Red Hat Linux 9

Red Hat Linux Reference Guide

Red Hat Linux 9: Red Hat Linux Reference Guide

Red Hat, Inc.

1801 Varsity Drive
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

rhl-rg(EN)-9-Print-RHI (2003-02-13T19:20)

The GPG fingerprint of the security@redhat.com key is:

CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

Introduction

Welcome to the *Red Hat Linux Reference Guide*.

The *Red Hat Linux Reference Guide* contains useful information about the Red Hat Linux system. From fundamental concepts, such as the structure of the Red Hat Linux file system, 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 the Red Hat Linux system works. Topics that you can explore within this manual include the following:

- The file system structure
- The boot process
- The X Window System
- Security tools
- Network services

1. Changes To This Manual

This manual has been reorganized for clarity and updated for the latest features of Red Hat Linux 9. Some of the changes include:

Updated The X Window System Chapter

: *The X Window System* has been completely revised and reorganized for clarity. New font configuration instructions were added as well.

A New `sysconfig` Chapter

: The `sysconfig` section of the *Boot Process, Init, and Shutdown* chapter has been expanded and converted into its own chapter.

Updated TCP Wrappers and `xinetd` Chapter

: The newly updated *TCP Wrappers and `xinetd`* chapter has been completely overhauled and reorganized for clarity.

Updated Users and Groups Chapter

: The *Users and Groups* chapter has been clarified, updated, and reorganized.

Updated Network Interfaces Chapter

: The *Network Interfaces* chapter has been updated and reorganized.

An Updated Apache HTTP Server Chapter

: The guide for migrating from version 1.3 to version 2.0 of the Apache HTTP Server has been updated. The list of server configuration options has been further updated and reorganized. Special thanks to **Gary Benson** and **Joe Orton** for their hard work on the Apache HTTP Server migration guide.

Before reading this guide, you should be familiar with the contents of the *Red Hat Linux Installation Guide* concerning installation issues, the *Red Hat Linux Getting Started Guide* for basic Linux concepts and the *Red Hat Linux Customization Guide* for general customization instructions. The *Red Hat Linux Reference Guide* contains information about topics for advanced users.

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

Note

Although this manual reflects the most current information possible, you should read the Red Hat Linux *Release Notes* for information that may not have been available prior to our documentation being finalized. The *Release Notes* can be found on the Red Hat Linux CD #1 and online at the following URL:

```
http://www.redhat.com/docs/manuals/linux
```

2. 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 any questions. The *Red Hat Linux Reference Guide* deals with the more technical aspects and options of a 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 informative 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 Section 2.1 *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 Section 2.2 *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 Section 2.3 *Documentation for Linux Gurus*.

2.1. 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 a 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 a 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 is 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 a 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 only need to 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 *Red Hat Linux Installation Guide* is a excellent reference for helping you get a Red Hat Linux system successfully installed and initially configured. The *Red Hat Linux Getting Started Guide* covers basic system commands, the graphical desktop environment, and many other fundamental concepts. You should start with these two books and use them to build the base of your knowledge of Red Hat Linux. 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:

2.1.1. 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.

2.1.2. 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 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://groups.google.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.

2.1.3. 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 8 Unleashed* by Bill Ball and Hoyle Duff; Pearson Education

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.

2.2. 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 *Red Hat Linux Customization Guide* is designed to help explain the various ways a 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 *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.

2.3. 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 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.

3. Document Conventions

When you read this manual, you will see that certain words are represented in different fonts, typefaces, 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 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 (such as 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.

Install the `webalizer` RPM if you want to use a Web server log file analysis program.

application

This style indicates that the program is an end-user application (as opposed to system software). For example:

Use **Mozilla** 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 exit your graphical session and return you to the graphical login screen or the console.

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 (such as text associated with a checkbox or field). Example:

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 **File** on a GNOME terminal, you will see the **New Tab** option that allows you to open multiple shell prompts in the same window.

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

Go to **Main Menu Button** (on the Panel) => **Programming** => **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 webpage 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` command to display the contents of a directory:

```
$ ls
Desktop                 about.html      logs          paulwesterberg.png
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.

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 note, tip, important, 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.

Tip

The directory /usr/share/doc contains additional documentation for packages installed on your system.

Important

If you modify the DHCP configuration file, the changes will not take effect until you restart the DHCP daemon.

Caution

Do not perform routine tasks as root — use a regular user account unless you need to use the root account for system administration tasks.

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.

4. 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 emulation, 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.

5. 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.

6. More to Come

The *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 the Red Hat Linux system — and your ability to use it.

That is where you can help.

6.1. We Need Feedback!

If you find an error in the *Red Hat Linux Reference Guide*, or if you have thought of a way to make this manual better, we would 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)-9-Print-RHI (2003-02-13T19:20)
```

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.

7. Sign Up for Support

If you have an edition of Red Hat Linux 9, 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 Red Hat Linux product you purchased:

- 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 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 Red Hat Linux box.

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

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

The Red Hat Documentation Team

I. System Reference

To manage the system effectively, it is crucial to know about its components and how they fit together. This part outlines many important aspects of the system. It covers the boot process, the basic file system layout, the location of crucial system files and file systems, and the basic concepts behind users and groups. Additionally, the X Window System is explained in detail.

Table of Contents

Chapter 1.
Boot Process, Init, and Shutdown

An important and powerful aspect of Red Hat Linux is the open, user-configurable method it uses for starting the operating system. Users are free to configure many aspects of the boot process, including specifying the programs launched at boot-time. Similarly, system shutdown gracefully terminates processes in an organized and configurable way, although customization of this process is rarely required.

Understanding how the boot and shutdown processes work not only allows customization of Red Hat Linux, but also makes it easier to troubleshoot problems related to starting or shutting down the system.

