your own virtual host section. (Remember that name-based virtual hosts will not work with your secure server — you will need to use IP address-based virtual hosts if you need SSL-enabled virtual hosts. Your non-secure server, however, will support both IP address and name-based virtual hosts.)

The virtual host example lines read as follows:

```
#<VirtualHost ip.address.of.host.some_domain.com>
#    ServerAdmin webmaster@host.some_domain.com
#    DocumentRoot /www/docs/host.some_domain.com
#    ServerName host.some_domain.com
#    ErrorLog logs/host.some_domain.com-error_log
#    CustomLog logs/host.some_domain.com-access_log common
#</VirtualHost>
```

Uncomment all of the lines. Then add the correct information for your machine and/or your virtual host to each line.

In the first line, change `ip.address.of.host.some_domain.com` to your server's IP address. Change the `ServerName` to a *valid* DNS name to use for the virtual host. (In other words, do not just make something up. Ask your system administrator if you do not know how to get a valid domain name.)

You will also need to uncomment one of the `NameVirtualHost` lines in `httpd.conf`:

```
#NameVirtualHost 12.34.56.78:80
#NameVirtualHost 12.34.56.78
```

Uncomment one of the lines and change the IP address to the IP address (and port if necessary) for that virtual host.

Many other configuration directives can be placed between the virtual host tags, depending upon why you are setting up a virtual host.

If you set up a virtual host and want it to listen on a non-default port (80 is the default port for non-secure Web communications; 443 is the default port for secure Web communications), you will need to set up a virtual host for that port and add a `Listen` directive to `httpd.conf`, corresponding to that port.

To have a virtual host work specifically for that port, add the port number to the first line of the virtual host configuration. The first line should look something like the following:

```
<VirtualHost ip_address_of_your_server:12331>
```

This line would create a virtual host that listens on port 12331. Substitute the port number you want to use for `12331` in the previous example.

Underneath the `Listen` lines in `httpd.conf`, add a line like the following, which will instruct your Web server to listen on port 12331:

```
Listen 12331
```

You must restart `httpd` to start a new virtual host. See Section 13.2, *Starting and Stopping `httpd`* for instructions on how to start and stop `httpd`.

Much more complete information about creating and configuring both name-based and IP address-based virtual hosts is provided on the Web at http://httpd.apache.org/docs/vhosts/. Please check the Apache Group's virtual host documentation for more details on using virtual hosts.

13.6 Additional Resources

A lot of useful information concerning Apache is available. Please review these sources, especially the official Apache website and the mod_ssl website.

13.6.1 Useful Websites

- http://httpd.apache.org — The official website for the Apache Web server with documentation on all the directives and default modules.

- http://www.modssl.org — The official website for mod_ssl.

- http://www.apacheweek.com — Published weekly online and distributed via email to over 12,000 people world-wide, Red Hat's Apache Week is the definitive online resource for all users of Apache and Apache-based Web servers. Apache Week covers in-depth features and provides simple tutorials and HOWTO's for novice and expert users alike.

13.6.2 Related Books

- *Apache Desktop Reference* by Ralf S. Engelschall; Addison Wesley

  Written by ASF member and mod_ssl author Ralf Engelschall, the *Apache Desktop Reference* provides a concise but comprehensive reference guide to using Apache at compilation, configuration, and run time. This book is also available for you to download online at http://www.apacheref.com/.

- *Professional Apache* by Peter Wainwright; Wrox Press Ltd

  Professional Apache is one of the many books in Wrox Press Ltd's "Programmer to Programmer" series, aimed at experienced Apache users and Web server administrators who are using Apache for the first time.

- *Administering Apache* by Mark Allan Arnold; Osborne Media Group

  This book is targetted at Internet Service Providers who aim to provide more secure services, Web developers, applications programmers, right up to the corporate system administrator.

- *Apache Server Unleashed* by Richard Bowen, et al; SAMS BOOKS

  Rich Bowen and Ken Coar's book aspires to be the definitive encyclopedic source for Apache.

- *Apache Pocket Reference* by Andrew Ford, Gigi Estabrook; O'Reilly

  Apache Pocket Reference by Andrew Ford is the latest addition to the O'Reilly Pocket Reference series with sixteen other titles to its name. On its back cover, O'Reilly markets the *Apache Pocket Reference* as a companion volume to *Writing Apache Modules with Perl and C* and *Apache: The Definitive Guide, 2nd Edition*.

14 Berkeley Internet Name Domain (BIND)

Today, the Internet and almost all local networks depend upon a working and reliable **Domain Name Service (DNS)**, which is used to resolve names of systems into IP addresses and vice versa.

In order to facilitate DNS on your network, a **nameserver** is required to translate these names into the IP addresses necessary to make the connection. In addition, a nameserver can translate IP addresses back into a system's name, commonly called a **reverse lookup**.

This chapter discusses BIND, the structure of its configuration files, and how it may be locally or remotely administered.

For BIND configuration instructions using the GUI BIND Configuration Tool, please see *Official Red Hat Linux Customization Guide*. Note that, if you are using the BIND Configuration Tool, you should not manually edit your BIND configuration files, due to the fact that any manual changes will be overwritten by the BIND Configuration Tool.

14.1 Introduction to DNS and BIND

Systems using IP networks must know the IP address of a remote machine in order to connect to it. However, most users prefer to use names of machines, such as hostname or a **fully qualified domain name (FQDN)**, to specify a system when connecting to it. In addition, many programs utilize domain names in their configuration files when referring to a remote system, in order to allow IP addresses to be changed without modifying the system's name, among other reasons. The service that facilitates this is caused DNS, and it is normally implemented using centralized servers that are authoritative for some domains and refer to other DNS servers for information they do not already know.

DNS is made possible through the use of nameserver daemons that perform the IP/name translation. A client application will request information from the nameserver, usually connecting to it on the server's port 53. The nameserver will attempt to resolve the FQDN based on its resolver library, which may contain authoritative information about the host requested or cached data about that name from an earlier query. If the nameserver does not already have the answer in its resolver library, it will turn to other nameservers, called **root nameservers**, to determine which nameservers are authoritative for the FQDN in question. Then, with that information, it will query the authoritative nameservers for that name to determine the IP address. If performing a reverse lookup, the same procedure is used, except the query is made with an unknown IP address rather than a name.

14.1.1 Zones

On the Internet, the FQDN of a host can be broken down into different sections, and these sections are organized in a hierarchy much like a tree, with a main trunk, primary branches, secondary branches, and so forth. Consider the following FQDN:

Figure 14–1   Example of a fully qualified domain name

```
bill.sales.domain.com
```

When looking at how a FQDN is resolved to find the IP address that relates to a particular system, you must read the name from right to left, with each level of the hierarchy divided by dots (.). In this example, the com defines the **top level domain** for this FQDN. The domain name is a sub-domain under com, with sales as a sub-domain under domain. The name furthest left in a FQDN is the hostname, identifying a particular machine.

Except for the hostname, every section is a called a **zone**, which defines a particular namespace. A **namespace** controls the naming of the sub-domains to its left. While this example only contains two sub-domains, a FQDN must contain at least one sub-domain but may include many more, depending upon the namespace organization in use.

Zones are defined on authoritative nameservers through the use of **zone files**, which describe the namespace of that zone, the mail servers to be used for a particular domain or sub-domain, and much more. Zone files are stored on **primary nameservers** (also called **master nameservers**), which are truly authoritative and where changes are made to the files, and **secondary nameservers** (also called **slave nameservers**), which receive their zone files from the primary nameservers. Any nameserver can be a primary and secondary nameserver for different zones at the same time, and they may also be considered authoritative for multiple zones. It all depends on the nameserver's particular configuration..

14.1.2 Types of Nameservers

There are four primary nameserver configuration types:

- *master* — Stores original and authoritative zone records for a certain namespace, answering questions from other nameservers searching for answers concerning that namespace.

- *slave* — Also answers queries from other nameservers concerning namespaces for which it is considered an authority. However, slave nameservers get their namespace information from master nameservers via a **zone transfer**, where the slave sends the master a NOTIFY request for a particular zone and the master responds with the information, if the slave is authorized to receive the transfer.

- *caching-only* — Offers name to IP resolution services but is not authoritative for any zones. Answers for all resolutions are usually cached in a database stored in memory for a fixed period of

time, usually specified by the retrieved zone record, for quicker resolution for other DNS clients after the first resolution.

- *forwarding* — Forwards requests to a specific list of nameservers to be resolved. If none of the specified nameservers can perform the resolution, the process stops and the resolution fails.

A nameserver may be one or more of these types. For example, a nameserver can be a master for some zones, a slave for others, and only offer forwarding resolution.

14.1.3 BIND as a Nameserver

Red Hat Linux includes BIND, which is a very popular, powerful, open source nameserver. BIND uses the `named` daemon to provide its name resolution services. All configuration information for BIND is kept in the `/etc/named.conf` file and its zone files are in the `/var/named` directory. The structure and options for these various types of files can be found in Section 14.2, *BIND Configuration Files*.

BIND version 9 includes a utility called `rndc` to allow the administration of the running `named` daemon. More information about `rndc` can be found in Section 14.3, *Using `rndc`*.

14.2 BIND Configuration Files

The BIND nameserver `named` server uses the `/etc/named.conf` file for configuration. All zone files are placed in the `/var/named` directory.

WARNING

Do manually edit the `/etc/named.conf` file or any files in the `/var/named` directory if you are using the BIND Configuration Tool. Any manual changes to those files will be overwritten the next time the BIND Configuration Tool is used.

The `/etc/named.conf` file must be free of errors in order for `named` to start. While some erroneous options used in with some statements are not considered critical enough to stop the server, any errors in the statements themselves will prevent `named` from starting.

14.2.1 `/etc/named.conf`

The `/etc/named.conf` file is a collection of statements using nested options placed in ellipses { }. A sample `/etc/named.conf` file is organized similar to Figure 14–2, *Sample organization of `/etc/named.conf`*.

Figure 14–2   Sample organization of `/etc/named.conf`

```
<statement-1> ["<statement-1-name>"] [<statement-1-class>] {
    <option-1>;
    <option-2>;
    <option-N>;
};

<statement-2> ["<statement-2-name>"] [<statement-2-class>] {
    <option-1>;
    <option-2>;
    <option-N>;
};

<statement-N> ["<statement-N-name>"] [<statement-N-class>] {
    <option-1>;
    <option-2>;
    <option-N>;
};
```

The "`<statement-name>`" is only needed with `acl`, `include`, `server`, `view`, and `zone` statements. The `<statement-N-class>` may only be specified with the `zone` statement.

Comments may be placed in `/etc/named` in nested C-style characters `/* */` or after `//` and `#` characters.

The following statements may be used in `/etc/named.conf`:

• `acl <acl-name>` — Configures an access control list of IP addresses to be allowed or disallowed certain `named` services. Most of the time, individual IP addresses or IP network notation (such as `10.0.1.0/24`) is used identify the exact IPs.

  A few access control lists are already defined, so you do not have to configure an `acl` statement to define them:

  – `any` — Matches every IP address.

  – `localhost` — Matches any IP address in use by the local system.

  – `localnets` — Matches any IP address on any network that the local system connects to with its interfaces.

  – `none` — Matches no IP addresses.

  When utilized with other `/etc/named.conf` statements and their options, `acl` statements can be very useful in ensuring the proper use of your BIND nameserver. Consider the example in Figure 14–3, *Example of `acl` statements in use*.

Figure 14–3   Example of `acl` statements in use

```
acl black-hats {
    10.0.2.0/24;
    192.168.0.0/24;
};

acl red-hats {
    10.0.1.0/24;
};

options {
    blackhole { black-hats; };
    allow-query { red-hats; };
    allow-recursion { red-hats; };
}
```

This `named.conf` contains two access control lists (`black-hats` and `red-hats`.

- `controls` — Configures various security requirements necessary to use the `rndc` command to administer the `named` service.

    See `/etc/named.conf` in Section 14.3.1 to see how the `controls` statement should look, including various options that may only be used with it.

- `include "<file-name>"` — Includes the specified file within the current configuration file, allowing sensitive configuration data (such as `keys`) to be placed in a separate file with permissions that prevent non-privileged users from reading it.

- `key "<key-name>"` — Defines a particular key by name. Keys are used to authenticate various actions, such as secure updates or the use of the `rndc` command. Two options are used with `key`:

    - `algorithm <algorithm-name>` — The type of algorithm used, such as `dsa` or `hmac-md5`.

    - `secret "<key-value>"` — The encrypted key.

    See Figure 14–22, *Sample* `key` *statement in* `/etc/named.conf` for an example of a `key` statement.

- `logging` — Allows for the use of multiple types of logs, called **channels**. By using the `channel` option within the `logging` statement, a customized type of log, with its own file name (`file`), size limit (`size`), versioning (`version`), and level of importance (`severity`), can be constructed. Once a customized channel has been defined, a `category` option is used the categorize the channel and begin logging when `named` is restarted.

By default, `named` logs standard messages to the `syslog` daemon, which places them in `/var/log/messages` as its default. This occurs due to the fact that several standard channels are built into BIND with various severity levels, such as one that handles informational logging messages (`default_syslog`) and another that specifically handles debugging messages (`default_debug`). A default category, called `default`, uses the built-in channels to do normal logging without any special configuration.

Customizing the logging process can be a very detailed process and is beyond the scope of this chapter. For information on creating custom BIND logs, see the *BIND 9 Administrator Reference Manual*.

- `options` — Assigns values to many assorted options, including the use of forwarders, the location of the `named` working directory, the names of the various files, and much more.

  The following options are among the most commonly used:

  - `allow-query` — Specifies which hosts are allowed to query this nameserver. By default, all hosts are allowed to query. An access control list or collection of IP addresses or networks may be used here to only allow particular hosts to query the nameserver.

  - `allow-recursion` — Similar to `allow-query`, except it applies to recursive queries. By default, all hosts are allowed to perform recursive queries on the nameserver.

  - `directory` — Changes the `named` working directory to something other than the default `/var/named`.

  - `forward` — Controls how forwarding occurs, if the `forwarders` option contains valid IP addresses designating where to sent requests.

    If the `first` option is used, then the nameservers specified in the `forwarders` option are queried first for the information, and if they do not have it, `named` will attempt the resolution itself.

    If the `only` option is used, `named` will not attempt the resolution itself if the forwarders are not successful.

  - `forwarders` — Specifies a list of nameservers where requests should be forwarded for resolution.

  - `listen-on` — Specifies the network interface that `named` will use to listen for queries. By default, all interfaces are used.

    This option is useful if you have more than one network interface and would like to limit the systems that can make requests of your nameserver. For example, if you have a machine serving as a gateway and a nameserver, and you would like to block any requests except those that originate from your private network, your `listen-on` option might look like Figure 14–4, *Example of* `listen-on` *option*.

Figure 14–4   Example of `listen-on` option

```
options {
  listen-on { 10.0.1.1; };
};
```

In this way, only requests that arrive from the network interface serving the private network (10.0.1.1) will be accepted.

– `notify` — Controls whether `named` notifies the slave servers when a zone is updated. The default is `yes`, but you can set this to `no`, to prevent slaves from being notified, or `explicit`, to only notify servers in an `also-notify` list.

– `pid-file` — Allows you to specify the location of the process ID file created by `named` when it starts.

– `statistics-file` — Allows you to specify the location of where the statistics file is written. By default, `named` statistics are saved in `/var/named/named.stats`.

Dozens of other options are also available, many of which rely upon one another to work properly. See the *BIND 9 Administrator Reference Manual* for more details.

• `server` — Defines particular options that affect how `named` should act toward remote name-servers, especially regarding notifications and zone transfers.

The `transfer-format` option controls whether one resource records is send with each message (`one-answer`) or multiple resource records are sent with each message (`many-answers`). While `many-answers` is more efficient, only newer BIND nameservers understand it.

• `trusted-keys` — Contains assorted public keys used for DNSSEC. See Section 14.4.3, *Security* for an introduction to BIND security.

• `view "<view-name>"` — Creates special views that respond with a particular type of information depending upon the host contacting the nameserver. This allows some hosts to receive one answer regarding a particular zone while other hosts receive totally different information. Alternatively, certain zones may only be made available to particular trusted hosts while non-trusted hosts may still make queries for other zones.

Multiple views may be used, so long as their names are unique. The `match-clients` option specifies the IP addresses that apply to a particular view. Any `option` statements may also be used within a view, overriding the global options already configured for `named`. Most `view` statements contain multiple `zone` statements that apply to the `match-clients` list. The order in which `view` statements are listed is important, as the first `view` statement that matches a particular client's IP address is used.

See Section 14.4.2, *Multiple Views* for more information about the `view` statement.

- zone "*<zone-name>*" — Specifies particular zones for which this nameserver is authoritative. The zone statement is primarily used to specify the file containing the zone's configuration and pass certain options about that zone to named that override other global option statements used in /etc/named.conf.

  The name of the zone is important, as it is the default value assigned to the $ORIGIN directive used in the zone file and is appended to non-FQDNs. So, for example, if this zone statement defines the namespace for domain.com, you should use domain.com as the *<zone-name>* so it will be placed at the end of hostnames used in the zone file.

  The most common zone statement options include:

  - allow-query — Specifies the clients that are allowed to request information about this zone. The default is to allow all query requests.

  - allow-transfer — Specifies the slave servers that are allowed to request a transfer of the zone's information. The default is to allow all transfer requests.

  - allow-update — Specifies the hosts that are allowed to dynamically update information in their zone. The default is to deny all dynamic update requests.

  > **CAUTION**
  >
  > Be very careful about allowing hosts to update information about their zone. Do not enable this option unless the host specified is completely trusted. It is a much better idea to have an administrator manually update the zone's records and reload the named service, if possible.

  - file — Specifies the name of the file in the named working directory (by default /var/named) that contains the zone's configuration data.

  - masters — Used if the zone is defined as a slave type. The masters option tells a slave's named the IP address(es) from which to request authoritative zone information.

  - notify — Works in a similar manner to the notify option used with the option statement.

  - type — Defines the type of zone. The following types may be used:

    ⇒ forward — Tells the nameserver to forward all requests for information about this zone to other nameservers.

    ⇒ hint — A special type of zone that is used to point to the root nameservers, which are used to resolve queries when a zone is not otherwise known. You should not need to configure a hint zone beyond the default in /etc/named.conf.

⇒ `master` — Designates this nameserver as authoritative for this zone. A zone should be set as the `master` type if you have the zone's configuration files on this system.

⇒ `slave` — Designates this nameserver as a slave server for this zone, telling `named` to request the zone's configuration files from the master nameserver's IP address for that zone.

– `zone-statistics` — Tells `named` to keep statistics concerning this zone, writing them to either the default location (`/var/named/named.stats`) or the place specially designated by the `statistics-file` option in the `server` statement, if it exists.

Sample Zone Statements

Most changes to the `/etc/named.conf` file of a master or slave nameserver concerns adding, modifying, or deleting `zone` statements. While these `zone` statements can contain many options, most nameservers use few of them. The following `zone` statements are very basic examples that may be used in a master-slave nameserver relationship.

A `zone` statement on a primary nameserver hosting the domain `domain.com` may look like Figure 14–5, *Example of a simple master `zone` statement*.

Figure 14–5   Example of a simple master `zone` statement

```
zone "domain.com" IN {
  type master;
  file "domain.com.zone";
  allow-update { none; };
};
```

This `zone` statement names the zone `domain.com`, sets the `type` as `master`, tells `named` to read the `/var/named/domain.com.zone` file to configure the zone, and to allow no updates by any other hosts.

A slave server's `zone` statement for `domain.com` might look like Figure 14–6, *Example of a simple slave `zone` statement*.

Figure 14–6   Example of a simple slave `zone` statement

```
zone "domain.com" {
  type slave;
  file "domain.com.zone";
  masters { 192.168.0.1; };
};
```

This `zone` statement tells `named` on the slave server to look to the `192.168.0.1` master server to find out the configuration information for the zone called `domain.com`. The information the slave server receives from the master server is saved in the `/var/named/domain.com.zone` file.

14.2.2 Zone Files

Zone files, which contain information about a particular namespace, are stored in the `named` working directory. By default, this is `/var/named`. Each zone file is named according to the `file` option data in the `zone` statement, usually in a way that relates to the domain in question and identifies the file as containing zone data, such as `example.com.zone`.

Each zone file may contain directives and resource records. **Directives** tell the nameserver to do a certain thing or apply a special setting to the zone. **Resource records** define the parameters of the zone, assigning an identity within the zone's namespace to particular systems. Directives are optional, but resource records are required to provide nameservice to that zone. All directives and resource records should go on their own lines.

Comments can be placed after semicolon characters (`;`) in zone files.

Zone File Directives

Directives are identified by the leading `$` character before the name of the directive and usually placed at the top of the zone file.

The following directives are the most commonly used:

- `$INCLUDE` — Tells `named` to include another zone file in this zone file at the place where the directive is used. This allows additional zone settings to be stored apart from the main zone file.

- `$ORIGIN` — Sets the domain name to be appended to any unqualified records, such as those that only specify the host and nothing more.

  For example, a zone file may contains the following line:

  ```
  $ORIGIN domain.com
  ```

  At this point, any names that are used in resource records and do not end in a trailing dot (`.`) will have this domain name added to them. So, in other words, when the zone record is read by the nameserver, the first line below will be interpreted as the second line:

  ```
  ftp               IN   CNAME    server1
  ftp.domain.com.   IN   CNAME    server1.domain.com.
  ```

  #### Note

  The use of the `$ORIGIN` directive is unnecessary if you name the zone in `/etc/named.conf` the same as the value you would assign to `$ORIGIN`. The zone's name is used as the `$ORIGIN` directive's value by default.

- $TTL — Sets the default **Time to Live (TTL)** value for the zone. This is the number, in seconds, given to nameservers that tells how long the zone's resource records should continue to be valid. A resource record can contains its own TTL value, which would override this directive.

  Increasing this value tells remote nameservers to cache this zone's information for a longer time. This reduces the number of queries made concerning this zone, but it also lengthens the amount of time required to proliferate resource record changes.