1.1. The Boot Process

Below are the basic stages of the boot process for an x86 system:

1. The system BIOS checks the system and launches the first stage boot loader on the MBR of the primary hard disk.
2. The first stage boot loader loads itself into memory and launches the second stage boot loader from the `/boot/` partition.
3. The second stage boot loader loads the kernel into memory, which in turn loads any necessary modules and mounts the root partition read-only.
4. The kernel transfers control of the boot process to the `/sbin/init` program.
5. The `/sbin/init` program loads all services and user-space tools, and mounts all partitions listed in `/etc/fstab`.
6. The user is presented with a login prompt for the freshly booted Linux system.

Because configuration of the boot process is more common than the customization of the shutdown process, the remainder of this chapter discusses in detail how the boot process works and how it can be customized to suite specific needs.

1.2. A Detailed Look at the Boot Process

The beginning of the boot process varies depending on the hardware platform being used. However, once the kernel is found and loaded by the boot loader, the default boot process is identical across all architectures. This chapter focuses on the x86 architecture.

1.2.1. The BIOS

When an x86 computer is booted, the processor looks at the end of system memory for the *Basic Input/Output System* or *BIOS* program and runs it. The BIOS controls not only the first step of the boot process, but also provides the lowest level interface to peripheral devices. For this reason it is written into read-only, permanent memory and is always available for use.

Other platforms use different programs to perform low-level tasks roughly equivalent to those of the BIOS on an x86 system. For instance, Itanium-based computers use the *Extensible Firmware Interface* (*EFI*) *Shell*, while Alpha systems use the *SRM console*.

Once loaded, the BIOS tests the system, looks for and checks peripherals, and then locates a valid device with which to boot the system. Usually, it checks any diskette drives and CD-ROM drives present for bootable media, then, failing that, looks to the system's hard drives. In most cases, the order of the drives searched while booting is controlled with a setting in BIOS, and it looks on the master IDE device on the primary IDE bus. The BIOS then loads into memory whatever program is residing in the first sector of this device, called the *Master Boot Record* or *MBR*. The MBR is only 512 bytes in size and contains machine code instructions for booting the machine, called a boot loader, along with the partition table. Once the BIOS finds and loads the boot loader program into memory, it yields control of the boot process to it.

1.2.2. The Boot Loader

This section looks at the boot loaders for the x86 platform. Depending on the system's architecture, the boot process may differ slightly. Please see Section 1.2.2.1 *Boot Loaders for Other Architectures* for a brief overview of non-x86 boot loaders.

Under Red Hat Linux two boot loaders are available: *GRUB* or *LILO*. GRUB is the default boot loader, but LILO is available for those who require or prefer it. For more information about configuring and using GRUB or LILO, see Chapter 2 *Boot Loaders*.

Both boot loaders for the x86 platform are broken into at least two stages. The first stage is a small machine code binary on the MBR. Its sole job is to locate the second stage boot loader and load the first part of it into memory.

GRUB is the newer boot loader and has the advantage of being able read ext2 and ext3 [1] partitions and load its configuration file — `/boot/grub/grub.conf` — at boot time. See Section 2.7 *GRUB Menu Configuration File* for information on how to edit this file.

With LILO, the second stage boot loader uses information on the MBR to determine the boot options available to the user. This means that any time a configuration change is made or kernel is manually upgraded, the `/sbin/lilo -v -v` command must be executed to write the appropriate information to the MBR. For details on doing this, see Section 2.8 *LILO*.

Tip

> If upgrading the kernel using the **Red Hat Update Agent**, the boot loader configuration file is updated automatically. More information on Red Hat Network can be found online at the following URL: https://rhn.redhat.com.

Once the second stage boot loader is in memory, it presents the user with the Red Hat Linux initial, graphical screen showing the different operating systems or kernels it has been configured to boot. On this screen a user can use the arrow keys to choose which operating system or kernel they wish to boot and press [Enter]. If no key is pressed, the boot loader will load the default selection after a configurable period of time has passed.

Note

> If Symmetric Multi-Processor (SMP) kernel support is installed, there will be more than one option present the first time the system is booted. In this situation, LILO will display `linux`, which is the SMP kernel, and `linux-up`, which is for single processors. GRUB displays Red Hat Linux (<*kernel-version*>-smp), which is the SMP kernel, and Red Hat Linux (<*kernel-version*>), which is for single processors.
>
> If any problems occur using the SMP kernel, try selecting the a non-SMP kernel upon rebooting.

1. GRUB reads ext3 file systems as ext2, disregarding the journal file. See the chapter titled *The ext3 File System* in the *Red Hat Linux Customization Guide* for more information on the ext3 file system.

Once the second stage boot loader has determined which kernel to boot, it locates the corresponding kernel binary in the `/boot/` directory. The kernel binary is named using the following format — `/boot/vmlinuz-`<*kernel-version*> file (where <*kernel-version*> corresponds to the kernel version specified in the boot loader's settings).

For instructions on using the boot loader to supply command line arguments to the kernel, see Chapter 2 *Boot Loaders*. For information on changing the runlevel at the GRUB or LILO prompt, see Section 2.10 *Changing Runlevels at Boot Time*.

The boot loader then places the appropriate *initial RAM disk* image, called an `initrd`, into memory. The `initrd` is used by the kernel to load drivers necessary to boot the system. This is particularly important if SCSI hard drives are present or if the systems uses the ext3 file system [2].

Warning

> Do not remove the `/initrd/` directory from the file system for any reason. Removing this directory will cause the system to fail with a kernel panic error message at boot time.

Once the kernel and the *initrd* image are loaded into memory, the boot loader hands control of the boot process to the kernel.

For a more detailed overview of the GRUB and LILO boot loaders, see Chapter 2 *Boot Loaders*.

1.2.2.1. Boot Loaders for Other Architectures

Once the Red Hat Linux kernel loads and hands off the boot process to the `init` command, the same sequence of events occurs on every architecture. So the main difference between each architecture's boot process is in the application used to find and load the kernel.

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

Consult the *Red Hat Linux Installation Guide* specific to these platforms for information on configuring their boot loaders.

1.2.3. The Kernel

When the kernel is loaded, it immediately initializes and configures the computer's memory and configures the various hardware attached to the system, including all processors, I/O subsystems, and storage devices. It then looks for the compressed `initrd` image in a predetermined location in memory, decompresses it, mounts it, and loads all necessary drivers. Next, it initializes virtual devices related to the file system, such as LVM or software RAID before unmounting the `initrd` disk image and freeing up all the memory the disk image once occupied.

The kernel then creates a root device, mounts the root partition read-only, and frees any unused memory.

At this point, the kernel is loaded into memory and operational. However, since there are no user applications that allow meaningful input to the system, not much can be done with it.

In order to set up the user environment, the kernel executes the `/sbin/init` program.

2. For details on making an *initrd*, see the chapter titled *The ext3 File System* in the *Red Hat Linux Customization Guide*.

1.2.4. The `/sbin/init` Program

The `/sbin/init` program (also called `init`) coordinates the rest of the boot process and configures the environment for the user.

When the `init` command starts, it becomes the parent or grandparent of all of the processes that start up automatically on a Red Hat Linux system. First, it runs the `/etc/rc.d/rc.sysinit` script, which sets the environment path, starts swap, checks the file systems, and takes care of everything the system needs to have done at system initialization. For example, most systems use a clock, so on them `rc.sysinit` reads the `/etc/sysconfig/clock` configuration file to initialize the hardware clock. Another example is if there are special serial port processes which must be initialized, `rc.sysinit` will execute the `/etc/rc.serial` file.

The `init` command then runs the `/etc/inittab` script, which describes how the system should be set up in each *SysV init runlevel* [3]. Among other things, the `/etc/inittab` sets the default runlevel and dictates that `/sbin/update` should be run whenever it starts a given runlevel [4].

Next, the `init` command sets the source function library, `/etc/rc.d/init.d/functions`, for the system. This spells out how to start or kill a program and how to determine the PID of a program.

The `init` program starts all of the background processes by looking in the appropriate `rc` directory for the runlevel specified as default in `/etc/inittab`. The `rc` directories are numbered to corresponds to the runlevel they represent. For instance, `/etc/rc.d/rc5.d/` is the directory for runlevel 5.

When booting to runlevel 5, the `init` program looks in the `/etc/rc.d/rc5.d/` directory to determine which processes to start and stop.

Below is an example listing of the `/etc/rc.d/rc5.d/` directory:

```
K05innd -> ../init.d/innd
K05saslauthd -> ../init.d/saslauthd
K10psacct -> ../init.d/psacct
K12cWnn -> ../init.d/cWnn
K12FreeWnn -> ../init.d/FreeWnn
K12kWnn -> ../init.d/kWnn
K12mysqld -> ../init.d/mysqld
K12tWnn -> ../init.d/tWnn
K15httpd -> ../init.d/httpd
K15postgresql -> ../init.d/postgresql
K16rarpd -> ../init.d/rarpd
K20bootparamd -> ../init.d/bootparamd
K20iscsi -> ../init.d/iscsi
K20netdump-server -> ../init.d/netdump-server
K20nfs -> ../init.d/nfs
K20rstatd -> ../init.d/rstatd
K20rusersd -> ../init.d/rusersd
K20rwalld -> ../init.d/rwalld
K20rwhod -> ../init.d/rwhod
K24irda -> ../init.d/irda
K25squid -> ../init.d/squid
K28amd -> ../init.d/amd
K34dhcrelay -> ../init.d/dhcrelay
K34yppasswdd -> ../init.d/yppasswdd
K35atalk -> ../init.d/atalk
K35dhcpd -> ../init.d/dhcpd
K35smb -> ../init.d/smb
K35vncserver -> ../init.d/vncserver
K35winbind -> ../init.d/winbind
```

3. For more information on SysV init runlevels, see Section 1.4 *SysV Init Runlevels*.
4. The `update` command is used to flush dirty buffers back to disk.

```
K40mars-nwe -> ../init.d/mars-nwe
K45arpwatch -> ../init.d/arpwatch
K45named -> ../init.d/named
K45smartd -> ../init.d/smartd
K46radvd -> ../init.d/radvd
K50netdump -> ../init.d/netdump
K50snmpd -> ../init.d/snmpd
K50snmptrapd -> ../init.d/snmptrapd
K50tux -> ../init.d/tux
K54pxe -> ../init.d/pxe
K55routed -> ../init.d/routed
K61ldap -> ../init.d/ldap
K65identd -> ../init.d/identd
K65kadmin -> ../init.d/kadmin
K65kprop -> ../init.d/kprop
K65krb524 -> ../init.d/krb524
K65krb5kdc -> ../init.d/krb5kdc
K70aep1000 -> ../init.d/aep1000
K70bcm5820 -> ../init.d/bcm5820
K74ntpd -> ../init.d/ntpd
K74ups -> ../init.d/ups
K74ypserv -> ../init.d/ypserv
K74ypxfrd -> ../init.d/ypxfrd
K84bgpd -> ../init.d/bgpd
K84ospf6d -> ../init.d/ospf6d
K84ospfd -> ../init.d/ospfd
K84ripd -> ../init.d/ripd
K84ripngd -> ../init.d/ripngd
K85zebra -> ../init.d/zebra
K90isicom -> ../init.d/isicom
K92ipvsadm -> ../init.d/ipvsadm
K95firstboot -> ../init.d/firstboot
S00microcode_ctl -> ../init.d/microcode_ctl
S05kudzu -> ../init.d/kudzu
S08ip6tables -> ../init.d/ip6tables
S08ipchains -> ../init.d/ipchains
S08iptables -> ../init.d/iptables
S09isdn -> ../init.d/isdn
S10network -> ../init.d/network
S12syslog -> ../init.d/syslog
S13portmap -> ../init.d/portmap
S14nfslock -> ../init.d/nfslock
S17keytable -> ../init.d/keytable
S20random -> ../init.d/random
S24pcmcia -> ../init.d/pcmcia
S25netfs -> ../init.d/netfs
S26apmd -> ../init.d/apmd
S28autofs -> ../init.d/autofs
S44acpid -> ../init.d/acpid
S55sshd -> ../init.d/sshd
S56rawdevices -> ../init.d/rawdevices
S56xinetd -> ../init.d/xinetd
S80sendmail -> ../init.d/sendmail
S80spamassassin -> ../init.d/spamassassin
S84privoxy -> ../init.d/privoxy
S85gpm -> ../init.d/gpm
S90canna -> ../init.d/canna
S90crond -> ../init.d/crond
```

```
S90cups -> ../init.d/cups
S90xfs -> ../init.d/xfs
S95anacron -> ../init.d/anacron
S95atd -> ../init.d/atd
S97rhnsd -> ../init.d/rhnsd
S99local -> ../rc.local
S99mdmonitor -> ../init.d/mdmonitor
```