Zone File Resource Records

Zone file resource records contain columns of data, separated by whitespace, that define the record. All zone file resource records are assigned a particular type, which designates the record's purpose. The following types of resource records are the most commonly used:

- A — Address record, which specifies an IP address to assign to a name.

Figure 14–7   Sample A record configuration

```
<host>      IN      A      <IP-address>
```

If the `<host>` value is omitted, then an A record points to a default IP address for the top of the namespace. This system will be the target of all non-FQDN requests.

Consider the following A record examples for the domain.com zone file:

Figure 14–8   Example A records

```
            IN      A       10.0.1.3
server1     IN      A       10.0.1.5
```

Requests for domain.com are pointed to 10.0.1.3, while requests for server1.domain.com are pointed to 10.0.1.5.

- CNAME — Canonical name record, which tells the nameserver that one name is also known as another.

Figure 14–9   Sample CNAME record configuration

```
<alias-name>      IN      CNAME      <real-name>
```

In Figure 14–9, *Sample CNAME record configuration*, any requests sent to the `<alias-name>` will point to the host named `<real-name>`. CNAME records are most commonly used to point services that use a common naming scheme to the correct host.

Consider the example in Figure 14–10, *Example CNAME record*, where an A record sets a particular hostname to an IP address and a CNAME record points the commonly used www hostname to it.

Figure 14–10   Example CNAME record

```
server1     IN      A       10.0.1.5
```

```
www          IN    CNAME   server1
```

- MX — Mail eXchange record, which tells where mail sent to a particular namespace controlled by this zone should go.

Figure 14–11   Sample MX record configuration

```
IN    MX      <preference-value>  <email-server-name>
```

In Figure 14–11, *Sample MX record configuration*, the `<preference-value>` allows you to numerically rank the email servers you would prefer to receive email for this namespace, giving preference to some email systems over others. The MX resource record with the lowest `<preference-value>` is preferred over the others, but you can set multiple email servers with the same value to distribute email traffic between them.

The `<email-server-name>` may be a hostname or FQDN, as long as it points to the correct system.

Figure 14–12   Example MX records

```
IN    MX    10      mail.domain.com.
IN    MX    20      mail2.domain.com.
```

In this example, the first `mail.domain.com` email server is preferred to the `mail2.domain.com` email server when receiving email destined for the `domain.com` domain.

- NS — NameServer record, which announces the authoritative nameservers for a particular zone.

Figure 14–13   Sample NS record configuration

```
IN    NS      <nameserver-name>
```

The `<nameserver-name>` should be a FQDN.

In Figure 14–14, *Example NS records*, two nameservers are listed as authoritative for a domain. It is not important whether these nameservers are slaves or if one is a master; they are both still considered authoritative.

Figure 14–14   Example NS records

```
IN    NS    dns1.domain.com.
IN    NS    dns2.domain.com.
```

- PTR — PoinTeR record, designed to point to another part of the namespace.

PTR records are primarily used for reverse name resolution, as they point IP addresses back to a particular name. See *Reverse Name Resolution Zone Files* in Section 14.2.2 for more examples of PTR records in use.

- SOA — Start Of Authority record, proclaiming important authoritative information about the namespace to the nameserver.

Located after the directives, an SOA record is the first resource record in a zone file.

Figure 14–15   Sample SOA record configuration

```
@    IN    SOA    <primary-name-server>
<hostmaster-email> (
                   <serial-number>
                   <time-to-refresh>
       <time-to-retry>
       <time-to-expire>
       <minimum-TTL> )
```

The @ symbol places the $ORIGIN directive (or the zone's name, if the $ORIGIN directive is not set) as the namespace being defined by this SOA resource record. The primary nameserver that is authoritative for this domain is used for the *<primary-name-server>*, and the email of the person to contact about this namespace is substituted for the *<hostmaster-email>*.

The *<serial-number>* is incremented every time you change the zone file so that named will know that it should reload this zone. The *<time-to-refresh>* tells any slave servers how long to wait before asking the master nameserver if any changes have been made to the zone. The *<serial-number>* value is used by the slave to determine if it is using outdated zone data and should refresh it.

The *<time-to-retry>* tells the slave nameserver the interval to wait before issuing another refresh request, if the master nameserver is not answering. If the master has not replied to a refresh request before the *<time-to-expire>* elapses, the slave stops responding as an authority for requests concerning that namespace.

The *<minimum-TTL>* requests that other nameservers cache the zone's information for at least this amount of time.

With BIND, all times refer to seconds. However, you can also use abbreviations for other units of time other than seconds, such as minutes (M), hours (H), days (D), and weeks (W). The table in Table 14–1, *Seconds compared to other time units* shows an amount of time in seconds and the equivalent time in another format.

Table 14–1   Seconds compared to other time units

Seconds	Other Time Units
60	1M
1800	30M

Seconds	Other Time Units
3600	1H
10800	3H
21600	6H
43200	12H
86400	1D
259200	3D
604800	1W

The following example demonstrates how a basic SOA resource record might look.

Figure 14–16   Example SOA records

```
@    IN    SOA    dns1.domain.com.     hostmaster.domain.com. (
                  2001062501 ; serial
                  21600      ; refresh after 6 hours
                  3600       ; retry after 1 hour
                  604800     ; expire after 1 week
                  86400 )    ; minimum TTL of 1 day
```

Zone File Examples

Seen individually, the directives and resource records can be difficult to grasp. However, everything makes much more sense when it is placed together in a common file.

In Figure 14–17, *An example of a basic zone file*, a very basic zone file is shown.

Figure 14–17   An example of a basic zone file

```
$ORIGIN domain.com
$TTL 86400
@    IN    SOA    dns1.domain.com.     hostmaster.domain.com. (
                  2001062501 ; serial
                  21600      ; refresh after 6 hours
                  3600       ; retry after 1 hour
                  604800     ; expire after 1 week
                  86400 )    ; minimum TTL of 1 day

     IN    NS     dns1.domain.com.
     IN    NS     dns2.domain.com.

     IN    MX     10    mail.domain.com.
```

```
          IN     MX     20      mail2.domain.com.

                 IN     A       10.0.1.5

   server1    IN     A       10.0.1.5
   server2    IN     A       10.0.1.7
   dns1       IN     A       10.0.1.2
   dns2       IN     A       10.0.1.3

   ftp        IN     CNAME   server1
   mail       IN     CNAME   server1
   mail2      IN     CNAME   server2
   www        IN     CNAME   server2
```

In this example, standard directives and `SOA` values are used. The authoritative nameservers are set to be `dns1.domain.com` and `dns2.domain.com`, which have `A` records that tie them to `10.0.1.2` and `10.0.1.3`, respectively.

The email servers configured with the `MX` records point to `server1` and `server2` via CNAME records. Since the `server1` and `server2` names do not end in a trailing dot (`.`), the `$ORIGIN` domain is placed after them, expanding them to `server1.domain.com` and `server2.domain.com`. Through the related `A` resource records, their IP addresses can be determined.

The popular FTP and Web services, available at the standard `ftp.domain.com` and `www.domain.com` names, are pointed toward machines providing the appropriate services for those names using `CNAME` records.

Reverse Name Resolution Zone Files

A reverse name resolution zone file is used to translate an IP address in a particular namespace into a FQDN. It looks very similar to a standard zone file, except that `PTR` resource records are used to link the IP addresses to a certain system's name.

A `PTR` record is written in a manner similar to Figure 14–18, *Sample `PTR` record configuration*.

Figure 14–18   Sample `PTR` record configuration

```
   <last-IP-digit>     IN     PTR    <FQDN-of-system>
```

The `<last-IP-digit>` relates to the last number in an IP address that should point to a particular system's FQDN.

In Figure 14–19, *An example of a basic reverse zone resolution file*, IP addresses `10.0.1.20` through `10.0.1.25` are pointed to corresponding FQDNs.

Figure 14–19   An example of a basic reverse zone resolution file

```
   $ORIGIN 1.0.10.in-addr.arpa
```

```
$TTL 86400
@     IN      SOA     dns1.domain.com.      hostmaster.domain.com. (
                      2001062501 ; serial
                      21600      ; refresh after 6 hours
                      3600       ; retry after 1 hour
                      604800     ; expire after 1 week
                      86400 )    ; minimum TTL of 1 day

      IN      NS      dns1.domain.com.
      IN      NS      dns2.domain.com.

20    IN      PTR     alice.domain.com.
21    IN      PTR     betty.domain.com.
22    IN      PTR     charlie.domain.com.
23    IN      PTR     doug.domain.com.
24    IN      PTR     ernest.domain.com.
25    IN      PTR     fanny.domain.com.
```

This zone file would be called into service with a `zone` statement in the `/etc/named.conf` file that looks similar to Figure 14–20, *An example of a reverse resolution `zone` statement*.

Figure 14–20   An example of a reverse resolution `zone` statement

```
zone "1.0.10.in-addr.arpa" IN {
  type master;
  file "domain.com.rr.zone";
  allow-update { none; };
};
```

There is very little difference between this example an a standard `zone` statement, except for how the zone is named. Note that a reverse name resolution zone requires the first three blocks of the IP address to be reversed and `".in-addr.arpa"` to be included after them. This allows the single block of IP numbers used in the reverse name resolution zone file to be correctly attached with this zone.

14.3 Using `rndc`

BIND includes a utility called `rndc` that allows you to administer the `named` daemon, locally or remotely, with command line statements. The `rndc` program uses the `/etc/rndc.conf` file for its configuration options, which can be overridden with command line options.

In order to prevent unauthorized users on other systems from controlling BIND on your server, a shared secret key method is used to explicitly grant privileges to particular hosts. In order for `rndc` to issue commands to any `named`, even on a local machine, the keys used in `/etc/named.conf` and `/etc/rndc.conf` must match.

14.3.1 Configuration Files

Before attempting to use the `rndc` command, verify that the proper configuration lines are in place in the necessary files. Most likely, your configuration files are not properly set if you run `rndc` and see a message that states:

```
rndc: connect: connection refused
```

/etc/named.conf

In order for `rndc` to be allowed to connect to your `named` service, you must have a `controls` statement in your `/etc/named.conf` file when `named` starts. The sample `controls` statement shown in Figure 14–21, *Sample `controls` statement in `/etc/named.conf`* will allow you to execute `rndc` commands locally.

Figure 14–21   Sample `controls` statement in `/etc/named.conf`

```
controls {
  inet 127.0.0.1 allow { localhost; } keys { <key-name>; };
};
```

This statement tells `named` to listen on the default TCP port 953 of the loopback address and allow `rndc` commands coming from the localhost, if the proper key is given. The `<key-name>` relates to the `key` statement, which is also in the `/etc/named.conf` file. A sample `key` statement is shown in Figure 14–22, *Sample `key` statement in `/etc/named.conf`*.

Figure 14–22   Sample `key` statement in `/etc/named.conf`

```
key "<key-name>" {
  algorithm hmac-md5;
  secret "<key-value>";
};
```

In this case, the `<key-value>` is a HMAC-MD5 key. You can generate your own HMAC-MD5 keys with the following command:

```
dnssec-keygen -a hmac-md5 -b <bit-length> -n HOST <key-file-name>
```

A key with at least a 256-bit length is good idea. The actual key that should be placed in the `<key-value>` area can found in the `<key-file-name>`.

The name of the key used in `/etc/named.conf` should be something other than `key`.

/etc/rndc.conf

To configure `rndc` to automatically use the key specified in `/etc/named.conf` for the localhost, three statements are needed. The `options` statement allows you to set the default server and key for `rndc` to use, as seen in Figure 14–23, *Sample `options` statement in `/etc/rndc.conf`*.

Figure 14–23   Sample `options` statement in `/etc/rndc.conf`

```
options {
  default-server  localhost;
  default-key     "<key-name>";
};
```

Optionally, the rndc command can be told to use a default key when accessing a particular server, as seen in Figure 14–24, *Sample `server` statement in `/etc/rndc.conf`*.

Figure 14–24   Sample `server` statement in `/etc/rndc.conf`

```
server localhost {
  key  "<key-name>";
};
```

However, this server statement is only really helpful if you connect to multiple servers with rndc.

The key is the most important statement in /etc/rndc.conf.

Figure 14–25   Sample `key` statement in `/etc/rndc.conf`

```
key "<key-name>" {
  algorithm hmac-md5;
  secret "<key-value>";
};
```

The *<key-name>* and *<key-value>* should be exactly the same as their settings in /etc/named.conf.

To test all of the settings, try the rndc reload command. You should see response similar to this:

```
rndc: reload command successful
```

If the command was not successful, carefully go over the /etc/named.conf and /etc/rndc.conf files and look for errors.

```
CAUTION
```

You should make sure that unprivileged users cannot read or write the /etc/rndc.conf file.

14.3.2 Command Line Options

An rndc command takes the following form:

Figure 14–26   Structure of a `rndc` command

```
rndc <options> <command> <command-options>
```

The `<options>` area is not required, and you do not have to use `<command-options>` unless the command requires them.

When executing `rndc` on a properly configured localhost, the following commands are available:

- `halt` — Stops the `named` service immediately.
- `querylog` — Turns on logging of all queries made by clients to this nameserver.
- `refresh` — Refreshes the nameserver's database.
- `reload` — Tells the nameserver to reload the zone files but keep all other previously cached responses. This allows you to make changes to zone files and have them take effect on your master and slave servers without losing all stored name resolutions.

  If your changes only affected a particular zone, you can tell `named` to only reload that one zone. Type the name of the zone after the `reload` command.
- `stats` — Dumps the current `named` stats to the `/var/named/named.stats` file.
- `stop` — Stops the server gracefully, saving any dynamic update and IXFR data before exiting.

Occasionally, you may want to override the default settings in the `/etc/rndc.conf` file. The following options are available:

- `-c <configuration-file>` — Tells `rndc` to use a configuration file other than the default `/etc/rndc.conf`.
- `-p <port-number>` — Specifies a different port number to use for the `rndc` connection other than the default 953.
- `-s <server>` — Tells `rndc` to send the command to a server other than the `default-server` option in `/etc/rndc.conf` file.

  In order for this to work, you must have configured the other `named` service to accept commands from your host and have the key for that nameservice.
- `-y <key-name>` — Allows you to specify a key other than the `default-key` option in `/etc/rndc.conf` file.

Additional information about these options can be found in the `rndc` man page.

14.4 BIND Advanced Features

Most BIND implementations only use `named` to provide name resolution services or to act as an authority for a particular domain or sub-domain. However, BIND version 9 has a number of advanced

features that, when properly configured and utilized, allow for a more secure and efficient DNS service.

CAUTION

Some of these advanced features, such as DNSSEC, TSIG, and IXFR, should only be used in network environments with nameservers that support the features. If your network environment includes non-BIND or older BIND nameservers, check to see if a particular advanced feature is available before attempting to use it.

Do not assume another type of nameserver supports all of these features, as many do not.

All of the features discussed here are discussed in greater detail in the *BIND 9 Administrator Reference Manual*. See Section 14.6, *Additional Resources* for places to find this manual.

14.4.1 DNS Protocol Enhancements

BIND supports **Incremental Zone Transfers (IXFR)**, where slave nameserver will only download the updated portions of a zone modified on a master nameserver. The standard transfer AXFR process requires that the entire zone be transferred to each slave nameserver for even the smallest change. For very popular domains with very lengthy zone files and many slave nameservers, IXFR makes the notification and update process much less resource intensive.

Note that IXFR is only available if you are also using **dynamic updating** to make changes to master zone records. If you are manually editing zone files to make changes, AXFR will be used. More information on dynamic updating is available in the *BIND 9 Administrator Reference Manual*.

14.4.2 Multiple Views

Through the use of the `view` statement in `/etc/named.conf`, BIND allows you to configure a nameserver to answer queries for some clients in a different way than it answers them for others.

This is primarily useful if you would like clients external to your network to not be able to perform a particular DNS service or see a particular type of information, while at the same time allowing internal clients to be able to do them.

The `view` statement uses the `match-clients` option to match IP addresses or entire networks and give them special options and zone data.

14.4.3 Security

BIND supports a number of different methods to protect the updating and transfer of zones, on both master and slave nameservers:

* *DNSSEC* — Short for **DNS SECurity**, this feature allows for zones to be cryptographically signed with a **zone key**.

  In this way, the information about a specific zone can be verified as coming from a nameserver that has signed it with a particular private key, as long as the recipient has that nameserver's public key.

  BIND version 9 also supports the SIG(0) public/private key method of message authentication.

* *TSIG* — Short for **Transaction SIGnatures**, a shared secret key exists on the master and slave server, verifying that a transfer from master to slave is authorized.

  This feature strengthens the standard IP address-based method of transfer authorization. An attacker would not only need to have access to the IP address to transfer the zone, but they would also need to know the secret key.

  BIND version 9 also support **TKEY**, which is another shared secret key method of authorizing zone transfers.

14.4.4 IP version 6

BIND version 9 can provide nameservice in IP version 6 (IPv6) environments, through the use of `A6` zone records.

If your network environment includes both IPv4 and IPv6 hosts, you should use the `lwresd` light-weight resolver daemon on your network clients. This daemon is essentially a very efficient, caching-only nameserver, which understands the new `A6` and `DNAME` records used with IPv6. See the `lwresd` man page for more information.

14.5 Common Mistakes to Avoid

It is very common for beginners to make mistakes when editing BIND configuration files or run in to initial difficulties using `named`. Be sure to avoid the following problems:

* *Make sure you increment the serial number when editing a zone file.*

  If the serial number is not incremented, your master nameserver may have the correct, new information, but your slave nameservers will never be notified of the change or attempt to refresh their data of that zone. After all, their serial number will match the one on the master nameserver, even though their data for the zone is completely different than what is on the master.

* *Be careful to use ellipses and semi-colons correctly in the* `/etc/named.conf` *file.*

An omitted semi-colon or unclosed ellipse section will cause `named` to refuse to start.

• *Remember to place dots (.) in zone files after all FQDNs and omit them on hostnames.*

The dot symbolizes that the name given is full and complete. If the dot is omitted, then `named` will place the name of the zone or the `$ORIGIN` value after the name to complete it.

• *If you are having problems with your firewall blocking connections from your `named` to other nameservers, you should manually tell `named` to use port 53 for upstream queries.*

BIND version 9 uses random ports above 1024 query other nameservers to resolve names, just like other DNS clients do, connecting to the remote nameserver's port 53. Some firewalls, however, expect nameservers to communicate with each other with both systems using port 53. You can place the following line in the `options` statement to force `named` to send queries from port 53:

```
query-source address * port 53;
```

14.6 Additional Resources

The following sources of information will provide additional background concerning the utilization of BIND.

14.6.1 Installed Documentation

• BIND features a full-range of installed documentation covering many different topics, each placed in its own subject directory:

   – `/usr/share/doc/bind-<version-number>` — Contains a README file with a list of the most recent features.

   – `/usr/share/doc/bind-<version-number>/arm` — Contains HTML and SGML of the *BIND 9 Administrator Reference Manual*, which details BIND resource requirements, how to configure different types of nameservers, perform load balancing, and other advanced topics. For most new users of BIND, this is the best place to start.

   – `/usr/share/doc/bind-<version-number>/draft` — Contains assorted technical documents that look at issues related to DNS service and some methods proposed to address them.

   – `/usr/share/doc/bind-<version-number>/misc` — Contains documents designed to address specific advanced issues. Users of BIND version 8 should consult the `migration` document for specific changes they must make when moving to BIND 9. The `options` file lists all of the options implemented in BIND 9 that are used in `/etc/named.conf`.

   – `/usr/share/doc/bind-<version-number>/rfc` — Every RFC document related to BIND is conveniently placed in this directory.

- The following man pages are useful:
  - `named` — Explores assorted arguments that can be used to control the BIND nameserver daemon, such as the use of an alternative configuration file and running on a different port number or as a different user.
  - `rndc` — Explains the different options available when using the `rndc` command to control a BIND nameserver.

14.6.2 Useful Websites

- http://www.isc.org/products/BIND — The home page of the BIND project, where you can find information concerning current releases and download a PDF version of the *BIND 9 Administrator Reference Manual*.

- http://www.redhat.com/mirrors/LDP/HOWTO/DNS-HOWTO.html — Covers the use of BIND as a resolving, caching nameserver or the configuration of various zone files necessary to serve as the primary nameserver for a domain.

14.6.3 Related Books

- *DNS and BIND* by Paul Albitz and Cricket Liu; O'Reilly & Associates — A popular reference that explains both common and esoteric BIND configuration options, as well as providing strategies to secure your DNS server.

- *The Concise Guide to DNS and BIND* by Nicolai Langfeldt; Que — Looks at the connection between multiple network services and BIND, with an emphasis on task-oriented, technical topics.

15 Lightweight Directory Access Protocol (LDAP)

15.1 What is LDAP?

LDAP (**Lightweight Directory Access Protocol**) is a proposed open standard for accessing global or local directory services over a network and/or the Internet. A directory, in this sense, is very much like a phone book. LDAP can handle other information, but at present it is typically used to associate names with phone numbers and email addresses. LDAP directories are designed to support a high volume of queries, but the data stored in the directory doesn't change very often.

LDAP is much more useful than a paper phone book, because LDAP's design is intended to support propagation over LDAP servers throughout the Internet, much like the **Domain Name Service** (**DNS**). DNS servers help to connect computers to one another based on fully qualified domain names or the type of service requested from a domain, such as mail exchange. Without DNS servers, hostnames could not be translated into IP addresses, which are required for TCP/IP communication. In the future, LDAP could provide the same type of global access to many types of directory information. Currently, LDAP is more commonly used within a single large organization, like a college or a company, for directory services.

LDAP is a client-server system. An LDAP client connects to an LDAP server and either queries it for information or provides information that needs to be entered into the directory. The server either answers the query, refers the query to another LDAP server, or accepts the information for incorporation into the directory, based on the permission of the user.

LDAP is sometimes known as **X.500 Lite**. X.500 is an international standard for directories and full-featured, but it is also complex, requiring a lot of computing resources and the full OSI stack. LDAP, in contrast, can run easily on a PC and over TCP/IP. LDAP can access X.500 directories but does not support every capability of X.500.