As illustrated in this listing, none of the scripts that actually start and stop the services are located in the `/etc/rc.d/rc5.d/` directory. Rather, all of the files in `/etc/rc.d/rc5.d/` are *symbolic links* pointing to scripts located in the `/etc/rc.d/init.d/` directory. Symbolic links are used in each of the `rc` directories so that the runlevels can be reconfigured by creating, modifying, and deleting the symbolic links without affecting the actual scripts they reference.

The name of each symbolic link begin with either a `K` or an `S`. The `K` links are processes that are killed on that runlevel, while those beginning with an `S` are started.

The `init` command first stops all of the `K` symbolic links in the directory by issuing the `/etc/rc.d/init.d/`<*command*> `stop` command, where <*command*> is the process to be killed. It then starts all of the `S` symbolic links by issuing `/etc/rc.d/init.d/`<*command*> `start`.

Tip

After the system is finished booting, it is possible to log in as root and execute these same scripts to start and stop services. For instance, the command `/etc/rc.d/init.d/httpd stop` will stop the Apache Web server.

Each of the symbolic links are numbered to dictate start order. The order in which the services are started or stopped can be altered by changing this number. The lower the number, the earlier it is started. Those symbolic links with the same number are started alphabetically.

Note

One of the last things the `init` program executes is the `/etc/rc.d/rc.local` file. This file is useful for system customization. See Section 1.3 *Running Additional Programs at Boot Time* for more on using the `rc.local` file.

After the `init` command has progressed through the appropriate `rc` directory for the runlevel, the `/etc/inittab` script forks a `/sbin/mingetty` process for each virtual console (login prompts) allocated to the runlevel. Runlevels 2 through 5 get all six virtual consoles, while runlevel 1 (single user mode) gets only one and runlevels 0 and 6 get none. The `/sbin/mingetty` process opens communication pathways to *tty* devices [5], sets their modes, prints the login prompt, gets the user name, and initiates the login process for the user.

In runlevel 5, the `/etc/inittab` runs a script called `/etc/X11/prefdm`. The `prefdm` script executes the preferred X display manager — `gdm`, `kdm`, or `xdm`, depending on the contents of the `/etc/sysconfig/desktop` file.

At this point, the system is operating on runlevel 5 and displaying a login screen.

5. See Section 5.3.11 */proc/tty/* for more information on `tty` devices.

1.3. Running Additional Programs at Boot Time

The /etc/rc.d/rc.local script is executed by the init command at boot time or when changing runlevels. Adding commands to this script is an easy way to perform necessary tasks like starting special services or initialize devices without writing complex initialization scripts in the /etc/rc.d/init.d/ directory and creating symbolic links.

The /etc/rc.serial script is used if serial ports must be setup at boot time. This script runs setserial commands to configure the system's serial ports. See the setserial man page for more information.

1.4. SysV Init Runlevels

The SysV init runlevel system provides a standard process for controlling which programs init launches or halts when initializing a runlevel. SysV init was chosen because it is easier to use and more flexible than the traditional BSD-style init process.

The configuration files for SysV init are located in the /etc/rc.d/ directory. Within this directory, are the rc, rc.local, rc.sysinit, and, optionally, the rc.serial scripts as well as the following directories:

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

The init.d/ directory contains the scripts used by the /sbin/init command when controlling services. Each of the numbered directories represent the six default runlevels configured by default under Red Hat Linux.

1.4.1. Runlevels

Runlevels are a state, or *mode*, defined by the services listed in the SysV /etc/rc.d/rc<*x*>.d/ directory, where <*x*> is the number of the runlevel.

The idea behind SysV init runlevels revolves around the fact that different systems can be used in a different ways. For example, a server runs more efficiently without the drag on system resources created by the X Window System. Other times, a system administrator may need to operate the system at a lower runlevel to perform diagnostic tasks, like fixing disk corruption in runlevel 1, when no other users can possibly be on the system.

The characteristics of a given runlevel determines which services are halted and started by init. For instance, runlevel 1 (single user mode) halts any network services, while runlevel 3 starts these services. By assigning specific services to be halted or started on a given runlevel, init can quickly change the mode of the machine without the user manually stopping and starting services.

The following runlevels are defined by default for Red Hat Linux:

- 0 — Halt
- 1 — Single-user text mode
- 2 — Not used (user-definable)
- 3 — Full multi-user text mode
- 4 — Not used (user-definable)

- `5` — Full multi-user graphical mode (with an X-based login screen)
- `6` — Reboot

In general, users operate Red Hat Linux at runlevel 3 or runlevel 5 — both full multi-user modes. Users sometimes customize runlevels 2 and 4 to meet specific needs. since they are not used.

The default runlevel for the system is listed in `/etc/inittab`. To find out the default runlevel for a system, look for the line similar to the one below near the top of `/etc/inittab`:

```
id:5:initdefault:
```

The default runlevel listed in the example above is five, as the number after the first colon indicates. To change it, edit `/etc/inittab` as root.

Warning

Be very careful when editing `/etc/inittab`. Simple typos can cause the system to become unbootable. If this happens, either use a boot diskette, enter single-user mode, or enter rescue mode to boot the computer and repair the file.

For more information on single-user and rescue mode, see the chapter titled *Rescue Mode* in the *Red Hat Linux Customization Guide*.

It is possible to change the default runlevel at boot-time by modifying the arguments passed by the boot loader to the kernel. For information on changing the runlevel at boot time, see Section 2.10 *Changing Runlevels at Boot Time*.

1.4.2. Runlevel Utilities

One of the best ways to configure runlevels is to use an *initscript utility*. These tools are designed to simplify the task of maintaining files in the SysV init directory hierarchy and relieves system administrators from having to directly manipulate the numerous symbolic links in the subdirectories of `/etc/rc.d/`.

Red Hat Linux provides three such utilities:

- `/sbin/chkconfig` — The `/sbin/chkconfig` utility is a simple command-line tool for maintaining the `/etc/rc.d/init.d` directory hierarchy.
- **/sbin/ntsysv** — The ncurses-based **/sbin/ntsysv** utility provides an interactive text-based interface, which some find easier to use than `chkconfig`.
- **Services Configuration Tool** — The graphical **Services Configuration Tool** (`redhat-config-services`) program is a flexible GTK2-based utility for configuring runlevels.

Please refer to the chapter titled *Controlling Access to Services* in *Red Hat Linux Customization Guide* for more information regarding these tools.

1.5. Shutting Down

To shut down Red Hat Linux, the root user may issue the `/sbin/shutdown` command. The `shutdown` man page has a complete list of options, but the two most common uses are:

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

After shutting everything down, the `-h` option will halt the machine, and the `-r` option will reboot.

Non-root users can use the `reboot` and `halt` commands to shut down the system while in runlevels 1 through 5. However, not all Linux operating systems support this feature.

If the computer does not power itself down, be careful not turn off the computer until a message appears indicating that the system is halted.

Failure to wait for this message can mean that not all the hard drive partitions are unmounted, and can lead to file system corruption.

Chapter 2.
Boot Loaders

Before Red Hat Linux can run, it must be loaded into memory by a special program called a *boot loader*. A boot loader usually exists on the system's primary hard drive (or other media device) and has the sole responsibility of loading the Linux kernel with its required files or (in some cases) other operating systems into memory.

2.1. Boot Loaders and System Architecture

Each architecture capable of running Red Hat Linux uses a different boot loader. For example, the Alpha architecture uses the `aboot` boot loader, while the Itanium architecture uses the `ELILO` boot loader.

This chapter discusses commands and configuration options for the two boot loaders supplied with Red Hat Linux for the x86 architecture: GRUB and LILO.

2.2. GRUB

GNU GRand Unified Boot loader or GRUB is a program which enables the user to select which installed operating system or kernel to load at system boot time. It also allows the user to pass arguments to the kernel.

2.2.1. GRUB and the x86 Boot Process

This section discusses in more detail the specific role GRUB plays when booting an x86 system. For an look at the overall boot process, see Section 1.2 *A Detailed Look at the Boot Process*.

GRUB loads itself into memory in the following stages:

1. *The Stage 1 or primary boot loader is read into memory by the BIOS from the MBR*[1]. The primary boot loader exists on less than 512 bytes of disk space within the MBR and is capable of loading either the Stage 1.5 or Stage 2 boot loader.
2. *The Stage 1.5 boot loader is read into memory by the Stage 1 boot loader, if necessary.* Some hardware requires an intermediate step to get to the Stage 2 boot loader. This is sometimes true when the `/boot` partition is above the 1024 cylinder head of the hard drive or when using LBA mode. The Stage 1.5 boot loader is found either on the `/boot` partition or on a small part of the MBR and the `/boot` partition.
3. *The Stage 2 or secondary boot loader is read into memory.* The secondary boot loader displays the GRUB menu and command environment. This interface allows you to select which operating system or Linux kernel to boot, pass arguments to the kernel, or look at system parameters, such as available RAM.
4. *The secondary boot loader reads the operating system or kernel and* `initrd` *into memory.* Once GRUB determines which operating system to start, it loads it into memory and transfers control of the machine to that operating system.

The boot method used to boot Red Hat Linux is called the *direct loading* method because the boot loader loads the operating system directly. There is no intermediary between the boot loader and the kernel.

1. For more on the system BIOS and the MBR, see Section 1.2.1 *The BIOS*.

The boot process used by other operating systems may differ. For example, Microsoft's DOS and Windows operating systems, as well as various other proprietary operating systems, are loaded using a *chain loading* boot method. Under this method, the MBR points to the first sector of the partition holding the operating system. There it finds the files necessary to actually boot that operating system.

GRUB supports both direct and chain-loading boot methods, allowing it to boot almost any operating system.

Warning

During installation, the Microsoft's DOS and Windows installation program completely overwrites the MBR, destroying any existing boot loader. If creating a dual-boot system, it is best to install the Microsoft operating system first. For instructions on how to do this, see the appendix titled *Installing Red Hat Linux in a Dual-Boot Environment* in the *Red Hat Linux Installation Guide*.