This chapter will refer to the configuration and use of OpenLDAP, an open source implementation of LDAP. OpenLDAP includes slapd (a stand-alone LDAP server), slurpd (a stand-alone LDAP replication server), libraries implementing the LDAP protocol, utilities, tools, and sample clients.

15.2 Pros and Cons of LDAP

The main benefit of using LDAP is the consolidation of certain types of information within your organization. For example, all of the different lists of users within your organization can be merged into one LDAP directory. This directory can be queried by any LDAP-enabled applications that need this information. It can also be used by users who need directory information.

Other LDAP benefits include its ease of implementation (compared to X.500) and its well-defined Application Programming Interface (API), which means that the number of LDAP-enabled applications and LDAP gateways should increase in the future.

On the negative side, if you want to use LDAP, you will need LDAP-enabled applications or the ability to use LDAP gateways. While LDAP usage should only increase, currently there are not very many LDAP-enabled applications available for Linux. Also, while LDAP does support some access control, it does not possess as many security features as X.500.

15.3 Uses for LDAP

Several Netscape applications, including web browsers using the Netscape Roaming Access feature, are LDAP-enabled. Sendmail can use LDAP to look up addresses. Your organization can use LDAP as an organization-wide directory and/or name service (in place of NIS or flat files). You can even use a personal LDAP server to keep track of your own email address book (see Section 15.11, *Additional Resources*).

Since LDAP is an open and configurable protocol, it can be used to store almost any type of information relating to a particular organizational structure.

15.3.1 LDAP Applications

Several LDAP client applications are available that greatly simplify viewing and changing LDAP information:

- LDAP Browser/Editor — A user-friendly tool written in 100% Java for easy deployment across different platforms, available at http://www.iit.edu/~gawojar/ldap

- GQ — A GTK-based LDAP client, available with the Red Hat Linux 7.2 distribution or at http://biot.com/gq

- kldap — An LDAP client for the KDE Project, available at http://www.mountpoint.ch/oliver/kldap

15.3.2 LDAP and PAM

LDAP can be used as an authentication service via the `pam_ldap` module. LDAP is commonly used as a central authentication server so that users have a unified login that covers console logins, POP servers, IMAP servers, machines connected to the network using Samba, and even Windows NT/2000 machines. Using LDAP, all of these login situations can rely on the same user ID and password combination, greatly simplifying administration. The `pam_ldap` module is provided in the `nss_ldap` package.

15.4 LDAP Terminology

An **entry** is one unit in an LDAP directory. An entry is identified or referenced by its unique **Distinguished Name** (**DN**).

An entry has **attributes**, which are pieces of information directly associated with the entry. For example, an organization could be an LDAP entry. Attributes associated with the organization might be its fax number, its address, and so on. People can also be entries in the LDAP directory. Common attributes for people include their telephone numbers and their e-mail addresses.

Certain attributes are required, while other attributes are optional. An **objectclass** sets which attributes are required and which are optional. Objectclass definitions are found in various schema files, located in the `/etc/openldap/schema` directory.

The **LDAP Data Interchange Format** (LDIF) is an ASCII text format for LDAP entries. Files that import or export data to and from LDAP servers must be in LDIF format. An LDIF entry looks like this:

```
[<id>]
dn: <distinguished name>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
```

An entry can contain as many `<attrtype>`: `<attrvalue>` pairs as needed. A blank line indicates that the entry is finished and that another entry is about to begin.

CAUTION

> Your `<attrtype>` and `<attrvalue>` pairs *must* be defined in a schema before they can be used. You cannot simply define them in an LDIF file and expect an LDAP server without corresponding data in its schema files to be able to use this information.

Everything enclosed within `<  >` is variable and can be set by you when you add an LDAP entry, with the exception of the `<id>`. The `<id>` is a number normally set by the LDAP tools when you add an entry, and you will probably never need to manually set one.

15.5 OpenLDAP 2.0 Enhancements

OpenLDAP 2.0 represents a major upgrade for the application, bringing with it:

- *LDAPv3 Support* — Now works with SASL, TLS, and SSL, among other improvements, in full compliance with RFC 2251-2256; many of the changes since LDAPv2 are aimed to help make LDAP a much more secure protocol.

- *IPv6 Support* — Now supports the next generation Internet Protocol.

- *LDAP Over IPC* — OpenLDAP can communicate within a particular system without having to go over a network, making it more secure.

- *Updated C API* — Improves the way programmers can connect to and use the application.

- *LDIFv1 Support* — Full compliance with the LDAP Data Interchange Format (LDIF) version 1.

- *Enhanced Stand-Alone LDAP Server* — Includes an updated access control system, thread pooling, better tools and much more.

15.6 OpenLDAP Files

OpenLDAP configuration files are installed into the /etc/openldap directory. If you do an ls on /etc/openldap, you will see something like:

```
ldap.conf         ldapsearchprefs.conf  schema
ldapfilter.conf   ldaptemplates.conf    slapd.conf
```

15.6.1 Edit **/etc/openldap/slapd.conf**

The slapd.conf file, located in /etc/openldap, contains the configuration information needed by your slapd LDAP server. You will need to edit this file to make it specific to your domain and server.

The suffix line names the domain for which the LDAP server will provide information. The suffix line should be changed from:

```
suffix          "dc=your-domain, dc=com"
```

so that it reflects your domain name. For example:

```
suffix          "dc=acmewidgets, dc=com"
```

or

```
suffix          "dc=acmeuniversity, dc=edu"
```

The rootdn entry is the DN for a user who is unrestricted by the access control or administrative limit parameters set for operations on the LDAP directory. The rootdn user can be thought of as the root user for the LDAP directory. The rootdn line needs to be changed from:

```
rootdn          "cn=root, dc=your-domain, dc=com"
```

to something like:

```
rootdn          "cn=root, dc=redhat, dc=com"
```

or

```
rootdn          "cn=ldapmanager, dc=my_organization, dc=org"
```

Change the rootpw line from:

```
rootpw          secret
```

to something like

```
rootpw          {crypt}s4L9sOIJo4kBM
```

In the above example, you are using an encrypted root password, which is a much better idea than leaving a plain text root password in the slapd.conf file. To make this crypt string, you can use Perl:

```
perl -e "print crypt('passwd','a_salt_string');"
```

In the previous Perl line, *salt_string* is a two character salt, and *passwd* is the plain text version of the password.

You could also copy a passwd entry out of /etc/passwd, but this won't work if the passwd entry is an MD5 password (the default in Red Hat Linux 7.2).

15.6.2 The `schema` Directory

New to OpenLDAP version 2, the schema directory holds the various LDAP definitions, previously located in the slapd.at.conf and slapd.oc.conf files. All **attribute syntax definitions** and **objectclass definitions** are now located in the different schema files. The various schema files are referenced in /etc/openldap/slapd.conf using include lines, as shown in this example:

```
include  /etc/openldap/schema/core.schema
include  /etc/openldap/schema/cosine.schema
include  /etc/openldap/schema/inetorgperson.schema
include  /etc/openldap/schema/nis.schema
include  /etc/openldap/schema/rfc822-MailMember.schema
include  /etc/openldap/schema/autofs.schema
include  /etc/openldap/schema/kerberosobject.schema
```

```
CAUTION
```

You should not modify any of the schema items defined in the schema files installed by OpenLDAP.

You can extend the schema used by OpenLDAP to support additional attribute types and object classes using the default schema files as a guide. To do this, create a `local.schema` file in the `/etc/openldap/schema` directory. Reference this new schema within `slapd.conf` by adding the following line below your default `include` schema lines:

```
include   /etc/openldap/schema/local.schema
```

Next, go about defining your new attribute types and object classes within the `local.schema` file. Many organizations use existing attribute types and object classes from the schema files installed by default and modify them for use in the `local.schema` file. This can help you to learn the schema syntax while meeting the immediate needs of your organization.

Extending schemas to match certain specialized requirements is quite involved and beyond the scope of this chapter. Visit http://www.openldap.org/doc/admin/schema.html for information on writing new schema files.

15.7 OpenLDAP Daemons and Utilities

The OpenLDAP package includes two daemons: `slapd` and `slurpd`.

The `slapd` daemon is the stand-alone LDAP daemon, which you'll need to run to support LDAP.

The `slurpd` daemon controls the replication of LDAP directories over a network by sending changes from the master LDAP directory to slave LDAP directories. You won't need to run `slurpd` unless you have more than one LDAP server on your network. If you have two or more LDAP servers, `slurpd` will keep the various LDAP directories in sync.

OpenLDAP also includes some utilities in `/usr/bin` for adding, modifying and deleting entries in an LDAP directory:

*   `ldapmodify` — Modify entries in an LDAP database, accepting input via a file or standard input.

*   `ldapadd` — Adds entries to your directory, accepting input via a file or standard input; `ldapadd` is actually a hard link to `ldapmodify -a`.

*   `ldapsearch` — Searches for entries in the LDAP directory using a shell prompt.

- ldapdelete — Deletes entries from an LDAP directory, accepting input via a file or a shell prompt.

With the exception of ldapsearch, each of these utilities is much more easily used by referencing a file with the changes to be made rather than typing the commands one after the other. Each of their respective man pages covers the syntax of these files.

To import or export blocks of information with a slapd directory or perform similar administrative tasks, different utilities, located in /usr/sbin, are required:

- slapadd — Adds entries from an LDIF file to an LDAP directory. For example, execute /usr/sbin/slapadd -l *ldif* where *ldif* is the name of the LDIF file containing the new entries.

- slapcat — Pulls entries out of an LDAP directory and saves them in an LDIF file. For example, execute /usr/sbin/slapcat -l *ldif* where *ldif* is the name of the target LDIF file to contain the entries from the LDAP directory.

- slapindex — Reindexes the slapd database based on the actual current database content. Execute /usr/sbin/slapindex to begin reindexing.

- slappasswd — Generates a user password value for use with ldapmodify or the rootpw value in /etc/openldap/slapd.conf. Execute /usr/sbin/slappasswd to create the password.

WARNING

Be sure to stop slapd before using slapadd, slapcat or slapin-dex. Otherwise, you are risking the consistency of your LDAP database.

See the man pages for each of these utilities for more information about how to use them.

15.8 Modules for Adding Extra Functionality to LDAP

Red Hat Linux includes several packages that add functionality to LDAP.

The nss_ldap module is an LDAP module for the **Solaris Nameservice Switch** (NSS). NSS is a set of C library extensions necessary for accessing LDAP directory information, instead of or in addition to the **Network Information Service** (NIS) name service and/or flat files. The nss_ldap module is needed to use LDAP as a native name service.

The pam_ldap module is needed to integrate LDAP authentication into the Pluggable Authentication Modules (PAM) API. If you use pam_ldap, users can authenticate and change their password

using LDAP directories. The `nss_ldap` and `pam_ldap` modules are provided in the `nss_ldap` package.

Red Hat Linux also includes LDAP modules for the Apache Web server. The `auth_ldap` module is for authenticating HTTP clients against the user entries in an LDAP directory. The `php-ldap` module adds LDAP support to the PHP4 HTML-embedded scripting language. The `auth_ldap` and `php-ldap` modules must be compiled into Apache as **Dynamic Shared Objects** (**DSO**s) in order to work.

15.9 LDAP How To: A Quick Overview

This section provides a quick overview of the steps you will need to take to get an LDAP directory working.

1. Make sure the `openldap`, `openldap-servers`, and `openldap-clients` RPMs, and any other LDAP-related RPMs that you need, are installed.

2. Refer to either the Quick Start Guide at the OpenLDAP site (http://www.openldap.org/doc/ad-min/quickstart.html — start at "Edit the configuration file" since the LDAP files are already installed) or see the LDAP Linux HOWTO (http://www.redhat.com/mirrors/LDP/HOWTO/LDAP-HOWTO.html) for instructions on using LDAP on your system. Both of these documents cover the rest of these steps in more detail.

3. Edit the `/etc/openldap/slapd.conf` file to configure it for your system. (See Section 15.6.1, *Edit /etc/openldap/slapd.conf* for more information on editing `slapd.conf`.)

4. Start `slapd` by typing `/sbin/service/ldap start`. (After you have configured LDAP correctly, you should use chkconfig or `serviceconf` to configure LDAP to start up with the system.)

5. Create your LDAP directory (examples of LDAP entries are provided at the PADL Software website at http://www.padl.com/ldap_examples.html).

6. Add entries to your LDAP directory with `ldapadd` or with a script.

7. Use `ldapsearch` to see if `slapd` is working.

8. At this point, your LDAP directory should exist. The next step is to configure your LDAP-enabled applications so that they can use the LDAP directory.

15.10 Configuring Your System to Authenticate Using OpenLDAP

This section provides a brief overview of how to configure your Red Hat Linux system to authenticate using OpenLDAP. Unless you're an OpenLDAP expert, you will probably need more documentation

than is provided here. Please refer to the references provided in Section 15.11, *Additional Resources* for more information.

15.10.1 Install the Necessary LDAP Packages

First, you should make sure that the appropriate packages are installed on both the LDAP server and the LDAP client machines. The LDAP server needs the `openldap-server` package.

The LDAP client machines need the following packages installed: `openldap`, `openl-dap-clients`, `auth_ldap`, and `nss_ldap`.

15.10.2 Edit the Configuration Files

Edit `/etc/openldap/slapd.conf`

Next, edit the `slapd.conf` file to make sure it matches the specifics of your organization.

Please refer to Section 15.6.1, *Edit /etc/openldap/slapd.conf* for instructions on editing `slapd.conf`.

Edit `ldap.conf`

Edit the `ldap.conf` files in `/etc` and in `/etc/openldap` on the LDAP server and clients.

Edit `/etc/ldap.conf`, the configuration file for `nss_ldap` and `pam_ldap`, to reflect your organization and search base. The file `/etc/openldap/ldap.conf` is the configuration file for the command line tools like `ldapsearch`, `ldapadd`, etc., and it will also need to be edited for your LDAP setup. Client machines will need to have both of these files modified.

Edit `/etc/nsswitch.conf`

To use `nss_ldap`, you'll need to add `ldap` to the appropriate fields in `/etc/nsswitch.conf`. (Be very careful when editing this file; be sure that you know what you're doing.) For example:

```
passwd: files ldap
shadow: files ldap
group: files ldap
```

PAM and LDAP

To have standard PAM-enabled applications use LDAP for authentication, run `authconfig` and select **Use LDAP**. (PAM is beyond the scope of this LDAP overview, so if you need help, consult Chapter 7, *Pluggable Authentication Modules (PAM)* and the PAM man pages.)

15.10.3 Migrate Your Old Authentication Information to LDAP Format

The `/usr/share/openldap/migration` directory contains a set of shell and Perl scripts for migrating your old authentication information into LDAP format. (You must have Perl installed on your system to use these scripts.)

First, you'll need to modify the `migrate_common.ph` file so that it reflects your domain. The default DNS domain should be changed from:

```
$DEFAULT_MAIL_DOMAIN = "padl.com";
```

to something like:

```
$DEFAULT_MAIL_DOMAIN = "your_company.com";
```

The default base should also be changed, from:

```
$DEFAULT_BASE = "dc=padl,dc=com";
```

to something like:

```
$DEFAULT_BASE = "dc=your_company,dc=com";
```

Next, you'll need to decide which script to use. The following table may help you:

Table 15–1   LDAP Migration Scripts

Existing name service	Is LDAP running?	Use this script:
`/etc` flat files	yes	`migrate_all_on-line.sh`
`/etc` flat files	no	`migrate_all_of-fline.sh`
NetInfo	yes	`migrate_all_net-info_online.sh`
NetInfo	no	`migrate_all_net-info_offline.sh`
NIS (YP)	yes	`migrate_all_nis_on-line.sh`
NIS (YP)	no	`migrate_all_nis_of-fline.sh`

Run the appropriate script based on your existing name service.

The `README` and the `migration-tools.txt` files in `/usr/share/openldap/migration` provide more details on how to migrate the information.

15.11 Additional Resources

A lot of useful information concerning LDAP is available. Please review these sources, especially the OpenLDAP website and the LDAP HOWTO, before beginning to set up and configure LDAP on your system.

15.11.1 Installed Documentation

- The `ldap` man page is a good place to get started for an introduction to LDAP. Also, man pages exist for the various LDAP daemons and utilities. Please check the man pages if you need more information on `ldapmodify`, `ldapsearch`, and the like.

- `/usr/share/docs/openldap-versionnumber` — Contains a general `README` document and miscellaneous information.

15.11.2 Useful Websites

- http://www.openldap.org — Home of the OpenLDAP Project, the collaborative effort to develop a "robust, commercial-grade, fully featured, and open source LDAP suite of applications and development tools."

- http://www.redhat.com/mirrors/LDP/HOWTO/LDAP-HOWTO.html — LDAP Linux HOWTO document, covering the installation through authentication and logging.

- http://www.padl.com — Developers of `nss_ldap` and `pam_ldap`, among other useful LDAP tools.

- http://www.innosoft.com/ldapworld — Contains information concerning LDAP RFCs and LDAP version 3 specifications.

- http://www.kingsmountain.com/ldapRoadmap.shtml — Jeff Hodges' LDAP Road Map contains links to several useful FAQs and emerging news concerning the LDAP protocol.

- http://www.rudedog.org/auth_ldap — Home of the `auth_ldap` authentication module for Apache.

- http://www.stanford.edu/~bbense/Inst.html — Discusses the use of LDAP with Sendmail.

- http://www.webtechniques.com/archives/2000/05/wilcox — A useful look at managing groups in LDAP.

- http://www.ldapman.org/articles — Articles that offer a good introduction to LDAP, including methods to design an directory tree and customizing directory structures.

15.11.3 Related Books

- *Implementing LDAP* by Mark Wilcox; Wrox Press, Inc.

- *Understanding and Deploying LDAP Directory Services* by Tim Howes et al.; Macmillan Technical Publishing

16   Email

Email is one of the most widely used services on the Internet. Red Hat Linux offers many ways for you to utilize email, whether you are a desktop user or a system administrator.

This chapter looks at popular email protocols that are in use today and various programs designed to accomplish different types of tasks when dealing with email.

16.1 Protocols

Email, like other network services, uses a variety of protocols. These protocols allow different machines, often running different operating systems and utilizing different email programs, to communicate with one another and transfer mail so it arrives in the proper place.

The following protocols are those most commonly used to transfer email from system to system.

16.1.1 IMAP

The **Internet Message Access Protocol (IMAP)** is a method used by email client applications to access remotely stored messages. When using IMAP, commonly called IMAP4 after the version of the protocol used, the email messages remain on the remote mail server, where the user can read or delete them and create, rename, or delete mailboxes to store the email.

In addition, IMAP is fully compatible with important Internet messaging standards, such as the Multipurpose Internet Mail Extensions (MIME), to allow attachments to be received. Many email clients that use IMAP can also be configured to cache a copy of the messages locally, so that you can browse previously read messages when you are not directly connected to the IMAP server.

IMAP is primarily utilized by users that may access their email using multiple machines, as messages are stored in a central location and can be accessed by any system with an IMAP mail client and a connection to the remote IMAP server. Also, users that connect to the Internet or a private network via a low-bandwidth connection often use IMAP because only the email header information is pulled off at first. This allows them to defer the downloading of messages containing large attachments until a time when their limited bandwidth is not in use. In the same way, email that the user does not want can be deleted without viewing the message body, saving the need to even download it through their network connection.

The **Request for Comment (RFC)** documents that cover IMAP contain the assorted details and specifics about how the protocol is designed to work. RFC-1730 first defined the way IMAP is used in version 4, but RFC-2060 discusses the current IMAP implementation used with many IMAP servers, called version IMAP4rev1.

The `imap` package in Red Hat Linux allows users to connect to your system and receive their email using IMAP. Secure IMAP connections are supported through Secure Socket Layer (SSL) technology built into the `imapd` daemon, allowing it to use the `/usr/share/ssl/certs/imapd.pem` certificate file. The `stunnel` program is not required to provide SSL-encryption for IMAP connections, though it can be used. See Section 16.6.2, *Secure Email Servers* for more information concerning these two encryption options.

Other free, as well as commercial, IMAP clients and servers are available, many of which extend the IMAP protocol and provide additional functionality. A comprehensive list can be found at http://www.imap.org/products/longlist.htm.

16.1.2 POP

The **Post Office Protocol (POP)** allows email clients to pull off email from remote servers and save those messages on their local machine. Most POP email clients are automatically configured to delete the message on the email server after it has been successfully transferred to the client's system, though this can usually be changed.

To connect to a POP server, the email client opens a TCP connection to port 110 on the server. At the time the connection is made, the POP server sends the POP client a greeting, after which the two machines send each other commands and responses specified in the protocol. As part of this communication, the POP client is asked to authenticate itself in what is called the **Authentication State**, where the user's username and password are sent to the POP server. If authentication is successful, then the POP client moves on to the **Transaction State**, where commands like `LIST`, `RETR`, and `DELE` can be used to list, download, and delete the messages from the server, respectively. Messages set to be deleted are not actually removed from the server until the POP client sends the `QUIT` command to end the session. At this point, the POP server enters the **Update State**, where it deletes the flagged messages and cleans up any resources remaining from this session.

POP is a much simpler protocol than IMAP, due to the fact that fewer commands can be sent between the client and the server. POP is also somewhat more popular, although most major email clients can use either protocol quite well.

Most POP users only have one system that they use to read email, and they download their messages to that machine for storage. POP also works well if you do not have a constant connection to the Internet or network containing your mail server, although IMAP can now be configured to store messages locally so that you can view them when disconnected from the network.

Several RFCs cover the POP protocol, but RFC-1939 defines the basic outline of POP3, the current version.

Occasionally, you may run into lesser-used POP protocol variants:

- *APOP* — POP3 with MDS authentication, where an encoded hash of your password is sent from the email client to the server rather sending the password in plaintext.

- *KPOP* — POP3 with Kerberos authentication. See Chapter 8, *Using Kerberos 5 on Red Hat Linux* for more information concerning Kerberos authentication.

- *RPOP* — POP3 with RPOP authentication, which utilizes an ID issued per user, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so RPOP is no more secure than standard POP.

Many POP servers, clients, and assorted other applications are available with Red Hat Linux. If you prefer a graphical email client, Mozilla Mail is an excellent choice. In addition, other email utilities, such as Fetchmail, can retrieve email via POP. If you are using your Red Hat Linux system as a mail server, the `imap` package contains POP2 (`ipop2`) and POP3 (`ipop3`) daemons in the `/usr/sbin` directory.

16.1.3 SMTP

While the IMAP and POP protocols involve allowing a user to be able to receive and read their email, the **Simple Mail Transfer Protocol (SMTP)** is used to send email. Outgoing email uses SMTP to move from the client's machine to the server, where it moves along toward its final destination. Or, two email servers attempting to move a message between one another utilize SMTP so they can communicate, even if they are totally different platforms.

SMTP uses port 25 on the server for its communication. A basic SMTP exchange begins with the connecting system issuing a `MAIL From:` `<email-address>` command to initiate exchange. The receiving system responds with a `250` message to acknowledge receipt of the first command. Then, the connecting system hands the email addresses to receive the message to the receiving system, followed by a `DATA` message. This tells the receiving system that the next part of the communication will be the actual body of the email message. When the connecting system is finished with the email message, it places a single dot (`.`) on a line. At that point, the message is considered sent.

SMTP also handles cases where email needs to be forwarded between systems, when the receiving system knows where to send the message. The protocol can verify that certain users are indeed served by a particular mail server (the `VRFY` command) or expand a mailing list (the `EXPN` command). Email can also be relayed between two SMTP servers, if both systems permit such activity.

Unlike IMAP and POP, SMTP does not require authentication in its most basic form. This has made possible a lot of spam, due to the fact that a non-local user could use your system to send or relay mail to entire lists of recipients, using your system's resources and bandwidth to deliver the junk mail. Modern SMTP applications have gone to great length to minimize this behavior by restricting relaying and allowing only known hosts to send email.

RFC-821 outlines the basic behavior of SMTP, but several SMTP extensions, made possible by RFC-1869, have added additional functionality to SMTP over the years by making new commands available. By initiating a conversation with an SMTP server with an `EHLO` command rather than `HELO`, the connecting server can identify itself as one that supports SMTP extensions. The receiving server

answers with a `250` line containing the various SMTP extensions it supports. Then, the connecting server can use the supported extensions as it wishes to accomplish the goals of the communication.

One noticeable extension concerns the addition of SMTP Authentication through the `AUTH` command as outlined in RFC-2554. Another widely used SMTP extension is detailed in RFC-2034, which discusses the use of dot-separated, standardized error codes to be used between SMTP applications. Reading the various RFCs that involve SMTP provides a background to the way email moves around the Internet. In addition, you can connect to an SMTP server via telnet by specifying port 25, such as `telnet localhost 25`. Executing a few commands and sending a mail manually is a good way to get a handle on how SMTP communications occur.

Red Hat Linux uses Sendmail as its SMTP program by default, although various other applications are available that have many of the same features but are easier to use, such as Postfix. Many email client applications connect to an SMTP server directly to send messages, although you can configure a locally running SMTP service to deliver your email for you, either directly to the final destination or to a central email server to then be forwarded on.

16.2 Different Types of Email Programs

There are three types of email programs, all of which perform a specific role in the process of moving and managing email messages. While most users are only aware of the specific email program they use to receive and send messages, each of these types is important for making sure email arrives at the correct destination.

16.2.1 Mail User Agent

A **Mail User Agent (MUA)** is a program that, at the very least, allows a user to read and compose email messages. An MUA is often referred to as an email client. Of course, many MUAs help users do more than that, including retrieving messages via the POP or IMAP protocols, setting up mailboxes to store messages, or helping present new messages to a Mail Transfer Agent program that will deliver them to their final destination.

MUA programs may be graphical, such as Mozilla Mail, or have a very simple, text-based interface, such as Mutt, or Pine.

16.2.2 Mail Transfer Agent

A **Mail Transfer Agent (MTA)** transfers email messages between machines using SMTP. A message may involve several MTAs as it moves to its intended destination. Most users are totally unaware of the presence of MTAs, even though every email message is sent through at least one MTA.

While the delivery of messages between machines may seem rather straightforward, the entire process of deciding if a particular MTA can or should accept a message for delivery to a remote host is quite

complicated. In addition, due to problems from spam, use of a particular MTA is usually restricted by the MTA's own configuration or network access to the system running it.

Many of the larger and more complex MUAs can also be used to send email. However, this action should not be confused with the actions of a true MTA. In order for users not running their own MTA to move outbound messages off of their machine and onto a remote machine for delivery, they must use a capacity in the MUA that transfers the message to an MTA they are authorized to use. However, the MUA does not directly deliver the message to the intended recipient's email server — that role is reserved for the MTA.

Red Hat Linux uses Sendmail as its default MTA, though others may be used in its place. It is important that you disable the currently running MTA before starting up another one, as both will try to claim port 25, the default SMTP port.

16.2.3 Mail Delivery Agent

A **Mail Delivery Agent (MDA)** is utilized by the MTA to deliver email to a particular user's mailbox. In many cases, an MDA is actually a **Local Delivery Agent (LDA)**, such as /bin/mail or Procmail. However, Sendmail can also play the role of an MDA, such as when it accepts a message for a local user and appends it to their email spool file. Any program that actually handles a message for delivery to the point where it can be read by an MUA can be considered an MDA. Note that MDAs do not transport messages between systems or interface with the end user.

Many users do not directly utilize MDAs, because only MTAs and MUAs are necessary to send and receive email. However, some MDAs may be used to sort messages before they are read by a user, which is a big help if you receive a lot of email.

16.3 Sendmail

Red Hat Linux uses Sendmail as its MTA to deliver messages, whether they are intended for users on the same system or for remote destinations. Other MTAs do exist (and can be used well with Red Hat Linux), but most administrators elect to use Sendmail as their MTA due to its power, scalability, and compliance to important Internet standards, such as SMTP.

Sendmail's core duty, like other MTAs, is to safely move email between hosts, usually utilizing the SMTP protocol. However, Sendmail is highly configurable, allowing you to control almost every aspect of how email is handled, including the protocol to be used.

16.3.1 History

Sendmail's roots can be traced to the birth of email, occurring in the decade before the birth of ARPANET, the precursor to the Internet. In those days, every user's mailbox was a file that only they had rights to read, and mail applications simply added text to that file. Every user had to wade through their mail file to find any old mail, and reading new mail was a chore. The first actual transfer

of a mail message file from one host to another didn't take place until 1972, where email began to to be moved by FTP over the NCP network protocol. This easier method of communication quickly became popular, even to the point where it made up most of ARPANET's traffic in less than a year.

However, a lack of standardization between competing protocols made email much harder to send from some systems, and this continued until the ARPANET standardized on TCP/IP in 1982. A new protocol, SMTP, materialized for message transporting. These developments, combined with HOSTS files being replaced with DNS, allowed full-featured MTAs to materialize. Sendmail, which grew out of an earlier email delivery system called Delivermail, quickly became the standard as the Internet began to expand and be widely utilized.

16.3.2 Purpose and Limitations

It is important to be aware of what Sendmail is and what it can do for you as opposed to what it is not. In these days of monolithic applications that fulfill multiple roles, you might initially think that Sendmail is the only application you need to run an email server within your organization. Technically, that is true, as Sendmail can spool mail to your users' directories and accepts new email via the command line. But, most users actually require much more than simple email delivery. They usually want to interact with their email using an MUA that uses POP or IMAP to download their messages to their local machine. Or, they may prefer a Web interface to gain access to their mailbox. These other applications can work in conjunction with Sendmail and SMTP, but they actually exist for different reasons and can operate separately from one another.

It is beyond the scope of this section to go into all that Sendmail should or could be configured to do. With literally hundreds of different options and rule sets, entire volumes are dedicated to helping explain everything that can be done and how to fix things that go wrong. You should consult the many excellent online and offline sources of information on Sendmail in order to shape it to fit your exact specifications.

However, you should understand what files are installed with Sendmail by default on your system and know how to make basic configuration changes. You should also be aware of how to stop unwanted email (spam) and how to extend Sendmail with the **Lightweight Directory Access Protocol (LDAP)**.

16.3.3 The Default Sendmail Installation

While you can download the source code for Sendmail and build your own copy, many users prefer to use the version of Sendmail installed by default with their Red Hat Linux system. You can also use the Red Hat Linux CD-ROMs to reinstall the sendmail RPM at a later point.

After installation, the sendmail executable is placed in the /usr/sbin directory.

Sendmail's lengthy and detailed configuration file (sendmail.cf) is installed in /etc. You should not edit the sendmail.cf file directly unless you know exactly what you are doing, due to the fact it is very lengthy and complex. Instead, to make configuration changes to Sendmail,

edit the /etc/mail/sendmail.mc file and use the included m4 macro processor to create a new /etc/sendmail.cf (after backing up the original /etc/sendmail.cf, of course). More information on configuring Sendmail can be found in Section 16.3.4, *Common Sendmail Configuration Changes*.

Various Sendmail configuration files are installed in /etc/mail including:

- access — Specifies which systems can use Sendmail for relaying email.

- domaintable — Allows you to provide domain name mapping.

- local-host-names — The place where you include all aliases for your machine.

- mailertable — Specifies instructions that override routing for particular domains.

- virtusertable — Permits you to do a domain-specific form of aliasing, allowing multiple virtual domains to be hosted on one machine.

Several of the configuration files in /etc/mail, such as access, domaintable, mailertable and virtusertable, must actually store their information in database files before Sendmail can use any configuration changes. To include any changes you make to these configurations in their database files, you must run the makemap hash /etc/mail/*<name>* < /etc/mail/*<name>* command, where *<name>* is the name of the configuration file to convert.

For example, if you want all email addressed to any domain.com account to be delivered to bob@otherdomain.com, you need to add a line to the virtusertable file:

Figure 16–1   `virtusertable` example

```
@domain.com        bob@otherdomain.com
```

Then, to add this new information to the virtusertable.db file, execute makemap hash /etc/mail/virtusertable < /etc/mail/virtusertable as root. This will create a new virtusertable.db that contains the new configuration.

16.3.4 Common Sendmail Configuration Changes

A default sendmail.cf file will be installed in /etc. The default configuration should work for most SMTP-only sites. It will *not* work for UUCP (UNIX to UNIX Copy) sites; you will need to generate a new sendmail.cf if you must use UUCP mail transfers.

If you need to generate a new /etc/sendmail.cf file to configure Sendmail, you should utilize the m4 macro processor. If you ever edit the /etc/mail/sendmail.mc to add functionality to Sendmail, backup your current /etc/sendmail.cf file, generate a new one by executing the m4 /etc/mail/sendmail.mc > /etc/sendmail.cf command, and add any previous changes

from the /etc/sendmail.cf you backed up to the new /etc/sendmail.cf file. After creating a new /etc/sendmail.cf, you must restart Sendmail to make it take effect. The easiest way to do this is to type the service sendmail restart command as root.

By default, the m4 macro processor is installed with Sendmail. The m4 macro processor is included with the sendmail-cf package, which is installed in /usr/share/sendmail-cf.

You should consult the /usr/share/sendmail-cf/README file before you edit any of the files in the directories under the /usr/share/sendmail-cf directory, as they can affect how future /etc/sendmail.cf files are configured.

Masquerading

One common Sendmail configuration is to have a single machine act as a mail gateway for all the machines on the network. For instance, a company may want to have a machine called mail.bigcorp.com that does all their mail. On that machine, add the names of machines for which mail.bigcorp.com will handle mail to /etc/mail/local-host-names. Here is an example:

Figure 16–2   Example settings for `local-host-names`

```
# sendmail.cw - include all aliases for your machine
# here.
torgo.bigcorp.com
poodle.bigcorp.com
devel.bigcorp.com
```

On the other machines, torgo, poodle, and devel, edit /etc/sendmail.cf to "masquerade" as mail.bigcorp.com when sending mail and to forward any local mail processing to bigcorp.com. Find the DH and DM lines in /etc/sendmail.cf and edit them as such:

Figure 16–3   Example settings for `sendmail.cf`

```
# who I send unqualified names to
# (null means deliver locally)
DRmail.bigcorp.com

# who gets all local email traffic
DHmail.bigcorp.com

# who I masquerade as (null for no masquerading)
DMbigcorp.com
```

With this type of configuration, all mail sent will appear as if it were sent from bigcorp.com, and any mail sent to torgo.bigcorp.com or the other hosts will be delivered to mail.bigcorp.com.

16.3.5 Stopping Spam with Sendmail

Email **spam** can be defined as unnecessary and unwanted email received by a user that probably does not know the sender and never requested the communication. It is a very disruptive, costly, and wide-spread abuse of Internet communication standards.

Sendmail has made it (relatively) easy to block new spamming techniques being employed to send junk email using your system. It even blocks many of the more usual spamming methods by default. You would need to consciously activate them by changing your `/etc/mail/sendmail.mc` file in a particular way to make your system susceptible.

For example, forwarding of SMTP messages, also referred to as **SMTP relaying**, has been disabled by default since Sendmail version 8.9. Before this change occurred, Sendmail would direct your mail host (`x.org`) to accept messages from one party (`y.com`) and send them to a different party (`z.net`). Now, however, you have to specifically tell Sendmail to permit a domain to relay mail through your domain. Simply edit the `/etc/mail/relay-domains` file and restart Sendmail by typing the `service sendmail restart` command as root to activate the changes.

However, many times, your users may be bombarded with spam from other servers throughout the Internet that are beyond your control. In these instances, you can use Sendmail's access control features available through the `/etc/mail/access` file. As root, add the domains that you would like to block or specifically allow access, such as:

Figure 16–4 Example settings for `access`

```
badspammer.com          550 Go away and don't spam us anymore
tux.badspammer.com      OK
10.0                    RELAY
```

Because `/etc/mail/access` is a database, you need to use `makemap` to activate your changes by recreating the database map. This is easily done by running the `makemap hash /etc/mail/access < /etc/mail/access` command as root.

This example shows that any email sent from `badspammer.com` would be blocked with a 550 RFC-821 compliant error code and message back to the spammer, except for email sent from the `tux.badspammer.com` sub-domain, which would be accepted. The last line shows that any email sent from the 10.0.*.* network can be relayed through your mail server.

As you might expect, this example only scratches the surface of what Sendmail can do in terms of allowing or blocking access. See the `/usr/share/doc/sendmail/README.cf` for more detailed information and examples.

16.3.6 Using Sendmail with LDAP

Using the **Lightweight Directory Access Protocol (LDAP)** is a very quick and powerful way to find specific information about a particular user from a much larger group. For example, you could use an

LDAP server to look up a particular email address from a common corporate directory by a user's last name. In this kind of implementation, LDAP is largely separate from Sendmail, with LDAP storing the hierarchical user information and Sendmail only being given the result of LDAP queries in pre-addressed email messages.

However, Sendmail supports a much greater integration with LDAP, where it uses LDAP to replace separately maintained files, such as `aliases` and `virtusertables`, on different mail servers that work together to support a medium- to enterprise-level organization. In short, you can use LDAP to abstract the mail routing level from Sendmail and its separate configuration files to a powerful LDAP cluster that is being leveraged by many different applications.

The current version of Sendmail contains support for LDAP. To extend your Sendmail server using LDAP, first get an LDAP server, such as OpenLDAP, running and properly configured. Then, you need to edit your `/etc/mail/sendmail.mc` to include:

Figure 16–5   Example settings for LDAP in `sendmail.mc`

```
LDAPROUTE_DOMAIN('yourdomain.com')dnl
FEATURE('ldap_routing')dnl
```

Note

This is only for a very basic configuration of Sendmail with LDAP. Your configuration should differ greatly from this depending on your implementation of LDAP, especially if you wish to configure several Sendmail machines to use a common LDAP server.

Consult `/usr/share/doc/sendmail/README.cf` for detailed LDAP routing configuration instructions and examples.

Next, recreate your `/etc/sendmail.cf` file by running `m4` and restarting Sendmail. See Section 16.3.4, *Common Sendmail Configuration Changes* for instructions on doing this.

For more information on LDAP, see Chapter 15, *Lightweight Directory Access Protocol (LDAP)*.

16.4 Fetchmail

Fetchmail is a program that can retrieve email from remote servers for on-demand TCP/IP connections. Many users appreciate the ability to separate the process of downloading their messages located on a remote server from the process of reading and organizing their email in an MUA. Designed with the needs of dial-up users in mind, Fetchmail will connect and quickly download all of your email messages to your mail spool file using any number of protocols, including the POP3 and IMAP. It can even forward your email messages to an SMTP server, if necessary.

Before attempting to use Fetchmail, be sure that it is installed on your system. If it is not, you can install it using the `fetchmail` RPM on the Red Hat Linux CD-ROMs.

Fetchmail is configured for each user through the use of a `.fetchmailrc` file in the user's home directory. A program bundled with Fetchmail, called `fetchmailconf`, is very helpful for configuring a basic `.fetchmailrc` file that you can customize at will.

Using preferences in the `.fetchmailrc` file, Fetchmail will check for email on a remote server and pull it off, attempting to deliver it to port 25 on the local machine, using the local MTA to place the email in the correct user's spool file. If Procmail is available, it can then be utilized to filter the email and place it in a mailbox so that it can be read by an MUA.

16.4.1 Fetchmail Configuration Options

Although it is possible to pass all options on the command line necessary to check for email on a remote server when executing Fetchmail, using a `.fetchmailrc` file is much easier. All of your configuration options go in the `.fetchmailrc` file, but you can override them at the time Fetchmail is run by specifying that option on the command line.

A user's `.fetchmailrc` file is divided into three particular types of configuration options:

- *global options* — Gives Fetchmail instructions that control the operation of the program or provide settings for every connection that checks for email.

- *server options* — Specifies necessary information about the server being polled, such as the hostname, as well as preferences you would like to see enforced with a particular email server, such as the port to check or number of seconds to wait before timing out. These options affect every user option utilized with that server.

- *user options* — Contains information, such as username and password, necessary to authenticate and check for email using a particular email server.

Global options go at the top of the `.fetchmailrc` file, followed by one or more server options, each of which designate a different email server that Fetchmail should check. User options follow server options for each user account you would like to check on that email server. Like server options, multiple user options may be specified for use with a particular server, such as when you would like to check multiple email accounts on the same server.

Server options are called into service in the `.fetchmailrc` file by the use of a special option verb, `poll` or `skip`, that precedes any of the server information. The `poll` action tells Fetchmail to use this server option when it is run, which actually checks it for email using the various user options. Any server options after a `skip` action, however, are not checked unless you specify this server's hostname when Fetchmail is invoked. The `skip` option allows you to set up test configurations in `.fetchmailrc` and only check using that server when specifically desired, without affecting any currently working configurations.

A sample `.fetchmailrc` file looks like this:

Figure 16–6   Example of a basic `.fetchmailrc` file

```
set postmaster "user1"
set bouncemail

poll pop.domain.com proto pop3
    user 'user1' there with password 'secret' is user1 here

poll mail.domain2.com
    user 'user5' there with password 'secret2' is user1 here
    user 'user7' there with password 'secret3' is user1 here
```

In this example, the global are options set so the user is sent email as a last resort (`postmaster` option) and all email errors are sent to the postmaster instead of the sender (`bouncemail` option). The `set` action tells Fetchmail that this line contains a global option. Then, two email servers are specified, one set to check using POP3, and the other for trying various protocols to find one that works. Two users are checked using the second server option, but all email found for any users is sent to user1's mail spool. This allows multiple mailboxes to be checked on multiple servers, while appearing in a single MUA inbox. Each user's specific information begins with the `user` action.

Note

You do not have to place your password in the `.fetchmailrc` file. You can omit the `with password '<password>'` section. Fetchmail will then ask for your password when it is started with the `fetchmail` command.

While you can set up your `.fetchmailrc` file manually, it is much easier to let the included `fetchmailconf` program do it for you. However, when testing new configurations, it is usually easier to edit the `.fetchmailrc` file directly.

As expected with a program that services such a mature network service as email and utilizes so many protocols, Fetchmail contains many different global, server, and local options. Many of these options are rarely used or only apply to very specific situations. The `fetchmail` man page explains each of these options in detail, but the most common ones are listed here.

Global Options

Each global option should be placed on a single line after a `set` action.

• `daemon <seconds>` — Tells Fetchmail to automatically use daemon-mode, where it will stay in the background and poll for mail at the interval specified.

- `postmaster` — Gives Fetchmail a local user to send mail to in case of delivery problems.

- `syslog` — Tells Fetchmail to start logging error and status messages in the system's log file. By default, this is `/var/log/maillog`.

Server Options

Place server options on their own line in `.fetchmailrc` after a `poll` or `skip` action.

- `auth <auth-type>` — Specifies the type of authentication to be used. By default, `pass-word` authentication is utilized, but some protocols support other types of authentication, including `kerberos_v5`, `kerberos_v4`, and `ssh`. If the `any` authentication type is used, Fetchmail will first try methods that do not require a password, then methods that mask your password, and finally attempt to send your password in the clear to authenticate to the server.

- `interval <number>` — Tells Fetchmail to only poll this server every `<number>` of times that it checks for email on all configured servers. This option might be used with email servers where you rarely receive messages.

- `port <port-number>` — Overrides the default port number for a specified protocol.

- `proto <protocol>` — Tells Fetchmail to use a specific protocol, such as `pop3` or `imap`, to check for messages on this server.

- `timeout <seconds>` — Configures Fetchmail to give up after a certain interval of server inactivity. If this value is not set, a default of `300` seconds is assumed.

User Options

User options may be placed on their own lines beneath a server option or on the same line as the server option. In either case, the user options follow the `user` option (defined below).

- `fetchall` — Orders Fetchmail to download all messages in the queue, including messages that have already been viewed. By default, Fetchmail only pulls down new messages.

- `fetchlimit <number>` — Only allows a certain number of messages to be pulled off before stopping.

- `flush` — Tells Fetchmail to delete all previously viewed messages in the queue before pulling down the new messages.