2.2.2. Features of GRUB

GRUB contains a number of features that make it preferable to other boot loaders available for the x86 architecture. Below is a partial list of some of the more important features:

- *GRUB provides a true command-based, pre-OS environment on x86 machines.* This affords the user maximum flexibility in loading operating systems with certain options or gathering information about the system. For years, many non-x86 architectures have employed pre-OS environments that allow system booting from a command line. While some command features are available with LILO and other x86 boot loaders, GRUB is more feature rich.
- *GRUB supports Logical Block Addressing (LBA) mode.* LBA places the addressing conversion used to find files in the hard drive's firmware, and is used on many IDE and all SCSI hard devices. Before LBA, boot loaders could encounter the 1024-cylinder BIOS limitation, where the BIOS could not find a file after that cylinder head of the disk. LBA support allows GRUB to boot operating systems from partitions beyond the 1024-cylinder limit, so long as the system BIOS supports LBA mode. Most modern BIOS revisions support LBA mode.
- *GRUB can read ext2 partitions.* This functionality allows GRUB to access its configuration file, /boot/grub/grub.conf, every time the system boots, eliminating the need for the user to write a new version of the first stage boot loader to MBR when configuration changes are made. The only time a user would need to reinstall GRUB on the MBR is if the physical location of the /boot partition is moved on the disk. For details on installing GRUB to the MBR, see Section 2.3 *Installing GRUB*.

2.3. Installing GRUB

If GRUB was not installed during the Red Hat Linux installation process it can be installed afterward. Once installed, it automatically becomes the default boot loader.

Before installing GRUB, make sure to use the latest GRUB package available or use the GRUB package from the Red Hat Linux installation CD-ROMs. For instructions on installing packages, see the chapter titled *Package Management with RPM* in the *Red Hat Linux Customization Guide*.

Once the GRUB package is installed, open a root shell prompt and run the command /sbin/grub-install <*location*>, where <*location*> is the location that the GRUB Stage 1 boot loader should be installed.

The following command installs GRUB to the MBR of the master IDE device on the primary IDE bus: `/sbin/grub-install /dev/hda`

The next time the system boots, the GRUB graphical boot loader menu will appear before the kernel loads into memory.

2.4. GRUB Terminology

One of the most important things to understand before using GRUB is how the program refers to devices, such as hard drives and partitions. This information is particularly important when configuring GRUB to boot multiple operating systems.

2.4.1. Device Names

Suppose a system has more than one hard drive. The first hard drive of the system is called `(hd0)` by GRUB. The first partition on that drive is called `(hd0,0)`, and the fifth partition on the second hard drive is called `(hd1,4)`. In general, the naming convention for file systems when using GRUB breaks down in this way:

(<*type-of-device*><*bios-device-number*>,<*partition-number*>)

The parentheses and comma are very important to the device naming conventions. 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 the 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

GRUB's numbering system for devices starts with `0`, not `1`. Failing to make this distinction is one of the most common mistakes made by new GRUB users.

The <*partition-number*> relates to the number of a specific partition on a disk device. Like the <*bios-device-number*>, the partition numbering starts at `0`. While most partitions are specified by numbers, if a 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 system 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, 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 MBR on the first device and `(hd3)` specifies the MBR on the fourth device.
- If a system has multiple drive devices, it is very important to know the drive boot order set in the BIOS. This is rather simple to do if a system has only IDE or SCSI drives, but if there is a mix of devices, it can become confusing.

2.4.2. File Names and Blocklists

When typing commands to GRUB involving a file, such as a menu list to use when allowing the booting of multiple operating systems, it is necessary to include the file immediately after specifying the device and partition.

A sample file specification to an absolute file name is organized as follows:

```
(<type-of-device><bios-device-number>,<partition-number>)/path/to/file
```

Most of the time, a user will specify files by the directory path on that partition, plus the file name.

It is also possible to specify files to GRUB that do not actually appear in the file system, such as a chain loader that appears in the first few blocks of a partition. To specify these files, you must provide a *blocklist*, which tells GRUB, block by block, where the file is located in the partition, since 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 a comma-delimited order.

The following is a sample 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. It is possible to leave off the offset number of blocks if 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
```

The following shows the `chainloader` command with a similar blocklist designation at the GRUB command line after setting the correct device and partition as root:

```
chainloader +1
```

2.4.3. GRUB's Root File System

Some users are confused by the use of the term "root file system" with GRUB. It is important to remember that GRUB's root file system has nothing to do with the Linux root file system.

The GRUB root file system is the root partition for a particular device. GRUB uses this information to mount the device and load files from it.

With Red Hat Linux, once GRUB has loaded its root partition (which equates to the `/boot` partition and 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 the root file system Linux users are familiar with. The original GRUB root file system and its mounts are forgotten; they only existed to boot the kernel file.

Refer to the `root` and `kernel` commands in Section 2.6 *GRUB Commands* for more information.

2.5. GRUB Interfaces

GRUB features three interfaces, which provide different levels of functionality. Each of these interfaces allows users to boot the linux kernel or other operating systems.

The interfaces are as follows:

Menu Interface

If GRUB was automatically configured by the Red Hat Linux installation program, this is the interface shown by default. A menu of operating systems or kernels preconfigured with their own boot commands are displayed as a list, ordered by name. 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 is set, so that GRUB will start loading the default option.

Press the [e] key to enter the entry editor interface or the [c] key to load a command line interface.

See Section 2.7 *GRUB Menu Configuration File* for more information on configuring this interface.

Menu Entry Editor Interface

To access the menu entry editor, press the [e] key from the boot loader menu. The GRUB commands for that entry are displayed here, and users may alter these command lines before booting the operating system by adding a command line ([o] inserts a new line after the current line and [O] inserts a new line before it), editing one ([e]), or deleting one ([d]).

After all changes are made, the [b] key executes the commands and boots the operating system. The [Esc] key discards any changes and reloads the standard menu interface. The [c] key loads the command line interface.

Tip

For information about changing runlevels with GRUB using the menu entry editor, refer to Section 2.10 *Changing Runlevels at Boot Time*.

Command Line Interface

The command line interface is the most basic of the GRUB interfaces, but it is also the one that grants the most control. The command line makes it possible to type any relevant GRUB commands 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.

See Section 2.6 *GRUB Commands*, for a list of common commands.

2.5.1. Order of the Interfaces

When GRUB loads its second stage boot loader, it first searches for its configuration file. Once found, it builds a menu list and displays the menu interface.

If the configuration file cannot be found, or if the configuration file is unreadable, GRUB loads the command line interface, allowing the user to type commands to complete the boot process.

If the configuration file is not valid, GRUB prints out the error and asks for input. This helps the user see precisely where the problem occurred. Pressing any key reloads the menu interface, where it is then possible to edit the menu option and correct the problem based on the error reported by GRUB. If the correction fails, GRUB reports an error and reloads the menu interface.

2.6. GRUB Commands

GRUB allows a number of useful commands in its command line interface. Some of the commands accept options after their name; these options should be separated from the command and other options on that line by space characters.

The following is a list 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, use `+1` as the file's name.
- `displaymem` — Displays the current use of memory, based on information from the BIOS. This is useful to determine how much RAM a system has prior to booting it.
- `initrd` <*file-name*> — Enables users to specify an initial RAM disk to use when booting. An `initrd` is necessary when the kernel needs certain modules in order to boot properly, such as when the root partition is formated with the ext3 file system.
- `install` <*stage-1*> <*install-disk*> <*stage-2*> `p` <*config-file*> — Installs GRUB to the system MBR.

  When using the `install` command the user must specify the following:

  - <*stage-1*> — Signifies a device, partition, and file where the first boot loader image can be found, such as `(hd0,0)/grub/stage1`.
  - <*install-disk*> — Specifies the disk where the stage 1 boot loader should be installed, such as `(hd0)`.
  - <*stage-2*> — Passes to the stage 1 boot loader the location of the stage 2 boot loader is located, such as `(hd0,0)/grub/stage2`.
  - `p` <*config-file*> — This option tells the `install` command to look for the menu configuration file specified by <*config-file*>. An example of a valid path to the configuration file is `(hd0,0)/grub/grub.conf`.

  Warning

  The `install` command will overwrite any other information in the MBR. If executed, any information (other than GRUB information) that is used to boot other operating systems, will be lost.

- `kernel` <*kernel-file-name*> <*option-1*> <*option-N*> — Specifies the kernel file to load from GRUB's root file system when using direct loading to boot the operating system. Options can follow the `kernel` command and will be passed to the kernel when it is loaded.

  For Red Hat Linux, an example `kernel` command looks like the following:

  ```
  kernel /vmlinuz root=/dev/hda5
  ```

  This line specifies that the `vmlinuz` file is loaded from GRUB's root file system, such as `(hd0,0)`. An option is also passed to the kernel specifying that when loading the root file system for the Linux kernel, it should be on `hda5`, the fifth partition on the first IDE hard drive. Multiple options may be placed after this option, if needed.

- `root` <*device-and-partition*> — Configures GRUB's root partition to be a specific device and partition, such as `(hd0,0)`, and mounts the partition so that files can be read.
- `rootnoverify` <*device-and-partition*> — Performs the same functions as the `root` command but does not mount the partition.

Commands other than these are available. Type `info grub` for a full list of commands.

2.7. GRUB Menu Configuration File

The configuration file (`/boot/grub/grub.conf`), which is used to create the list of operating systems to boot in GRUB's menu interface, essentially allows the user to select a pre-set group of commands to execute. The commands given in Section 2.6 *GRUB Commands* can be used, as well as some special commands that are only available in the configuration file.

2.7.1. Special Configuration File Commands

The following commands can only be used in the GRUB menu configuration file:

- `color` <*normal-color*> <*selected-color*> — Allows specific colors to be used in the menu, where two colors are configured as the foreground and background. Use simple color names, such as `red/black`. For example:
  `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 fails.
- `hiddenmenu` — If used, 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, prevents a user who does not know the password from editing the entries for this menu option.

  Optionally, it is possible to specify an alternate menu configuration file after the `password` <*password*> command. In this case, GRUB will restart the second stage boot loader and use the specified alternate configuration file to build the menu. If an alternate menu configuration file is left out of the command, then a user who knows the password is allowed to edit the current configuration file.
- `timeout` — If used, sets the interval, 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 hash mark (#) character can be used at the beginning of a line to place comments in the menu configuration file.

2.7.2. Configuration File Structure

The GRUB menu interface configuration file is `/boot/grub/grub.conf`. The commands to set the global preferences for the menu interface are placed at the top of the file, followed by the different entries for each of the operating systems or kernels listed in the menu.

The following is a very basic GRUB menu configuration file designed to boot either Red Hat Linux and Microsoft Windows 2000:

```
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz

# section to load linux
title Red Hat Linux (2.4.18-5.47)
        root (hd0,0)
```

```
        kernel /vmlinuz-2.4.18-5.47 ro root=/dev/sda2
        initrd /initrd-2.4.18-5.47.img