- `limit <max-number-bytes>` — Allows you to specify that only messages below a particular size may be retrieved. This option is useful with slow network links, when a large message will take too long to download.

- `password '<password>'` — Specifies the password to be used for this user.

- `preconnect "<command>"` — Tells Fetchmail to execute the specified command before retrieving messages for this user.

- `postconnect "<command>"` — Tells Fetchmail to execute the specified command after retrieving messages for this user.

- `ssl` — Allows Fetchmail to collect the message via an encrypted SSL connection, if the server supports this.

- `user "<username>"` — Sets the username used by Fetchmail to retrieve messages. This option should be listed before any other user options.

16.4.2 Fetchmail Command Options

Most of the Fetchmail options that may be used on the command line, when executing the `fetchmail` command, mirror the `.fetchmailrc` configuration options. This is done so that Fetchmail may be used with or without a configuration file. Most users will not use these options on the command line, as it is easier to leave them in the `.fetchmailrc` file to be utilized whenever Fetchmail is run.

However, there are times when you may want to run the `fetchmail` command with other options for a particular purpose. As any options specified at the command line override configuration file options, you can also issue command options to temporarily override a `.fetchmailrc` setting that is causing an error.

Informational or Debugging Options

Certain options used after the `fetchmail` command can provide you with important information.

- `--configdump` — Displays every possible option based on information from `.fetchmailrc` and Fetchmail defaults. No email is retrieved for any users when using this option.

- `-s` — Executes Fetchmail in silent mode, preventing any messages, other than errors, from appearing after the `fetchmail` command.

- `-v` — Executes Fetchmail in verbose mode, displaying every communication between Fetchmail and the remote email servers.

- `-V` — Causes Fetchmail to display detailed version information, list its global options, and show settings to be used with each user, including the email protocol and authentication method. No email is retrieved for any users when using this option.

Special Options

These options are occasionally useful for overriding defaults often found in the `.fetchmailrc` file.

- `-a` — Tells Fetchmail to download all messages from the remote email server, whether new or previously viewed. By default, Fetchmail will only download new messages.

- `-k` — Causes Fetchmail to leave the messages on the remote email server after downloading them. This option overrides the default behavior of deleting messages after downloading them.

- `-l <max-number-bytes>` — Tells Fetchmail to not download any messages over a particular size and leave them on the remote email server.

- `--quit` — Quits the Fetchmail daemon process.

More commands and `.fetchmailrc` options can be found on the `fetchmail` man page.

16.5 Procmail

Procmail allows you to filter email as it is received from a remote email server, or placed in your spool file on a local or remote email server. It is powerful, gentle on system resources, and widely utilized. Procmail, commonly referred to as a **Local Delivery Agent (LDA)**, plays a small role in delivering email to be read by an MUA.

In order to use Procmail, it must first be installed. Type the `rpm -q procmail` command to see if the `procmail` package is installed. If, for some reason, Procmail is not on your system, install it from the Red Hat Linux installation CD-ROMs.

Procmail can be invoked in several different ways. As email is placed on your email spool file, Procmail can be configured to start up, filter the email to locations configured for use with your MUA, and quit. Or, your MUA could be configured to bring up Procmail any time a message is received so that messages are moved into their correct mailboxes. In many cases, the presence of a `.procmailrc` file in the user's home directory will invoke Procmail, if Sendmail is being used.

The actions Procmail takes with an email are dependent upon instructions from particular **recipes**, or rules, that messages are matched against by the program. If a message matches the recipe, then the email will be placed in a certain file, deleted, or otherwise processed.

When Procmail starts, it reads the email message and separates the body from the header information. Next, Procmail looks for the `/etc/procmailrc` file and `rc` files in the `/etc/procmailrcs` directory for default, system-wide, Procmail environmental variables and recipes. Then, Procmail looks for a `.procmailrc` file in the user's home directory to find rules specific to that user. Many users also create additional `rc` files of their own for Procmail that are referred to by their `.procmailrc` file but may be turned on or off quickly if a mail filtering problem develops.

By default, no system-wide `rc` files exist in the `/etc` directory, and no user `.procmailrc` files exist. To begin using Procmail, you will need to construct a `.procmailrc` file with particular environment variables and recipes explaining what you would like to do with certain messages.

In most configurations, the decision as to whether Procmail starts and attempts to filter your email is based the existence of a user's `.procmailrc` file. To disable Procmail, but save your work on the `.procmailrc` file, move it to a similar file's name using the `mv ~/.procmailrc ~/.procmailrcSAVE` command. When you are ready to begin testing Procmail again, change the name of the file back to `.procmailrc`. Procmail will begin working again immediately.

16.5.1 Procmail Configuration

Procmail configuration files, most notably the user's .procmailrc, contain important environmental variables. These variables tell Procmail which messages to sort, what to do with the messages that do not match any recipes, and so on.

These environmental variables usually appear in the .procmailrc file at the beginning, in the following format, each on their own line:

Figure 16–7   Structure of an environmental variable line

```
<env-variable>="<value>"
```

In this example, the `<env-variable>` is the name of the variable, and the `<value>` section defines the variable.

Many environment variables are not used by most Procmail users, and many of the more important environment variables are already defined a default value. Most of the time, you will be dealing with the following variables:

- DEFAULT — Sets the default mailbox where messages that do not match any recipes will be placed.

  The default DEFAULT value is the same as $ORGMAIL.

- INCLUDERC — Specifies additional rc files containing more recipes for messages to be checked against. This allows you to break up your Procmail recipe lists into individual files that fulfill different roles, such as blocking spam and managing email lists, that can then be turned off or on by using comment characters in the user's .procmailrc file.

  For example, two lines in a user's .procmailrc file may look like this:

Figure 16–8   Example of the **INCLUDERC** option in use

```
MAILDIR=$HOME/Msgs
INCLUDERC=$MAILDIR/lists.rc
INCLUDERC=$MAILDIR/spam.rc
```

If the user would like to turn off the Procmail filtering of their email lists but leave spam control in place, they could simple comment out the first INCLUDERC line with a # character.

- LOCKSLEEP — Sets the amount of time, in seconds, between attempts by Procmail to use a particular lockfile. The default is eight seconds.

- LOCKTIMEOUT — Sets the amount of time, in seconds, that must pass after a lockfile was last modified before Procmail will assume that the lockfile is simply old and can be deleted. The default is 1024 seconds.

- LOGFILE — The location and file to contain any Procmail informational or error messages.

- `MAILDIR` — Sets the current working directory for Procmail. If set, all other Procmail paths are relative to this directory.

- `ORGMAIL` — Specifies the original mailbox, or another place to put the messages if they cannot be placed in the default or recipe-required location.

  By default, a value of `/var/spool/mail/$LOGNAME` is used.

- `SUSPEND` — Sets the amount of time, in seconds, that Procmail will pause if a necessary resource, such as swap space, is not available.

- `SWITCHRC` — Allows a user to specify an external file containing additional Procmail recipes, much like the `INCLUDERC` option, except that recipe checking is actually stopped on the referring configuration file and only the recipes on the `SWITCHRC`-specified file are used.

- `VERBOSE` — Causes Procmail to log much more information. This option is useful for debugging.

Other important environmental variables are pulled from your shell, such as `LOGNAME`, which is your login name, `HOME`, which is the location of your home directory, and `SHELL`, which is your default shell.

A comprehensive explanation of all environments variables, as well as their default values, is available on the `procmailrc` man page.

16.5.2 Procmail Recipes

New users often find the construction of recipes the most difficult part of learning to use Procmail. To some extent, this is understandable, as recipes do their message matching using **regular expressions**, which is a particular format used to specify qualifications for a matching string. However, regular expressions are not very difficult to construct and even less difficult to understand when read. Additionally, the consistency of the way Procmail recipes are written, regardless of regular expressions, makes it easy to figure out what is going on.

A thorough explanation of regular expressions is beyond the scope of this chapter. The structure of Procmail recipes is more important, and useful sample Procmail recipes can be found at various places on the Internet, including http://www.iki.fi/era/procmail/links.html. The proper use and adaptation of the regular expressions found in these recipe examples depends upon an understanding of Procmail recipe structure. Introductory information specific to basic regular expression rules can be found on the `grep` man page.

A Procmail recipe takes the following form:

Figure 16–9   Structure of a Procmail recipe

```
:0<flags>: <lockfile-name>

* <special-condition-character> <condition-1>
```

```
* <special-condition-character> <condition-2>
* <special-condition-character> <condition-N>

<special-action-character><action-to-perform>
```

The first two characters in a Procmail recipe are a colon and a zero. Various flags can optionally be placed after the zero to control what Procmail does when processing this recipe. A colon after the `<flags>` section specifies that a lockfile will be created for this message. If a lockfile is to be created, you specify its name in the `<lockfile-name>` space.

A recipe can contain several conditions to match against the message. If it has no conditions, every message will match the recipe. Regular expressions are placed in some conditions in order to facilitate a match with a message. If multiple conditions are used, they must all match in order for an action to be performed. Conditions are checked based on the flags set in the recipe's first line. Optional special characters placed after the `*` character can further control the condition.

The `<action-to-perform>` specifies what is to happen to a message if it matches one of the conditions. There can only be one action per recipe. In many cases, the name of a mailbox is used here to direct matching messages into that file, effectively sorting the email. Special action characters may also be used before the action is specified.

Delivering vs. Non-Delivering Recipes

The action used if the recipe matches a particular message determines whether recipe is considered delivering or non-delivering. A **delivering recipe** contains an action that writes the message to a file, sends the message to another program, or forwards the message to another email address. A **non-delivering recipe** covers any other actions, such as when a nesting block is used. A **nesting block** is an action contained in braces { } that designates additional actions to perform on messages that match the recipe's conditions. Nesting blocks can be nested, providing greater control for identifying and performing actions on messages.

Delivering recipes that match messages causes Procmail to perform the action specified and stop comparing the message against any other recipes. Messages that match conditions in non-delivering recipes will continue to be compared against other recipes in the current and following `rc` files. In other words, non-delivering recipes cause the message to continue through the recipes after the specified action is taken on it.

Flags

Flags are very important in determining how or if a recipe's conditions are compared to a message. The following flags are commonly used:

- A — Specifies that this recipe will only be used if the last previous recipe without an A or a flag also matched this message.

To ensure that the action on this last previous matching recipe was successfully completed before allowing a match on the current recipe, use the a flag instead.

- `B` — Parse the body of the message and look for matching conditions.

- `b` — Use the body in any resulting action, such as writing the message to a file or forwarding it. This is the default behavior.

- `c` — Generate a carbon copy of the email. This is useful with delivering recipes, since the required action can be performed on the message and a copy of the message can continue being processed in the `rc` files.

- `D` — Makes the `egrep` comparison case-sensitive. By default, the comparison process is not case-sensitive.

- `E` — Similar to the A flag, except that the conditions in this recipe are only compared to the message if the immediately preceding recipe without an E flag did not match. This is comparable to an else action.

  Use the e flag instead if you only want this recipe checked if the preceding recipe matched but the action failed.

- `f` — Uses the pipe as a filter.

- `H` — Parses the header of the message and looks for matching conditions. This occurs by default.

- `h` — Uses the header in a resulting action. This is the default behavior.

- `w` — Tells Procmail to wait for the specified filter or program to finish and report whether or not it was successful before considering the message filtered.

  If you would like to ignore "Program failure" messages when deciding whether a filter or action succeeded, use the W option instead.

Additional flags can be found in the `procmailrc` man page.

Specifying a Local Lockfile

Lockfiles are very useful with Procmail to ensure that more than one process does not try to alter a certain message at the same time. You can specify a local lockfile by placing a ： character after any flags on a recipe's first line. This will create a local lockfile based on the destination filename plus whatever has been set in the `LOCKEXT` global environment variable.

Alternatively, you can specify the name of the local lockfile to be used with this recipe after the ： character.

Special Conditions and Actions

Particular characters used before Procmail recipe conditions and actions change the way they are interpreted.

The following characters may be used after the `*` character at the beginning of a recipe's condition line:

- `!` — Inverts the condition, causing a match to occur only if the condition does not match the message.

- `<` — Checks to see if the message is under the specified number of bytes.

- `>` — Checks to see if the message is over a particular number of bytes.

The following characters are used to perform special actions:

- `!` — Tells Procmail to forward the message to the specified email addresses

- `$` — Refers to a variable set earlier in the `rc` file. This is usually used to set a common mailbox that will be referred to by various recipes.

- `|` — The pipe character tells Procmail to start a specific program to deal with this message.

- `{` and `}` — Constructs a nesting block, used to contain additional recipes to apply to matching messages.

If no special character is used at the beginning of the action line, then Procmail assumes that the action line is specifying a mailbox where the message should be written.

Recipe Examples

Procmail is an extremely flexible program, allowing you to match messages with very specific conditions and then perform detailed actions on them. As a result of this flexibility, however, composing a Procmail recipe from scratch to achieve a certain goal can be difficult for new users.

The best way to develop the skills to build Procmail recipe conditions stems from a strong understanding of regular expressions combined with looking at many examples built by others. The following very basic examples exist to serve as a demonstration of the structure of Procmail recipes and can provide the foundation for more intricate constructions.

The most basic recipes does not even contain conditions, as is demonstrated in Figure 16–10, *Example with no conditions*.

Figure 16–10 Example with no conditions

```
:0:
new-mail.spool
```

The first line starts the recipe by specifying that a local lockfile is to be created but does not specify a name, leaving Procmail to use the destination filename and the LOCKEXT to name it. No condition is specified, so every message will match this recipe and, therefore, will be placed in the single spool file called new-mail.spool, located within the directory specified by the MAILDIR environment variable. An MUA can then view the messages in this file.

This basic recipe could go at the end of all rc files to direct messages to a default location. A more complicated example might grab messages from a particular email address and throw them away, as can be seen in Figure 16–11, *Example of email sent to /dev/null*.

Figure 16–11   Example of email sent to `/dev/null`

```
:0
* ^From: spammer@domain.com
/dev/null
```

With this example, any messages sent by spammer@domain.com are immediately moved to /dev/null, deleting them.

CAUTION

Be very careful that a rule is working correctly before moving messages matching it to /dev/null, which is a permanent deletion. If your recipe conditions inadvertently catch unintended messages, you will not even know you are missing those messages unless the sender tells you.

A better solution is to point the recipe's action to a special mailbox that you can check from time to time in order to look for **false positives**, or messages that inadvertently matched the conditions. Once you are satisfied that no messages are accidentally being matched, you can delete the mailbox and direct the action to send the messages to /dev/null.

Procmail is primarily used as a filter for email, automatically placing it in the right place so that you do not have to sort it manually. The recipe in Figure 16–12, *Example of list filtering* grabs email sent from a particular mailing list and puts in the correct folder for you.

Figure 16–12   Example of list filtering

```
:0:
* ^(From|CC|To).*tux-lug
tuxlug
```

Any messages sent from the `tux-lug@domain.com` mailing list will be placed in the `tuxlug` mailbox automatically for your MUA. Note that the condition in this example will match the message if it has the mailing list's email address on the `From`, `CC`, or `To` lines.

Procmail can also be used to block spam, although this is not a good long-term solution for junk mail. Consider the following temporary spam filtering solution in Figure 16–13, *Example of a basic spam filter*, where multiple recipes are set to use a common mailbox to store the junk.

Figure 16–13   Example of a basic spam filter

```
SPAM=junk

:0:
* To??^$
$SPAM

:0:
* ^(To|CC):.*,.*,.*,.*,.*,.*,.*,.*,.*,.*,.*,
$SPAM

:0:
* ^Message-Id:.*<[^@]*>
$SPAM
```

In this example, the `junk` mailbox is associated with the `SPAM` variable, so that you can change the mailbox that holds your spam in one place. Then, three recipes look for messages to send to the `junk` mailbox.

The first recipe looks for messages that have no recipient in the `To` line. The second recipe matches any messages with 12 or more recipients. The third recipe looks for messages with a bad message ID.

These simple examples are provided to help get you started creating recipes. Consult the many Procmail online resources available from Section 16.7, *Additional Resources* to see more detailed and powerful recipes.

16.6 Security

Like any other service that flows over a network unencrypted, important email information, such as usernames, passwords, and entire messages, may be intercepted and viewed, all without the knowledge of the email server or client. When using standard POP and IMAP protocols, all authentication information is sent "in the clear," meaning that someone on a network between the client and the remote server can easily view it.

16.6.1 Secure Email Clients

Thankfully, most Linux MUAs designed to check email on remote servers support SSL to encrypt messages as they are sent back and forth over the network. In order to use SSL when retrieving email, it must be enabled on the email client and server.

SSL is usually very easy to enable on the client-side, often done with the click of a button in the MUA's configuration area. Secure IMAP and POP have known port numbers (993 and 995, respectively) that the MUA will use to authenticate and download messages.

Popular MUAs included with Red Hat Linux, such as Mozilla Mail, Mutt, and Pine, offer SSL-encrypted email sessions.

16.6.2 Secure Email Servers

Offering SSL encryption to IMAP and POP users on the email server is almost as easy. Red Hat Linux also includes the `stunnel` package, which is an SSL encryption wrapper that wraps around standard, non-secure network traffic for certain services and prevents interceptors from being able to "sniff" the communication between client and server. While `stunnel` can be used with more than email communication, it really shines when providing protection for normally insecure email protocols.

The `stunnel` program uses external SSL libraries, such as the OpenSSL libraries included with Red Hat Linux, to provide strong cryptography and protect your connections. You can apply to a **Certificate Authority (CA)** for an SSL certificate, or you can create a self-signed certificate to simply provide the benefit of the SSL encrypted communication.

To create a self-signed SSL certificate, change to the `/usr/share/ssl/certs` directory, type the `make stunnel.pem` command, and answer the questions. Then, use `stunnel` to start the mail daemon that you wish to use.

For example, the following command could be used to start the IMAP server included with Red Hat Linux:

```
/usr/sbin/stunnel -d 993 -l /usr/sbin/imapd imapd
```

You should now be able to open an IMAP email client and connect to your email server using SSL encryption. Of course, you will probably want to go a step further and configure your `stunnel`-wrapped IMAP server to automatically start up at the correct runlevels.

For more information about how to use `stunnel`, read the `stunnel` man page or refer to the documents in the `/usr/share/doc/stunnel-<version-number>` directory.

Alternatively, the `imap` package bundled with Red Hat Linux contains the ability to provide SSL encryption on its own without `stunnel`. For secure IMAP connections, create the SSL certificate by

changing to the /usr/share/ssl/certs directory and running the make imapd.pem command. Then, set the imaps service to start at the proper runlevels and restart xinetd to enable the service.

You can also use the ipop3 package bundled with Red Hat Linux to provide SSL encryption on its own without stunnel.

16.7 Additional Resources

Many users initially find email programs difficult to configure, primarily due to the large number of options available. Access to additional documentation can be very helpful, especially when setting sensitive configuration options.

16.7.1 Installed Documentation

*   Information about how to configure Sendmail is included with the sendmail and sendmail-cf packages.

    –   /usr/share/doc/sendmail/README.cf — Contains information on m4, file locations for Sendmail, supported mailers, how to access enhanced features, and much more.

    –   /usr/share/doc/sendmail/README — Contains information on the Sendmail directory structure, IDENT protocol support, details on directory permissions, and the common problems these permissions can cause if misconfigured.

    In addition, the sendmail and aliases man pages contain helpful information covering various Sendmail options and the proper configuration of the Sendmail /etc/mail/aliases file, respectively.

*   /usr/share/doc/fetchmail-<version-number> — Contains a full list of Fetchmail features in the FEATURES file and an introductory FAQ document.

*   /usr/share/doc/procmail-<version-number> — Contains a README file that provides an overview of Procmail, a FEATURES file that explores every program feature, and an FAQ file with answers to many common configuration questions.

    When learning how Procmail works and creating new recipes, these Procmail man pages are invaluable:

    –   procmail — Provides an overview of how Procmail works and the steps involved with filtering email.

    –   procmailrc — Explains the rc file format used to construct recipes.

    –   procmailex — Gives a number of useful, real-world examples of Procmail recipes.

    –   `procmailsc` — Explains the weighted scoring technique used by Procmail to see if a particular recipe matches a certain message.

16.7.2 Useful Websites

- http://www.redhat.com/mirrors/LDP/HOWTO/Mail-Administrator-HOWTO.html — Provides an overview of how email works and examines possible email solutions and configurations on the client and server sides.

- http://www.redhat.com/mirrors/LDP/HOWTO/Mail-User-HOWTO — Looks at email from the user's perspective, investigates various popular email client applications, and gives an introduction to topics such as aliases, forwarding, auto-replying, mailing lists, mail filters, and spam.

- http://www.redhat.com/mirrors/LDP/HOWTO/mini/Secure-POP+SSH.html — Demonstrates a way to retrieve POP email using SSH with port forwarding, so that your email passwords and messages will be transferred securely.

- http://www.sendmail.net — Contains news, interviews, and articles concerning Sendmail, including an expanded view of the many options available.

- http://www.sendmail.org — Offers a thorough technical breakdown of Sendmail features and configuration examples.

- http://tuxedo.org/~esr/fetchmail — The home page for Fetchmail, featuring an online manual and thorough FAQ.

- http://www.procmail.org — The home page for Procmail, with links to assorted mailing lists dedicated to Procmail as well as various FAQ documents.

- http://www.ling.helsinki.fi/users/reriksso/procmail/mini-faq.html — An excellent Procmail FAQ, with troubleshooting tips and details about file locking and the use of wildcard characters.

- http://www.uwasa.fi/~ts/info/proctips.html — Points out dozens of tips that make using Procmail in various situations much easier, including how to test `.procmailrc` files and use Procmail scoring to decide if a particular action should be taken.

16.7.3 Related Books

- *Sendmail* by Bryan Costales with Eric Allman et al; O'Reilly & Associates — A good Sendmail reference written with the assistance of the original creator of Delivermail and Sendmail.

- *Removing the Spam: Email Processing and Filtering* by Geoff Mulligan; Addison-Wesley Publishing Company — A volume that looks at various methods used by email administrators that use established tools, such as Sendmail and Procmail, to manage spam problems.

- *Internet Email Protocols: A Developer's Guide* by Kevin Johnson; Addison-Wesley Publishing Company — Provides a very thorough review of major email protocols and the security they provide.

- *Managing IMAP* by Dianna Mullet and Kevin Mullet; O'Reilly & Associates — Details the steps required to configure an IMAP server.

17   Network File System (NFS)

NFS (Network File System) exists to allow remote hosts to mount partitions on a particular system and use them as though they were local filesystems. This allows files to be organized in a central location, while providing the functionality of allowing authorized users continuous access to them.

Two versions of NFS are currently in use. NFS version 2, which has been around for several years, is widely supported by various operating systems. NFS version 3 has several more features, including a variable file handle size and better error reporting. Red Hat Linux supports both NFSv2 and NFSv3, and uses NFSv3 by default when connecting with a server which supports it.

This chapter will focus on NFS version 2, though many of the concepts discussed also apply to version 3. Additionally, only fundamental NFS concepts and supplemental information will be provided. For specific instructions regarding the configuration and operation of NFS on client or server machines, see the *Official Red Hat Linux Customization Guide*.

17.1 Methodology

Linux uses a combination of kernel-level support and continuously running daemon processes to provide NFS file sharing, and NFS support must be enabled in the Linux kernel to function. NFS uses **Remote Procedure Calls (RPC)** to route requests between clients and servers, meaning that the `portmap` service must be enabled and active at the proper runlevels for NFS communication to occur. Working with `portmap`, various other processes ensure that a particular NFS connection is allowed and may proceed without error:

- `rpc.mountd` — The running process that receives the mount request from an NFS client and checks to see if it matches with a currently exported filesystem.

- `rpc.nfsd` — The process that implements the user-level part of the NFS service. It works with the Linux kernel to meet the dynamic demands of NFS clients, such as providing additional server threads for NFS clients to utilize.

- `rpc.lockd` — A daemon that is not necessary with modern kernels. NFS file locking is now done by the kernel. It is included with the `nfs-utils` package for users utilizing older kernels that do not include this functionality by default.

- `rpc.statd` — Implements the **Network Status Monitor (NSM)** RPC protocol. This provides reboot notification when an NFS server is restarted without being gracefully brought down.

- `rpc.rquotad` — An RPC server that provides user quota information for remote users.

Not all of these programs are required for NFS service. The only services that must be enabled are `rpc.mountd`, `rpc.nfsd`, and `portmap`. The other daemons provide additional functionality, based on the particular requirements of your server environment.

NFS version 2 uses the **User Datagram Protocol (UDP)** to provide a stateless network connection between the client and server. (NFS version 3 can use UDP or TCP running over an IP.) The stateless UDP connection minimizes network traffic, as the NFS server sends the client a cookie after the client is authorized to access the shared volume. This cookie, or random value that is stored on the server's side, is passed with any RPC requests from the client to the server. The NFS server can be restarted without affecting the clients and the cookie remains intact.

Using NFS, authentication only occurs when the client is attempting to mount to a remote filesystem. The NFS server uses the /etc/hosts.allow and /etc/hosts.deny files to determine if a particular host should be specifically permitted or prevented access via NFS. Then, the NFS server refers to the /etc/exports file to uncover that host's privileges for the various mounts available. After granting access, any file and directory operations are sent to the server using remote procedure calls.

> **WARNING**
>
> **NFS mount privileges are granted specifically to a host, not a user. If you grant a host access to a particular part of your hard drive with NFS, users of that machine will have access to your shared data.**
>
> **When configuring the `/etc/exports` file, be extremely careful when sharing directories with read-write permissions (`rw`) to a remote host. Users of remote systems mounting your export will be able to modify data in the exported filesystem.**

17.1.1 NFS and `portmap`

NFS relies upon remote procedure calls (RPC) to function. portmap is required to map RPC requests to the correct services. RPC processes notify portmap when they start, revealing the port number they are monitoring and the RPC program numbers they expect to serve. The client system then contacts portmap on the server with a particular RPC program number. portmap then redirects the client to the proper port number to communicate with its intended service.

Because RPC-based services rely on portmap to make all connections with incoming client requests, portmap must be available before any of these services start. If, for some reason, the portmap service unexpectedly quits, restart portmap and any services running when it was started.

The portmap service can be used with the host access files (/etc/hosts.allow and /etc/hosts.deny) to control which remote systems are permitted to use RPC-based services on your machine. See Chapter 9, *TCP Wrappers and xinetd* for more information. Access control rules for portmap will affect all RPC-based services. Alternatively, you can specify each of the NFS

RPC daemons to be affected by a particular access control rule. The man pages for `rpc.mountd` and `rpc.statd` contain information regarding the precise syntax of these rules.

`portmap` Status

As `portmap` provides the coordination between RPC services and the port numbers utilized to communicate with them, it is useful to be able to get a picture of the current RPC services using `portmap` when troubleshooting. The `rpcinfo` command shows each RPC-based service with its port number, RPC program number, version, and IP protocol type (TCP or UDP).

To make sure the proper NFS RPC-based services are enabled for `portmap`, `rpcinfo` can be useful:

```
[root@bleach /]# rpcinfo -p some.machine.com
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp   1024  status
    100024    1   tcp   1024  status
    100011    1   udp    819  rquotad
    100011    2   udp    819  rquotad
    100005    1   udp   1027  mountd
    100005    1   tcp   1106  mountd
    100005    2   udp   1027  mountd
    100005    2   tcp   1106  mountd
    100005    3   udp   1027  mountd
    100005    3   tcp   1106  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1028  nlockmgr
    100021    3   udp   1028  nlockmgr
    100021    4   udp   1028  nlockmgr
[root@bleach /]#
```

The `-p` option probes the portmapper on the specified host, or defaults to localhost if no specific host is listed. Other options are available from the `rpcinfo` man page.

From the output above, various NFS services can be seen running. If one of the NFS services does not start up correctly, `portmap` will be unable to map RPC requests from clients for that service to the correct port. In many cases, restarting NFS as root (`service nfs restart`) will cause those service to correctly register with `portmap` and begin working.

17.2 NFS Server Configuration Files

Configuring a system to share files and directories using NFS is straightforward. Every filesystem being exported to remote users via NFS, as well as the access rights relating to those filesystems, is located in the `/etc/exports` file. This file is read by the `exportfs` command to give

rpc.mountd and rpc.nfsd the information necessary to allow the remote mounting of a filesystem by an authorized host.

The exportfs command allows you to selectively export or unexport directories without restarting the various NFS services. When exportfs is passed the proper options, the filesystems to be exported are written to /var/lib/nfs/xtab. Since rpc.mountd refers to the xtab file when deciding access privileges to a filesystem, changes to the list of exported filesystems take effect immediately.

Various options are available when using exportfs:

- -r Causes all directories listed in /etc/exports to be exported by constructing a new export list in /etc/lib/nfs/xtab. This option effectively refreshes the export list with any changes that have been made to /etc/exports.

- -a Causes all directories to be exported or unexported, depending on the other options passed to exportfs.

- -o options Allows the user to specify directories to be exported that are not listed in /etc/exports. These additional filesystem shares must be written in the same way they are specified in /etc/exports. This option is used to test an exported filesystem before adding it permanently to the list of filesystems to be exported.

- -i Tells exportfs to ignore /etc/exports; only options given from the command line are used to define exported filesystems.

- -u Unexports directories from being mounted by remote users. The command exportfs -ua effectively suspends NFS file sharing while keeping the various NFS daemons up. To allow NFS sharing to continue, type exportfs -r.

- -v Verbose operation, where the filesystems being exported or unexported are displayed in greater detail when the exportfs command is executed.

If no options are passed to the exportfs command, it displays a list of currently exported filesystems.

Changes to /etc/exports can also be read by reloading the NFS service with the service nfs reload command. This keeps the NFS daemons running while re-exporting the /etc/exports file.

17.2.1 /etc/exports

The /etc/exports file is the standard for controlling which filesystems are exported to which hosts, as well as specifying particular options that control everything. Blank lines are ignored, comments can be made using #, and long lines can be wrapped with a backslash (\). Each exported filesystem should be on its own line. Lists of authorized hosts placed after an exported filesystem

must be separated by space characters. Options for each of the hosts must be placed in parentheses directly after the host identifier, without any spaces separating the host and the first parenthesis.

In its simplest form, `/etc/exports` only needs to know the directory to be exported and the hosts permitted to utilize it:

```
/some/directory bob.domain.com
/another/exported/directory 192.168.0.3
```

After re-exporting `/etc/exports` with the `/sbin/service nfs reload` command, the `bob.domain.com` host will be able to mount `/some/directory`, and `192.168.0.3` can mount `/another/exported/directory`. Because no options are specified in this example, several default NFS preferences take effect:

- `ro` Read-only. Hosts mounting this filesystem will not be able to change it. To allow hosts to make changes to the filesystem, you must specify `rw` (read-write).

- `async` Allows the server to write data to the disk when it sees fit. While this is not important if the host is accessing data as read-only, if a host is making changes to a read-write filesystem and the server crashes, data could be lost. By specifying the `sync` option, all file writes must be committed to the disk before the write request by the client is actually completed. This may lower performance.

- `wdelay` Causes the NFS server to delay writing to the disk if it suspects another write request is imminent. This can improve performance by reducing the number of times the disk must be accessed by separate write commands, reducing write overhead. Use `no_wdelay` to turn this feature off, which only works if you are using the `sync` option.

- `root_squash` Makes any client accesses to the exported filesystem, made as the root user on the client machine, take place as the nobody user ID. This effectively "squashes" the power of the remote root user to the lowest local user, preventing remote root users from acting as though they were the root user on the local system. Alternatively, the `no_root_squash` option turns off root squashing. To squash every remote user, including root, use the `all_squash` option. To specify the user and group IDs to use with remote users from a particular host, use the `anonuid` and `anongid` options, respectively. In this way, you can create a special user account for remote NFS users to share and specify (`anonuid=<uid-value>,anongid=<gid-value>`), where `<uid-value>` is the user ID number and `<gid-value>` is the group ID number.

In order to override these defaults, you must specify an option that takes its place. For example, if you do not specify `rw`, then that export will only be shared read-only. Each default for every exported filesystem must be explicitly overridden. Additionally, other options are available where no default value is in place. These include the ability to disable sub tree checking, allow access from insecure ports, and allow insecure file locks (necessary for certain early NFS client implementations). See the `exports` man page for details on these lesser used options.

When specifying hosts to be allowed to use a particular exported filesystem, a variety of methods can be used, including:

- *single host* — Where one particular host is specified with a fully qualified domain name, host-name, or IP address.

- *wildcards* — Where a `*` or `?` character is used to take into account a grouping of fully qualified domain names or IP addresses or those that match a particular string of letters.

  However, be careful when using wildcards with fully qualified domain names, as they tend to be more exact than you would expect. For example, the use of `*.domain.com` as wildcard will allow sales.domain.com to access the exported filesystem, but not bob.sales.domain.com. To match both possibilities, as well as sam.corp.domain.com, you would have to provide `*.domain.com *.*.domain.com`.

- *IP networks* — Allows the matching of hosts based on their IP addresses within a larger network. For example, `192.168.0.0/28` will allow the first 16 IP addresses, from 192.168.0.0 to 192.168.0.15, to access the exported filesystem but not 192.168.0.16 and higher.

- *netgroups* — Permits an NIS netgroup name, written as `@<group-name>`, to be used. This effectively puts the NIS server in charge of access control for this exported filesystem, where users can be added and removed from an NIS group without affecting `/etc/exports`.

> **CAUTION**
>
> The way in which the `/etc/exports` file is formatted is very important, particularly concerning the use of space characters. Remember to always separate exported filesystems from hosts and hosts from one another with a space character. However, there should be no other space characters in the file unless they are used in comment lines.
>
> For example, the following two lines do not mean the same thing:
>
> ```
> /home bob.domain.com(rw)
> /home bob.domain.com (rw)
> ```
>
> The first line allows only users from `bob.domain.com` read-write access to the /home directory. The second line allows users from `bob.domain.com` to mount the directory read-only (the default), but the rest of the world can mount it read-write. Be careful where space characters are used in `/etc/exports`.

17.3 NFS Client Configuration Files

Any NFS share made available by a server can be mounted using various methods. Of course, the share can be manually mounted, using the `mount` command, to acquire the exported filesystem at a particular mount point. However, this requires that the root user type the `mount` command every time the system restarts. In addition, the root user must remember to unmount the filesystem when shutting down the machine. Two methods of configuring NFS mounts include modifying the `/etc/fstab` or utilizing the `autofs` service.

17.3.1 `/etc/fstab`

Placing a properly formatted line in the `/etc/fstab` file has the same effect as manually mounting the exported filesystem. The `/etc/fstab` file is read by the `/etc/rc.d/init.d/netfs` script at system startup. The proper filesystem mounts, including NFS, are put into place.

A sample `/etc/fstab` line to mount an NFS export looks like the following:

```
<server-host>:</path/to/shared/directory> </local/mount/point>
nfs <options> 0 0
```

The `<server-host>` relates to the hostname, IP address, or fully qualified domain name of the server exporting the filesystem. The `</path/to/shared/directory>` tells the server what export to mount. The `</local/mount/point>` specifies where on the local filesystem to mount the exported directory. This mount point must exist before `/etc/fstab` is read or the mount will fail. The `nfs` option specifies the type of filesystem being mounted.

The `<options>` area specifies how the filesystem is to be mounted. For example, if the options area states `rw,suid` on a particular mount, the exported filesystem will be mounted read-write and the user and group ID set by the server will be used. Note, parentheses are not to be used here. For more mount options, see Section 17.3.3, *Common NFS Mount Options*.

17.3.2 `autofs`

One drawback to using `/etc/fstab` is that, regardless of how much you use that mounted filesystem, your system must dedicate resources to keep that mount in place. This is not a problem with one or two mounts, but when your system is maintaining mounts to a dozen systems at one time, overall system performance can suffer. An alternative to `/etc/fstab` is to use the kernel-based `auto-mount` utility, which will mount and unmount NFS filesystems automatically, saving resources.

The `autofs` script, located in `/etc/rc.d/init.d`, is used to control `automount` through the `/etc/auto.master` primary configuration file. While `automount` can be specified on the command line, it is more convenient to specify the mount points, hostname, exported directory, and options in a set of files rather than typing them all by hand. By running `autofs` as a service that starts and

stops in designated runlevels, the mount configurations in the various files can be automatically implemented. In order to use `autofs`, you must have the `autofs` RPM installed on your system.

The `autofs` configuration files are arranged in a parent-child relationship. A main configuration file (`/etc/auto.master`) refers mount points on your system that are linked to a particular **map type**, which take the form of other configuration files, programs, NIS maps, and other less common mount methods. The `auto.master` file contains lines referring to each of these mount points, organized like this:

```
<mount-point>     <map-type>
```

The `<mount-point>` indicates where the device or exported filesystem should mount on your local filesystem. The `<map-type>` relates to the way in which the mount point will be mounted. The most common method for auto mounting NFS exports is to use a file as the map type for the particular mount point. The map file, usually named `auto.<mount-point>`, where `<mount-point>` is the mount point designated in `auto.master`, contains lines that look like this:

```
<directory>   <mount-options>   <host>:<exported-filesystem>
```

The `<directory>` refers to the directory within the mount point where the exported filesystem should be mounted. Much like a standard `mount` command, the host exporting the filesystem, as well as the fil esystem being exported, are required in the `<host>:<exported-filesystem>` section. To specify particular options to be used when mounting the exported filesystem, place them in the `<mount-options>` section, separated by commas. For NFS mounts that use `autofs`, you should definitely place `-fstype=nfs` in the `<mount-options>` section, at a minimum.

While `autofs` configuration files can be used for a variety of mounts to many types of devices and filesystems, they are particularly useful in creating NFS mounts. For example, some organizations store a user's `/home` directory on a central server via an NFS share. Then, they configure the `auto.master` file on each of the workstations to point to an `auto.home` file containing the specifics for how to mount the `/home` directory via NFS. This allows the user to access personal data and configuration files in their `/home` directory by logging in anywhere on the internal network. The `auto.master` file in this situation would look similar to this:

```
/home   /etc/auto.home
```

This sets up the `/home` mount point on the local system to be configured by the `/etc/auto.home` file, which may look similar to this:

```
*   -fstype=nfs,soft,intr,rsize=8192,wsize=8192,nosuid  server.domain.com:/home/&
```

This line states that any directory a user tries to access under the local `/home` directory (due to the asterisk character) should result in an NFS mount on the server.domain.com system within its exported `/home` filesystem. The mount options specify that each `/home` directory NFS mounts should use a particular collection of settings. For more information on mount options, including the ones used in this example, see Section 17.3.3, *Common NFS Mount Options*.

17.3.3 Common NFS Mount Options

Beyond mounting a filesystem via NFS on a remote host, a number of different options may be specified at the time of the mount that can make it easier to use. These options can be utilized with manual `mount` commands, `/etc/fstab` settings, and `autofs`, and other mounting methods.

The following options are the most popular for NFS mounts:

- `hard` or `soft` specifies whether the program using a file via an NFS connection should stop and wait (`hard`) for the server to come back online if the host serving the exported filesystem is unavailable, or if it should report an error (`soft`).

  If you specify `hard`, you will not be able to terminate the process waiting for the NFS communication to resume unless you also specify the `intr` option.

  If you specify `soft`, you can set an additional `timeo=<value>` option, where `<value>` specifies the number of seconds to pass before the error is reported.

- `intr` allows NFS requests to be interrupted if the server goes down or cannot be reached.

- `nolock` is occasionally required when connecting to older NFS server. To require locking, use the `lock` option.

- `noexec` does not permit the execution of binaries on the mounted filesystem. This is useful if your Red Hat Linux system is mounting a non-Linux filesystem via NFS that contains binaries that will not execute on your machine.

- `nosuid` does not allow set-user-identifier or set-group-identifier bits to take effect.

- `rsize=8192` and `wsize=8192` may speed up NFS communication for reads (`rsize`) and writes (`wsize`) by setting a larger data block size, in bytes, to be transferred at one time. Be careful when changing these values; some older Linux kernels and network cards may not work well with larger block sizes.

- `nfsvers=2` or `nfsvers=3` specify which version of the NFS protocol to use.

Many more options are available on the `mount` man page, including options to be used when mounting non-NFS filesystems.

17.4 Securing NFS

NFS works well for sharing entire filesystems with a large number of known hosts in a largely transparent manner. Many users accessing files over an NFS mount may not be aware that the filesystem they are using is not local to their system. However, with ease of use comes a variety of potential security problems.

The following points should be considered when exporting NFS filesystems on a server or mounting them on a client. Doing so will minimize NFS security risks and better protect your data and equipment.

17.4.1 Host Access

NFS controls who can mount an exported filesystem based on the host making the mount request, not the user that will utilize the filesystem. Hosts must be given explicit rights to mount the exported filesystem. Access control is not possible for users, other than file and directory permissions. In other words, when you export a filesystem via NFS to a remote host, you are not only trusting the host you are allowing to mount the filesystem. You are also allowing any user with access to that host to use your filesystem as well. The risks of doing this can be controlled, such as requiring read-only mounts and squashing users to a common user and group ID, but these solutions may prevent the mount from being used in the way originally intended.

Additionally, if an attacker gains control of the DNS server used by the system exporting the NFS filesystem, the system associated with a particular hostname or fully qualified domain name can be pointed to an unauthorized machine. At this point, the unauthorized machine *is* the system permitted to mount the NFS share, since no username or password information is exchanged to provide additional security for the NFS mount. The same risks hold true to compromised NIS servers, if NIS netgroups are used to allow certain hosts to mount an NFS share. By using IP addresses in /etc/exports, this kind of attack is more difficult.

Wildcards should be used sparingly when granting host access to an NFS share. The scope of the wildcard may encompass systems that you may not know exist and should not be allowed to mount the filesystem.

17.4.2 File Permissions

Once the NFS filesystem is mounted read-write by a remote host, protection for each shared file involves its permissions, and its user and group ID ownership. If two users that share the same user ID value mount the same NFS filesystem, they will be able to modify each others files. Additionally, anyone logged in as root on the client system can use the su command to become a user who could access particular files via the NFS share.

The default behavior when exporting a filesystem via NFS is to use **root squashing**. This sets the user ID of anyone utilizing the NFS share as the root user on their local machine to a value of the server's nobody account. You should never turn off root squashing unless multiple users with root access to your server does not bother you.

If you are only allowing users to read files via your NFS share, consider using the all_squash option, which makes every user accessing your exported filesystem to take the user ID of the nobody user.

17.5 Additional Resources

Administering an NFS server can be a challenge. Many options, including quite a few not mentioned in this chapter, are available for exporting NFS filesystems or mounting them as a client. Consult these sources of information for more details.

17.5.1 Installed Documentation

- `/usr/share/doc/nfs-utils-<version-number>` — Covers the way NFS is implemented in Linux, including a look at various NFS configurations and their impact on file transfer performance.

- The following man pages are very helpful:

  - `mount` — Contains a comprehensive look at mount options for both NFS server and client configurations.

  - `fstab` — Gives details for the format of the `/etc/fstab` file used to mount filesystems at system boot.

  - `nfs` — Provides detail on NFS-specific filesystem export and mount options.

  - `exports` — Shows common options used in the `/etc/exports` file when exporting NFS filesystems.

17.5.2 Related Books

- *Managing NFS and NIS* by Hal Stern, Mike Eisler, and Ricardo Labiaga; O'Reilly & Associates — Makes an excellent reference guide for the many different NFS export and mount options available.

- *NFS Illustrated* by Brent Callaghan; Addison-Wesley Publishing Company — Provides comparisons of NFS to other network filesystems and shows, in detail, how NFS communication occurs.

18   Firewalling with `iptables`

The Linux kernel contains advanced tools for **packet filtering**, the process of controlling network packets as they attempt to enter, move through, and exit your system. Pre-2.4 kernels contained the ability to manipulate packets using `ipchains` which used lists of rules that apply to packets at each step of the filtering process. The introduction of the 2.4 kernel brought with it `iptables`, which is similar to `ipchains` but greatly expands on the scope and control available when filtering packets.

This chapter focuses on essential packet filtering basics, defining the differences between `ipchains` and `iptables`, explaining various options available with `iptables` commands, and showing how filtering rules can be preserved between system reboots.

If you require instructions for constructing `iptables` rules or setting up a firewall based on these rules, please see the *Official Red Hat Linux Customization Guide*.

18.1 Packet Filtering

Traffic moves through a network in **packets**, which are collections of data in particular sizes. A file sent over a network between two computers may be comprised of many packets, each of which holds a small part of the file data. The sending computer takes the file and breaks it into packets to be sent over the network, using the rules of the particular network protocol being utilized. The other computer receives the packets and, using the method specified by the protocol, reassembles the packets into the file.

Every packet contains information which helps it navigate the network and move to its destination. The packet can tell computers along the way, as well as the destination machine, where it came from, where it is going, and what type of packet it is, among other things. Most packets are designed to carry data, although some protocols use packets in special ways. The **Transmission Control Protocol (TCP)**, for example, uses a SYN packet, which contains no data, to initiate a communication between two systems.

The Linux kernel contains the built-in ability to filter packets, allowing some of them into the system while stopping others in their tracks. The 2.4 kernel contains three tables of chains; here we describe the filter table. The filter table contains three sets of rule lists by default, called the INPUT, OUTPUT, and FORWARD chains; every packet being sent in or out of the machine is subject to one of these lists of rules. When a packet enters the system via a network interface, the kernel decides if it is destined for the local system (INPUT chain) or another destination (FORWARD chain) to determine the rule list to use with it. In the same way, if a packet originates on the system and attempts to leave the system, the kernel will check it against the OUTPUT chain.

Each packet may need be checked against many, many rules before emerging at the end of the chain. The structure and purpose of these rules may vary, but they usually seek to identify a packet coming

from or going to a particular IP address or set of addresses when using a particular protocol and network service.

Regardless of their destination, when packets match a particular rule on one of the rule lists, they are designated for a particular **target**, or action to be applied to them. If the rule specifies an ACCEPT target for a matching packet, the packet skips the rest of the rule checks and is allowed to continue to its destination. If a rule specifies a DROP target, the packet is "dropped," meaning that the packet is refused access to the system and nothing is sent back to the host that sent the packet. If a rule specifies a REJECT target, the packet is dropped, but an error packet is sent to the packet's originator.

Every chain has a default policy to ACCEPT, DROP, or REJECT the packet, or possibly, QUEUE the packet to be passed to userspace. If none of the rules in the chain apply to the packet, then the packet is dealt with in accordance with the chain's defualt policy.

The `iptables` command allows you to configure these rule lists, as well as set up new chains and tables to be used for your particular situation.

18.2 Differences between `iptables` and `ipchains`

At first glance, `ipchains` and `iptables` appear to be quite similar. After all, both methods of packet filtering use chains of rules operating within the Linux kernel to decide not only which packets to let in or out, but also what to do with packets that match certain rules, however, `iptables` offer a much more extensible way of filtering packets, giving an administrator a finer grained level of control without building too much complexity into the entire system.

Specifically, users comfortable with `ipchains` should be aware of the following significant differences between `ipchains` and `iptables` before attempting to use `iptables`:

• *Under `iptables`, each filtered packet is only processed using rules from one chain rather than multiple chains.* In other words, a FORWARD packet coming into a system using `ipchains` would have to go through the INPUT, FORWARD, and OUTPUT chains in order to move along to its destination. However, `iptables` only sends packets to the INPUT chain if they are destined for the local system and only sends them to the OUTPUT chain if the local system generated the packets. For this reason, you must be sure to place the rule designed to catch a particular packet in the correct chain that will actually see the packet.

    The advantage is that you now have finer-grained control over the disposition of each packet. If you are attempting to block access to a particular website, it is now possible to block access attempts from clients running on hosts which use your host as a gateway. An OUTPUT rule which denies access will no longer prevent access for hosts which use your host as a gateway.

• *The DENY target has been changed to DROP.* In `ipchains`, packets that matched a rule in a chain could be directed to the DENY target, which silently dropped the packet. This target must be changed to DROP in `iptables` to have the same effect.

- *Order matters when placing options in a chain rule.* Previously, with `ipchains`, it did not matter very much how you ordered the rule options when typing the rule. The `iptables` command is a bit pickier about where some options may go. For example, you must now specify the source or destination port after the protocol (ICMP, TCP, or UDP) to be used in a chain's rule.

- *When specifying network interfaces to be used with a rule, you must only use incoming interfaces (`-i` option) with INPUT or FORWARD chains and outgoing interfaces (`-o` option) with FOR- WARD or OUTPUT chains.* This is necessary due to the fact that OUTPUT chains are no longer used by incoming interfaces, and INPUT chains are not seen by packets moving through outgoing interfaces.

This is by no means a comprehensive list of the changes, given that `iptables` represents a funda- mentally rewritten network filter in use with the kernel. For more specific information, consult the *Linux 2.4 Packet Filtering HOWTO* and the sources found in Section 18.5, *Additional Resources*.

18.3 Options Used in `iptables` Commands

Rules that allow packets to be filtered by the kernel are put into place by running the `iptables` command with a number of options after it that identify the types of packets being filtered, the source or destination of those packets, and what to do with the packet if it matches the rule. The options used with a particular `iptables` rule must be grouped logically, based on the purpose and conditions of the overall rule, in order for the rule to be valid.

18.3.1 Tables

A powerful aspect of `iptables` is that multiple tables can be used to decide the fate of a particular packet, depending upon the type of packet being monitored and what is to be done with the packet. Thanks to the extensible nature of `iptables`, specialized tables can be created and stored in the `/etc/modules/<kernel-version>/kernel/net/ipv4/netfilter` directory to meet special goals. Think of `iptables` as being able to run multiple sets of `ipchains` rules in defined chains, with each set fulfilling a particular role.

The default table, named `filter`, contains the standard built-in INPUT, OUTPUT, and FORWARD chains. This is somewhat similar to the standard chains in use with `ipchains`. However, by default, `iptables` also includes two additional tables that perform specific packet filtering jobs. The `nat` table can be used to modify the source and destination addresses recorded in packets, and the `mangle` table allows you to alter packets in specialized ways.

Each table contains default chains that perform necessary tasks based on the purpose of the table, but you can easily set up new chains in each of the tables.

18.3.2 Structure

Many `iptables` commands take the following structure:

```
iptables [-t <table-name>] <command> <chain-name>
<parameter-1> <option-1>
     <parameter-n> <option-n>
```

In this example, the `<table-name>` option allows the user to select a table other than the default `filter` table to use with the command. The `<command>` option is the center of the command, dictating a specific action to perform, such as appending or deleting a rule from a particular chain, which is specified by the `<chain-name>` option. Following the `<chain-name>` are pairs of parameters and options that actually define the way the rule will work and what will happen when a packet matches the rule.

When looking at the structure of an `iptables` command, it is important to remember that, unlike most other commands, the length and complexity of an `iptables` command can change based on its purpose. A simple command to remove a rule from a chain can be very short, while a command designed to filter packets from a particular subnet using a variety of specific parameters and options can be rather lengthy. One way to think about `iptables` commands is to recognize that some parameters and options used may create the need to use other parameters and options to further specify the previous option's request. In order to construct a valid rule, this must continue until every parameter and option that requires another set of options is satisfied.

Type `iptables -h` to see a comprehensive list of `iptables` command structures.

18.3.3 Commands

Commands tell `iptables` to perform a specific action, and only one command is allowed per `iptables` command string. Except for the help command, all commands are written in upper-case characters.

The `iptables` commands are:

*   `-A` — Appends the `iptables` rule to the end of the specified chain. This is the command used to simply add a rule when rule order in the chain does not matter.

*   `-C` — Checks a particular rule before adding it to the user-specified chain. This command can help you construct complicated `iptables` rules by prompting you for additional parameters and options.

*   `-D` — Deletes a rule in a particular chain by number (such as 5 for the fifth rule in a chain). You can also type the entire rule, and `iptables` will delete the rule in the chain that matches it.

*   `-E` — Renames a user-defined chain. This does not affect the structure of the table at all. Rather, it just saves you the trouble of deleting the chain, creating it under the new name, and reconfiguring all of your rules for that chain.

*   `-F` — Flushes the selected chain, which effectively deletes every rule in the the chain. If no chain is specified, this command flushes every rule from every chain.

- `-h` — Provides a list of helpful command structures, as well as a quick summary of command parameters and options.

- `-I` — Inserts a rule in a chain at a particular point. Assign a number to the rule to be inserted and `iptables` will put it there. If no number is specified, `iptables` will place your command at the top of the rule list.

CAUTION

Be aware of which option (`-A` or `-I`) you are using when adding a rule. The order of the rules can be very important when determining if a particular packet applies to one rule or another. Make sure when adding a rule to the beginning or end of the chain that it does not affect other rules in that chain.

- `-L` — Lists all of the rules in the chain specified after the command. To list all rules in all chains in the default `filter` table, do not specify a chain or table. Otherwise, the following syntax should be used to list the rules in a specific chain in a particular table:

      iptables -L <chain-name> -t <table-name>

  Powerful options for the `-L` command that provide rule numbers and allow more verbose rule descriptions, among others, are described in Section 18.3.7, *Listing Options*.

- `-N` — Creates a new chain with a user-specified name.

- `-P` — Sets the default policy for a particular chain, so that when packets traverse an entire chain without matching a rule, they will be sent on to a particular target, such as ACCEPT or DROP.

- `-R` — Replaces a rule in a particular chain. You must use a rule's number after the chain's name to replace that rule. The first rule in a chain relates to rule number `1`.

- `-X` — Deletes a user-specified chain. Deleting a built-in chain for any table is not allowed.

- `-Z` — Zeros the byte and packet counters in all chains for a particular table.

18.3.4 Parameters

Once certain `iptables` commands are specified, including those used to add, append, delete, insert, or replace rules within a particular chain, parameters are required to begin the construction of the packet filtering rule.

- `-c` Resets the counters for a particular rule. This parameter accepts the PKTS and BYTES options to specify what counter to reset.

- −d Sets the destination hostname, IP address, or network of a packet that will match the rule. When matching a network, you can use two different methods for signifying the netmasks, such as `192.168.0.0/255.255.255.0` or `192.168.0.0/24`.

- −f Applies this rule only to fragmented packets.

  By using the `!` option after this parameter, only unfragmented packets will be matched.

- −i Sets the incoming network interface, such as `eth0` or `ppp0`, to use with a particular rule. With `iptables`, this optional parameter may only be used with the INPUT and FORWARD chains when used with the `filter` table and the PREROUTING chain with the `nat` and `mangle` tables.

  This parameter features several useful options that may be used before specifying the name of an interface:

  – `!` — Tells this parameter not to match, meaning that any specified interfaces are specifically excluded from this rule.

  – `+` — A wildcard character used to match all interfaces that match a particular string. For example, the parameter `-i eth+` would apply this rule to any Ethernet interfaces on your system but exclude any other interfaces, such as `ppp0`.

  If the `-i` parameter is used but no interface is specified, then every interface is affected by the rule.

- −j Tells `iptables` to jump to a particular target when a packet matches a particular rule. Valid targets to be used after the `-j` option include the standard options, ACCEPT, DROP, QUEUE, and RETURN, as well as extended options that are available through modules loaded by default with the Red Hat Linux `iptables` RPM package, such as LOG, MARK, and REJECT, among others. See the `iptables` man page for more information on these and other targets, including rules regarding their use, as many targets may only be used with a particular table.

  Other than specifying target action, you may also direct a packet matching this rule to a user-defined chain outside of the current chain. This allows you to apply other rules against this packet, further filtering it with more specific criteria.

  If no target is specified, the packet moves past the rule with no action taken. However, the counter for this rule is still increased by 1, as the packet matched the specified rule.

- −o Sets the outgoing network interface for a particular rule, and may only be used with OUTPUT and FORWARD chains in the `filter` table and the POSTROUTING chain in the `nat` and `mangle` tables. This parameter's options are the same as those of the incoming network interface parameter (`-i`).

- −p Sets the IP protocol for the rule, which can be either `icmp`, `tcp`, `udp`, or `all`, to match every possible protocol. In addition, lesser used protocols listed in `/etc/protocols` can also be utilized. If this option is omitted when creating a rule, the `all` option is the default.

- −s Sets the source for a particular packet, using the same syntax as the destination (`-d`) parameter.

18.3.5 Match Options

Different network protocols provide specialized matching options which may be set in specific ways to match a particular packet using that protocol. Of course, the protocol must first be specified in the `iptables` command, such as using `-p tcp <protocol-name>`, to make the options for that protocol available.

TCP Protocol

These match options are available for the TCP protocol (`-p tcp`):

*   `--dport` Sets the destination port for the packet. You can use either a network service name (such as www or smtp), port number, or range of port numbers to configure this option. To browse the names and aliases of network services and the port numbers they use, view the `/etc/services` file. You can also use `--destination-port` to specify this match option.

    To specify a specific range of port numbers, separate the two numbers with a colon (`:`), such as `-p tcp --dport 3000:3200`. The largest valid range is `0:65535`.

    You may also use an exclamation point character (`!`) as a flag after the `--dport` option to tell `iptables` to match all packets which do not use that network service or port.

*   `--sport` Sets the source port of the packet, using the same options as `--dport`. You can also use `--source-port` to specify this match option.

*   `--syn` Causes all TCP packets designed to initiate communication, commonly called **SYN packets**, to match with this rule. Any packets that carry a data payload will not be touched. Placing an exclamation point character (`!`) as a flag after the `--syn` option causes all non-SYN packets to be matched.

*   `--tcp-flags` Allows TCP packets with specific bits, or flags, set to be matched with a rule. The `--tcp-flags` match option accepts two parameters after it, which are flags for the various bits arranged in a comma-separated list. The first parameter is the mask, which sets the flags to be examined on the packet. The second parameter refers to the flags that must be set in the packet to make a match. The possible flags are `ACK`, `FIN`, `PSH`, `RST`, `SYN`, and `URG`. In addition, `ALL` and `NONE` can also be used to match every flag or none of them.

    For example, an `iptables` rule which contains `-p tcp --tcp-flags ACK,FIN,SYN SYN` will only match TCP packets that have the SYN flag set and the ACK and FIN flags unset.

    Like many other options, using the exclamation point character (`!`) after `--tcp-flags` reverses the effect of the match option, so that the second parameter's flags must not be set in order to match.

- `--tcp-option` Attempts to match with TCP-specific options that can be set within a particular packet. This match option can also be reversed with the exclamation point character (`!`).

UDP Protocol

These match options are available for the UDP protocol (`-p udp`):

- `--dport` Specifies the destination port of the UDP packet, using the service name, port number, or range of port numbers. The `--destination-port` match option may be used instead of `--dport`. See the `--dport` match option in *TCP Protocol* in Section 18.3.5 for various ways to use this option.

- `--sport` Specifies the source port of the UDP packet, using the service name, port number, or range of port numbers. The `--source-port` match option may be used instead of `--sport`. See the `--dport` match option in *TCP Protocol* in Section 18.3.5 for various ways to use this option.

ICMP Protocol

Packets using the Internet Control Message Protocol (ICMP) can be matched using the following option when `-p icmp` is specified:

- `--icmp-type` Sets the name or number of the ICMP type to match with the rule. A list of valid ICMP names can be seen by typing the `iptables -p icmp -h` command.

Modules with Additional Match Options

Additional match options, which are not specific to a particular protocol, are also available through modules which load when the `iptables` command utilizes them. To utilize a match option module, you must load the module by name by including `-m <module-name>` in the `iptables` command creating a rule.

A large number of modules, each with their own specific match options, are available by default. It is even possible to create your own modules to provide additional match option functionality, perhaps for specific network requirements. Many modules exist, but only the most popular ones are discussed here.

The `limit` module allows you to place a limit on how many packets will be matched to a particular rule. This is especially beneficial when logging rule matches so that a flood of matching packets will not fill up your logs with repetitive messages or use too many system resources.

- `--limit` — Sets the number of matches for a particular range of time, specified with a number and time modifier arranged in a `<number>/<time>` format. For example, using `--limit 5/hour` only lets a rule match five times in a single hour.

  If a number and time modifier are not used, the default value of `3/hour` is assumed.

- `--limit-burst` — Sets a limit on the number of packets able to match a rule at one time. This option should be used in conjunction with the `--limit` option, and it accepts a number to set the burst threshold.

  If no number is specified, only five packets are initially able to match the rule.

The `state` module, which uses the `--state` match option, can match a packet with these particular connection states:

- `ESTABLISHED` The matching packet is associated with other packets in an established connection.

- `INVALID` The matching packet cannot be tied to a known connection.

- `NEW` The matching packet is either creating a new connection or is part of a two-way connection not previously seen.

- `RELATED` The matching packet is starting a new connection related in some way to an existing connection.

These connection states can be used in combination with one another by separating them with commas, such as `-m state --state INVALID,NEW`.

To specifically match a particular hardware MAC address of an Ethernet device, use the `mac` module, which accepts `--mac-source` plus a MAC address as an option. To exclude a MAC address from a rule, place an exclamation point (`!`) after the `--mac-source` match option.

To view other match options available through modules, see the `iptables` man page.

18.3.6 Target Options

Once a packet has matched a particular rule, the rule can direct the packet to a number of different targets that decide its fate and, possibly, take additional actions, such as logging the action. Additionally, each chain has a default target, which is used if none of the rules on that chain match a packet, or if none of the rules which match the packet specify a target.

There are only a few standard targets available to decide what happens with the packet:

- `<user-defined-chain>` The name of a previously created and defined chain within this table with rules that will be checked against this packet, in addition to any other rules in any other chains that must be checked against this packet. This type of target is useful for further scrutinizing a packet before deciding what should happen to it or logging information about the packet.

- `ACCEPT` — Allows the packet to successfully move on to its destination (or another chain, if one is configured to follow the successful chain).

- `DROP` — Drops the packet on the floor. The system that sent the packet is not notified of the failure. The packet is simply removed from the rule checking the chain and discarded.

- QUEUE — The packet is queued for handling in userspace, where something else (a user or an application, for instance) can do something with it.

- RETURN — Stops checking the packet against rules in the current chain. If the packet with a RETURN target matches a rule in a chain called from another chain, the packet is returned to the first chain to resume rule checking where it left off. If the RETURN rule is used on a built-in chain and the packet cannot move up to its previous chain, the default target for the current chain decides what is done with it.

In addition to these standard targets, various other targets may be used with extensions called **target modules**, which work in a similar manner to match option modules. For more information about match option modules, see *Modules with Additional Match Options* in Section 18.3.5.

There are many extended target modules, most of which only apply to specific tables or situations. A couple of the most popular target modules included by default in Red Hat Linux are:

- LOG Logs all packets that match this rule. Since the packets are logged by the kernel, the `/etc/syslog.conf` file determines where these log entries are written. By default, they are placed in the `/var/log/messages` file.

    Various options can be used after the LOG target to specify the way in which logging occurs:

    - `--log-level` Sets the priority level a of logging event. A list of priority levels can be found in the `syslog.conf` man page, and their names can be used as options after the `--log-level` option.

    - `--log-ip-options` Any options set in the header of a IP packet is logged.

    - `--log-prefix` Places a string of text before the log line when it is written. Accepts up to 29 characters after the `--log-prefix` option. This is also useful for writing syslog filters for use in conjunction with packet logging.

    - `--log-tcp-options` Any options set in the header of a TCP packet is logged

    - `--log-tcp-sequence` Writes the TCP sequence number for the packet in the log.

- REJECT Sends an error packet back to the system which sent the packet, and then DROPs the packet. This target is useful if you would like to notify the system sending the matching packet of the problem.

    The REJECT target accepts a `--reject-with <type>` option to provide more detail to be sent with the error packet. The message `port-unreachable` is the default `<type>` error given if no other option is used. For a full list of `<type>` options that can be used, see the `iptables` man page.

Other target extensions, including several that are useful with masquerading using the `nat` table, can be found in the `iptables` man page.

18.3.7 Listing Options

The default list command, `iptables -L`, provides a very basic overview of the defualt filter table's current rile chains. Additional options exist that provide more information and arrange that information in specific ways:

- `-v` Display verbose output, such as the number of packets and bytes each chain has seen, the number of packets and bytes each rule has matched, and which interfaces apply to a particular rule.

- `-x` Expands numbers into their exact values. On a busy system, the number of packets and bytes seen by a particular chain or rule may be abbreviated using K (thousands), M (millions), and G (billions) at the end of the number. This option forces the full number to be displayed.

- `-n` Displays IP addresses and port numbers in numeric format, rather than the default hostname and network service format.

- `--line-numbers` Lists rules in each chain next to their numeric order in the chain. This option is useful when attempting to delete a specific rule in a chain, or to locate where to insert a rule within a chain.

18.4 Storing `iptables` Information

Rules created with the `iptables` command are only stored in RAM. If you were to restart your system after setting up various `iptables` rules, they would be lost and you would need to retype them. If you want particular rules to take effect whenever your system boots, you need to save them to the `/etc/sysconfig/iptables` file.

To do this, arrange your tables, chains, and rules the way they should be the next time the system boots or `iptables` is restarted, and type the `/sbin/service iptables save` command as the root user. This causes the `iptables` init script to run the `/sbin/iptables-save` program and write the current `iptables` configuration to the `/etc/sysconfig/iptables` file. This file should only be readable by root, so your precise packet filtering rules are not viewable by average users.

The next time the system boots, the `iptables` init script will reapply the rules saved in `/etc/sysconfig/iptables` by using the `/sbin/iptables-restore` command.

While it is always a good idea to test a new `iptables` rule before committing it to the `/etc/sysconfig/iptables` file, it is possible to copy `iptables` rules into this file from another system's version of this file. This allows you to quickly distribute sets of `iptables` rules to many different machines at once. Simply restart `iptables` to make the new rules take effect.

18.5 Additional Resources

Packet filtering and `iptables` are complex subjects. Additional information can be helpful in providing alternative viewpoints and methods for controlling network traffic on your system.

18.5.1 Installed Documentation

• The `iptables` man page contains a comprehensive description of various commands, parameters, and other options that assist in the addition of new tables and construction of chain rules.

18.5.2 Useful Websites

• http://netfilter.samba.org — Contains assorted information about `iptables`, including an FAQ addressing specific problems you may see and various helpful guides by Rusty Russell, the Linux IP firewall maintainer. The HOWTO documents here cover subjects such as basic networking concepts, 2.4 kernel packet filtering and NAT configurations, and Netfilter.

• http://www.linuxnewbie.org/nhf/intel/security/iptables_basics.html —A very basic and general look at the way packets move through the Linux kernel, plus an introduction to constructing simple `iptables` commands.

• http://securityportal.com/cover/coverstory20010122.html — An introductory look at new packet handling features in the 2.4 kernel, including stateful firewalling, defending against Denial of Service attacks and stealth scans, filtering by MAC address, and enhanced packet logging.

Part IV Appendixes

A General Parameters and Modules

This appendix is provided to illustrate *some* of the possible parameters that may be needed by certain drivers[1] for particular hardware devices. In most cases, these additional parameters are unnecessary, since the kernel may already be able to use the device without them. You should only use the settings provided in this appendix if you are having trouble getting Red Hat Linux to use a particular device or you need to override the system's default parameters for the device.

During the installation of Red Hat Linux, some limits are placed on filesystems and particular device drivers supported by the kernel. After installation, however, support exists for all filesystems available under Linux. At the time of installation, the modularized kernel has support for (E)IDE devices (including ATAPI CD-ROM drives), SCSI adapters, and network cards.

Note

Because Red Hat Linux supports installation on many different types of hardware, some drivers (including those for SCSI adapters, network cards, and many CD-ROMs) are not built into the Linux kernel used by the installation program. Rather, they are available as modules and are loaded as you need them during the installation process. If necessary, you will have the chance to specify options for these modules when they are loaded from the driver disk.

To specify module parameters when a driver is loaded, type **linux expert** at the boot: prompt and insert the driver disk when prompted to do so by the installation program. After reading the driver disk, the installation program will ask you to select the type of device you are configuring. On that screen, you can elect to specify a module parameter. Then, the installation program will display a screen where you can type the correct parameters based on the particular type of device you are configuring.

After the installation is complete, you may want to rebuild a kernel that includes support for your specific hardware configuration. Note that in most cases, a custom-built kernel is not necessary. See the *Official Red Hat Linux Customization Guide* for more information about rebuilding your kernel.

---

[1] A **driver** is a type of software that helps your system use a particular hardware device. Without the driver, the kernel may not know how to correctly utilize the device.

A.1 Specifying Module Parameters

If you are providing parameters upon loading a module, you can usually specify them using one of two different methods:

- Specify a full set of parameters in one statement. For example, the parameter `cdu31=0x340,0` could be used with a Sony CDU 31 or 33 at port 340 with no IRQ.

- Specify the parameters individually. This method is used when one or more parameters in the first set are not needed. For example, `cdu31_port=0x340 cdu31a_irq=0` can be used as the parameter for the same CD-ROM used as an example for the first method. An *OR* is used in the CD-ROM, SCSI, and Ethernet tables in this appendix to show where the first parameter method stops and the second method begins.

Note

Only use one method, and not both, when loading a module with particular parameters.

CAUTION

When a parameter has commas, make sure you do *not* put a space after a comma.

A.2 CD-ROM Module Parameters

Note

Not all of the CD-ROM drives that are listed are supported. Please check the Hardware Compatibility List on Red Hat's website at http://hardware.red-hat.com to make sure your CD-ROM drive is supported.

Even though parameters are specified after loading the driver disk and specifying the device, one of the more commonly used parameters (hd*X*=cdrom) *can* be entered at the boot prompt (boot:) during installation. This exception to the rule is allowed since it deals with support for IDE/ATAPI CD-ROMs, which is already part of the kernel.

In the following tables, most modules listed without any parameters can either be auto-probed to find the hardware or they require you to manually change settings in the module source code and recompile.

Table A–1   Hardware Parameters

Hardware	Module	Parameters
ATAPI/IDE CD-ROM Drives		hd*X*=cdrom
Aztech CD268-01A, Orchid CD-3110, Okano/Wearnes CDD110, Conrad TXC, CyCDROM CR520, CyCDROM CR540 (non-IDE)	`aztcd.o`	aztcd=*io_port*
Sony CDU-31A CD-ROM	`cdu31a.o`	cdu31a=*io_port*,*IRQ OR* cdu31a_port=*base_addr* cdu31a_irq=*irq*
Philips/LMS CDROM drive 206 with cm260 host adapter card	`cm206.o`	cm206=*io_port*,*IRQ*
Goldstar R420 CD-ROM	`gscd.o`	gscd=*io_port*
ISP16, MAD16, or Mozart sound card CD-ROM interface (OPTi 82C928 and OPTi 82C929) with Sanyo/Panasonic, Sony, or Mitsumi drives	`isp16.o`	isp16=*io_port*,*IRQ*,*dma*, *drive_type* *OR* isp16_cdrom_base=*io_port* isp16_cdrom_irq=*IRQ* isp16_cdrom_dma=*dma* isp16_cdrom_type=*drive_type*
Mitsumi CD-ROM, Standard	`mcd.o`	mcd=*io_port*,*IRQ*
Mitsumi CD-ROM, Experimental	`mcdx.o`	mcdx=*io_port_1*,*IRQ_1*, *io_port_n*,*IRQ_n*
Optics storage 8000 AT "Dolphin" drive, Lasermate CR328A	`optcd.o`	
Parallel-Port IDE CD-ROM	`pcd.o`	
SB Pro 16 Compatible	`sbpcd.o`	sbpcd=*io_port*

Hardware	Module	Parameters
Sanyo CDR-H94A	sjcd.o	sjcd=*io_port* OR sjcd_base=*io_port*
Sony CDU-535 & 531 (some Procomm drives)	sonycd535.o	sonycd535=*io_port*

Here are some examples of these modules in use:

Table A–2   Hardware Parameters Configuration Examples

Configuration	Example
ATAPI CD-ROM, jumpered as master on the second IDE channel	hdc=cdrom
non-IDE Mitsumi CD-ROM on port 340, IRQ 11	mcd=0x340,11
Three non-IDE Mitsumi CD-ROM drives using the experimental driver, io ports 300, 304, and 320 with IRQs 5, 10 and 11	mcdx=0x300,5,0x304,10,0x320,11
Sony CDU 31 or 33 at port 340, no IRQ	cdu31=0x340,0 *OR* cdu31_port=0x340 cdu31a_irq=0
Aztech CD-ROM at port 220	aztcd=0x220
Panasonic-type CD-ROM on a SoundBlaster interface at port 230	sbpcd=0x230,1
Phillips/LMS cm206 and cm260 at IO 340 and IRQ 11	cm206=0x340,11
Goldstar R420 at IO 300	gscd=0x300
Mitsumi drive on a MAD16 soundcard at IO Addr 330 and IRQ 1, probing DMA	isp16=0x330,11,0,Mitsumi
Sony CDU 531 at IO address 320	sonycd535=0x320

Note

Most newer Sound Blaster cards come with IDE interfaces. For these cards, you do not need to use sbpcd parameters; only use hd*X* parameters.

A.3 SCSI parameters

Table A–3   SCSI Parameters

Hardware	Module	Parameters
Adaptec 28xx, R9xx, 39xx	`aic7xxx.o`	
3ware Storage Controller	`3w-xxxx.o`	
NCR53c810/820/720, NCR53c700/710/700-66	`53c7,8xx.o`	
AM53/79C974 (PC-SCSI) Driver	`AM53C974.o`	
Most Buslogic (now Mylex) cards with "BT" part number	`BusLogic.o`	
Mylex DAC960 RAID Controller	`DAC960.o`	
MCR53c406a-based SCSI	`NCR53c406a.o`	
Initio INI-9100UW	`a100u2w.o`	a100u2w=*io*,*IRQ*,*scsi_id*
Adaptec AACRAID	`aacraid.o`	
Advansys SCSI Cards	`advansys.o`	
Adaptec AHA-152x	`aha152x.o`	aha152x=*io*,*IRQ*,*scsi_id*
Adaptec AHA 154x amd 631x-based	`aha1542.o`	
Adaptec AHA 1740	`aha1740.o`	

Hardware	Module	Parameters
Adaptec AHA-274x, AHA-284x, AHA-29xx, AHA-394x, AHA-398x, AHA-274x, AHA-274xT, AHA-2842, AHA-2910B, AHA-2920C, AHA-2930/U/U2, AHA-2940/W/U/UW/AU/, U2W/U2/U2B/, U2BOEM, AHA-2944D/WD/UD/UWD, AHA-2950U2/W/B, AHA-3940/U/W/UW/, AUW/U2W/U2B, AHA-3950U2D, AHA-3985/U/W/UW, AIC-777x, AIC-785x, AIC-786x, AIC-787x, AIC-788x , AIC-789x, AIC-3860	`aic7xxx.o`	
ACARD ATP870U PCI SCSI Controller	`atp870u.o`	
Compaq Smart Array 5300 Controller	`cciss.o`	
Compaq Smart/2 RAID Controller	`cpqarray.o`	
Compaq FibreChannel Controller	`cpqfc.o`	
Domex DMX3191D	`dmx3191d.o`	
Data Technology Corp DTC3180/3280	`dtc.o`	

Hardware	Module	Parameters
DTP SCSI host adapters (EATA/DMA) PM2011B/9X ISA, PM2021A/9X ISA, PM2012A, PM2012B, PM2022A/9X EISA, PM2122A/9X, PM2322A/9X, SmartRAID PM3021, PM3222, PM3224	`eata.o`	
DTP SCSI Adapters PM2011, PM2021, PM2041, PM3021, PM2012B, PM2022, PM2122, PM2322, PM2042, PM3122, PM3222, PM3332, PM2024, PM2124, PM2044, PM2144, PM3224, PM3334	`eata_dma.o`	
Sun Enterprise Network Array (FC-AL)	`fcal.o`	
Future Domain TMC-16*xx* SCSI	`fdomain.o`	
NCR5380 (generic driver)	`g_NCR5380.o`	
ICP RAID Controller	`gdth.o`	
I2O Block Driver	`i2o_block.o`	
IOMEGA MatchMaker parallel port SCSI adapter	`imm.o`	
Always IN2000 ISA SCSI card	`in2000.o`	in2000=*setup_string*:*value OR* in2000 *setup_string=value*
Initio INI-9X00U/UW SCSI host adapters	`initio.o`	
IBM ServeRAID	`ips.o`	
AMI MegaRAID 418, 428, 438, 466, 762	`megaraid.o`	

Hardware	Module	Parameters
NCR SCSI controllers with 810/810A/815/ 825/825A/860/875/876/895 chipsets	`ncr53c8xx.o`	ncr53c8xx=*option1*:*value1*,*option2*:*value2*,... OR ncr53c8xx="*option1*:*value1 option2*:*value2*..."
Pro Audio Spectrum/Studio 16	`pas16.o`	
PCI-2000 IntelliCache	`pci2000.o`	
PCI-2220I EIDE RAID	`pci2220i.o`	
IOMEGA PPA3 parallel port SCSI host adapter	`ppa.o`	
Perceptive Solutions PSI-240I EIDE	`psi240i.o`	
Qlogic 1280	`qla1280.o`	
Qlogic 2*x*00	`qla2x00.o`	
QLogic Fast SCSI FASXXX ISA/VLB/PCMCIA	`qlogicfas.o`	
QLogic ISP2100 SCSI-FCP	`qlogicfc.o`	
QLogic ISP1020 Intelligent SCSI cards IQ-PCI, IQ-PCI-10, IQ-PCI-D	`qlogicisp.o`	
Qlogic ISP1020 SCSI SBUS	`qlogicpti.o`	
Future Domain TMC-885, TMC-950 Seagate ST-01/02, Future Domain TMC-8*xx*	`seagate.o`	controller_type=2 base_address=*base_addr* irq=*IRQ*
Cards with the sym53c416 chipset	`sym53c416.o`	sym53c416=*PORTBASE,[IRQ] OR* sym53c416 io=*PORTBASE* irq=*IRQ*
Trantor T128/T128F/T228 SCSI Host Adapter	`t128.o`	
Tekram DC-390(T) PCI	`tmscsim.o`	

Hardware	Module	Parameters
UltraStor 14F/34F (not 24F)	`u14-34f.o`	
UltraStor 14F, 24F, and 34F	`ultrastor.o`	
WD7000 Series	`wd7000.o`	

Here are some examples of these modules in use:

Table A–4   SCSI Parameters Configuration Examples

Configuration	Example
Adaptec AHA1522 at port 330, IRQ 11, SCSI ID 7	aha152x=0x330,11,7
Adaptec AHA1542 at port 330	bases=0x330
Future Domain TMC-800 at CA000, IRQ 10	controller_type=2 base_address=0xca000 irq=10

A.4 Ethernet parameters

Table A–5   Ethernet Module Parameters

Hardware	Module	Parameters
3Com 3c501	`3c501.o`	3c501=*io_port,IRQ*
3Com 3c503 and 3c503/16	`3c503.o`	3c503=*io_port,IRQ OR* 3c503 io=*io_port_1,io_port_n* irq=*IRQ_1,IRQ_n*
3Com EtherLink Plus (3c505)	`3c505.o`	3c505=*io_port,IRQ OR* 3c505 io=*io_port_1,io_port_n* irq=*IRQ_1,IRQ_2*
3Com EtherLink 16	`3c507.o`	3c507=*io_port,IRQ OR* 3c507 io=*io_port* irq=*IRQ*
3Com EtherLink III	`3c509.o`	3c509=*io_port,IRQ*
3Com ISA EtherLink XL "Corkscrew"	`3c515.o`	

Hardware	Module	Parameters
3Com EtherLink PCI III/XL Vortex (3c590, 3c592, 3c595, 3c597) Boomerang (3c900, 3c905, 3c595)	`3c59x.o`	
RTL8139, SMC EZ Card Fast Ethernet	`8139too.o`	
RealTek cards using RTL8129 or RTL8139 Fast Ethernet chipsets	`8139too.o`	
Apricot 82596	`82596.o`	
Ansel Communications Model 3200	`ac3200.o`	ac3200=*io_port*,*IRQ OR* ac3200 io=*io_port_1*,*io_port_n* irq=*IRQ_1*,*IRQ_n*
Alteon AceNIC Gigabit	`acenic.o`	
Aironet Arlan 655	`arlan.o`	
Allied Telesis AT1700	`at1700.o`	at1700=*io_port*,*IRQ OR* at1700 io=*io_port* irq=*IRQ*
Crystal Semiconductor CS89[02]0	`cs89x0.o`	
EtherWORKS DE425 TP/COAX EISA, DE434 TP PCI, DE435/450 TP/COAX/AUI PCI DE500 10/100 PCI Kingston, LinkSys, SMC8432, SMC9332, Znyx31[45], and Znyx346 10/100 cards with DC21040 (no SROM), DC21041[A], DC21140[A], DC21142, DC21143 chipsets	`de4x5.o`	de4x5=*io_port OR* de4x5 io=*io_port* de4x5 args='eth*X[fdx]* autosense=*MEDIA_STRING*'

Hardware	Module	Parameters
D-Link DE-600 Ethernet Pocket Adapter	`de600.o`	
D-Link DE-620 Ethernet Pocket Adapter	`de620.o`	
DIGITAL DEPCA & EtherWORKS DEPCA, DE100, DE101, DE200 Turbo, DE201Turbo DE202 Turbo TP/BNC, DE210, DE422 EISA	`depca.o`	depca=*io_port*,*IRQ OR* depca io=*io_port* irq=*IRQ*
Digi Intl. RightSwitch SE-X EISA and PCI	`dgrs.o`	
Davicom DM9102(A)/DM9132/ DM9801 Fast Ethernet	`dmfe.o`	
Intel EtherExpress/1000 Gigabit	`e1000.o`	
Cabletron E2100	`e2100.o`	e2100=*io_port*,*IRQ,mem OR* e2100 io=*io_port* irq=*IRQ* mem=*mem*
Intel EtherExpress Pro10	`eepro.o`	eepro=*io_port*,*IRQ OR* eepro io=*io_port* irq=*IRQ*
Intel i82557/i82558 PCI EtherExpressPro driver	`eepro100.o`	
Intel EtherExpress 16 (i82586)	`eexpress.o`	eexpress=*io_port*,*IRQ OR* eexpress io=*io_port* irq=*IRQ*
SMC EtherPower II 9432 PCI (83c170/175 EPIC series)	`epic100.o`	
Racal-Interlan ES3210 EISA	`es3210.o`	

Hardware	Module	Parameters
ICL EtherTeam 16i/32 EISA	`eth16i.o`	eth16i=*io_port*,*IRQ OR* eth16i ioaddr=*io_port* IRQ=*IRQ*
EtherWORKS 3 (DE203, DE204 and DE205)	`ewrk3.o`	ewrk=*io_port*,*IRQ OR* ewrk io=*io_port* irq=*IRQ*
A Packet Engines GNIC-II Gigabit	`hamachi.o`	
HP PCLAN/plus	`hp-plus.o`	hp-plus=*io_port*,*IRQ OR* hp-plus io=*io_port* irq=*IRQ*
HP LAN Ethernet	`hp.o`	hp=*io_port*,*IRQ OR* hp io=*io_port* irq=*IRQ*
100VG-AnyLan Network Adapters HP J2585B, J2585A, J2970, J2973, J2573 Compex ReadyLink ENET100-VG4, FreedomLine 100/VG	`hp100.o`	hp100=*io_port*,*name OR* hp100 hp100_port=*io_port* hp100_name=*name*
IBM Token Ring 16/4	`ibmtr.o`	ibmtr=*io_port* *OR* io=*io_port*
AT1500, HP J2405A, most NE2100/clone	`lance.o`	
Mylex LNE390 EISA	`lne390.o`	
NatSemi DP83815 Fast Ethernet	`natsemi.o`	
NE1000 / NE2000 (non-pci)	`ne.o`	ne=*io_port*,*IRQ OR* ne io=*io_port* irq=*IRQ*
PCI NE2000 cards RealTEk RTL-8029, Winbond 89C940, Compex RL2000, PCI NE2000 clones, NetVin, NV5000SC, Via 82C926, SureCom NE34	`ne2k-pci.o`	
Novell NE3210 EISA	`ne3210.o`	

Hardware	Module	Parameters
MiCom-Interlan NI5010	`ni5010.o`	
NI5210 card (i82586 Ethernet chip)	`ni52.o`	ni52=*io_port*,*IRQ OR* ni52 io=*io_port* irq=*IRQ*
NI6510 Ethernet	`ni65.o`	
AMD PCnet32 and AMD PCnetPCI	`pcnet32.o`	
SysKonnect SK-98*XX* Gigabit	`sk98lin.o`	
SMC Ultra and SMC EtherEZ ISA ethercard (8K, 83c790)	`smc-ultra.o`	smc-ultra=*io_port*,*IRQ OR* smc-ultra io=*io_port* irq=*IRQ*
SMC Ultra32 EISA Ethernet card (32K)	`smc-ultra32.o`	
Sun BigMac Ethernet	`sunbmac.o`	
Sundance ST201 Alta	`sundance.o`	
Sun Happy Meal Ethernet	`sunhme.o`	
Sun Quad Ethernet	`sunqe.o`	
ThunderLAN	`tlan.o`	
Digital 21x4x Tulip PCI Ethernet cards SMC EtherPower 10 PCI(8432T/8432BT) SMC EtherPower 10/100 PCI(9332DST) DEC EtherWorks 100/10 PCI(DE500-XA) DEC EtherWorks 10 PCI(DE450) DEC QSILVER's, Znyx 312 etherarray Allied Telesis LA100PCI-T Danpex EN-9400, Cogent EM110	`tulip.o`	io=*io_port*

Hardware	Module	Parameters
VIA Rhine PCI Fast Ethernet cards with either the VIA VT86c100A Rhine-II PCI or 3043 Rhine-I D-Link DFE-930-TX PCI 10/100	`via-rhine.o`	
AT&T GIS (nee NCR) WaveLan ISA Card	`wavelan.o`	wavelan=*[IRQ,0],io_port,NWID*
WD8003 and WD8013-compatible Ethernet cards	`wd.o`	wd=*io_port,IRQ,mem, mem_end OR* wd io=*io_port* irq=*IRQ* mem=*mem* mem_end=*end*
Compex RL100ATX-PCI	`winbond.o`	
Packet Engines Yellowfin	`yellowfin.o`	
Broadcom BCM5700 10/100/1000 ethernet adapter	`bcm5700`	
Intel Ether Express/100 driver	`e100`	
Shared-Memory IBM Token Ring 16/4	`ibmtr`	
IBM Olympic-based PCI token ring	`olympic`	
SIS 900/701G PCI Fast Ethernet	`sis900`	

Here are some examples of these modules in use:

Table A–6   Ethernet Parameter Configuration Examples

Configuration	Example
NE2000 ISA card at IO address 300 and IRQ 11	ne=0x300,11 ether=0x300,11,eth0
Wavelan card at IO 390, autoprobe for IRQ, and use the NWID to 0x4321	wavelan=0,0x390,0x4321 ether=0,0x390,0x4321,eth0

A.4.1 Using Multiple Ethernet Cards

You can use multiple Ethernet cards in one machine. If each card uses a different driver (for example, a 3c509 and a DE425), you simply need to add alias (and possibly options) lines for each card to /etc/modules.conf. See the *Official Red Hat Linux Customization Guide* for more information.

If any two Ethernet cards use the same driver (such as two 3c509 cards or a 3c595 and a 3c905), you will need to either give the two card addresses on the driver's options line (for ISA cards) or simply add one alias line for each card (for PCI cards).

For additional information about using more than one Ethernet card, see the *Linux Ethernet-HOWTO* at http://www.redhat.com/mirrors/LDP/HOWTO/Ethernet-HOWTO.html.

Index