# section to load Windows 2000
title windows
        rootnoverify (hd0,0)
        chainloader +1
```

This file tells GRUB to build a menu with Red Hat Linux as the default operating system and sets it to autoboot after 10 seconds. Two sections are given, one for each operating system entry, with commands specific to the system disk partition table.

Note

Note that the default is specified as a number. This refers to the first `title` line GRUB comes across. If you want `windows` to be the default, change the `default=0` to `default=1`.

Configuring a GRUB menu configuration file to boot multiple operating systems is beyond the scope of this chapter. Please consult Section 2.11 *Additional Resources* for a list of additional resources.

2.8. LILO

LILO is an acronym for the *LI*nux *LO*ader and has been used to boot Linux on x86 systems for many years. Although GRUB is now the default boot loader, some users prefer to use LILO because it is more familiar to them and others use it out of necessity, since GRUB may have trouble booting some hardware.

2.8.1. LILO and the x86 Boot Process

This section discusses in detail the specific role LILO plays when booting an x86 system. For a detailed look at the overall boot process, see Section 1.2 *A Detailed Look at the Boot Process*.

LILO loads itself into memory almost identically to GRUB, except it is only a two stage loader.

1. *The Stage 1 or primary boot loader is read into memory by the BIOS from the MBR*[2]. The primary boot loader exists on less than 512 bytes of disk space within the MBR. It only loads the Stage 2 boot loader and passes disk geometry information to it.
2. *The Stage 2 or secondary boot loader is read into memory.* The secondary boot loader displays the Red Hat Linux initial screen. This screen allows you to select which operating system or Linux kernel to boot.
3. *The Stage 2 boot loader reads the operating system or kernel and `initrd` into memory.* Once LILO determines which operating system to start, it loads it into memory and hands control of the machine to that operating system.

Once the Stage 2 boot loader is in memory, LILO displays the initial Red Hat Linux screen with the different operating systems or kernels it has been configured to boot. By default, if Red Hat Linux is the only operating system installed, **linux** will be the only available option. If the system has multiple processors there will be a **linux-up** option for the single processor kernel and a **linux** option for the

2. For more on the system BIOS and the MBR, see Section 1.2.1 *The BIOS*.

multiple processor (SMP) kernel. If LILO is configured to boot other operating systems, those boot entries also appear on this screen.

The arrow keys allow a user to highlight the desired operating system and the [Enter] key begins the boot process.

To access a `boot:` prompt, press [Ctrl]-[X].

2.8.2. LILO versus GRUB

In general, LILO works similarly to GRUB except for three major differences:

- It has no interactive command interface.
- It stores information about the location of the kernel or other operating system it is to load on the MBR.
- It cannot read ext2 partitions.

The first point means the command prompt for LILO is not interactive and only allows one command with arguments.

The last two points mean that if you change LILO's configuration file or install a new kernel, you must rewrite the Stage 1 LILO boot loader to the MBR by using the following command:

```
/sbin/lilo -v -v
```

This method is more risky than the method used by GRUB because a misconfigured MBR leaves the system unbootable. With GRUB, if the configuration file is erroneously configured, it will default to its command line interface where the user can boot the system manually.

Tip

If upgrading the kernel using the **Red Hat Update Agent**, the MBR will be updated automatically. More information about RHN is available online at the following URL: https://rhn.redhat.com

2.9. Options in `/etc/lilo.conf`

The LILO configuration file is `/etc/lilo.conf`. The `/sbin/lilo` commands uses this file to determine what information to write to the MBR.

Warning

Before editing `/etc/lilo.conf`, be sure to make a backup copy of the file. Also, have a working boot floppy available so that changes can be made to the MBR if there is a problem. See the man page for `mkbootdisk` for more information on creating a boot disk.

The `/etc/lilo.conf` file is used by the `/sbin/lilo` command to determine which operating system or kernel to load and where it should be installed.

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. Next is a more detailed look at the lines of this file:

- `boot=/dev/hda` — Instructs LILO to install itself on the first hard disk of the first IDE controller.
- `map=/boot/map` — Locates the map file. In normal use, this should not be modified.
- `install=/boot/boot.b` — Instructs 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.
- `prompt` — Instructs 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 access 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 an unbootable state.
- `default=linux` — Refers to the default operating system for LILO to boot as seen in 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 which Linux kernel to boot with this particular boot option.
- `label=linux` — Names the operating system option in the LILO screen. In this case, it is also the name 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 a SCSI card, hard drive, or any other device needed to load the kernel. You should never try to share initial ram disks between machines.
- `read-only` — Specifies that the root partition (see the `root` line below) is read-only and cannot be altered during the boot process.
- `root=/dev/hda5` — Specifies which disk partition to use as the root partition.

- `other=/dev/hda1` — Specifies the partition containing DOS.

2.10. Changing Runlevels at Boot Time

Under Red Hat Linux, it is possible to change the default runlevel at boot time.

If using LILO, access the `boot:` prompt by typing [Ctrl]-[X]. Then type:

```
linux <runlevel-number>
```

In this command, replace <*runlevel-number*> with either the number of the runlevel to boot into (**1** through **5**), or the words **single** or **emergency**.

If using GRUB, follow these 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 the number of the runlevel you wish to boot into (**1** through **5**), or the words **single** or **emergency** and press [Enter].
- You will be returned to the GRUB screen with the kernel information. Press the [b] key to boot the system.

For more information about runlevels, see Section 1.4.1 *Runlevels*.

2.11. Additional Resources

This chapter is only intended as an introduction to GRUB and LILO. Consult the following resources to discover more about how GRUB and LILO work.

2.11.1. Installed Documentation

- `/usr/share/doc/grub-`<*version-number*>`/` — This directory contains good information about using and configuring GRUB. The <*version-number*> in the path to this file corresponds to the version of the GRUB package installed.
- The GRUB info page, accessible by typing the `info grub` command, contains a tutorial, a user reference manual, a programmer reference manual, and a FAQ document about GRUB and its usage.
- `/usr/share/doc/lilo-`<*version-number*>`/` — This directory contains a wealth of information about using and configuring LILO. In particular, the `doc/` subdirectory contains a postscript file called `User_Guide.ps` that is highly informative. The <*version-number*> in the path to this directory corresponds to the version of the LILO package installed.

2.11.2. 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 a 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 a system from scratch, including an overview of GRUB command line options.
- http://www.tldp.org/HOWTO/mini/LILO.html — This mini-HOWTO discusses various uses for LILO, including booting operating systems other than Linux.

Chapter 3.

File System Structure

3.1. Why Share a Common Structure?

An operating system's file system 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 storage device. It is crucial for a variety of reasons that users, as well as programs, be able to refer to a common guideline to know where to read and write files.

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

- Shareable vs. unsharable files
- Variable vs. static files

Shareable files are those that can be accessed by various hosts; *unsharable* files are not available to any other hosts. *Variable* files can change at any time without any intervention; *static* files, such as read-only 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 manner is to help correlate the function of the file with the permissions assigned to the directories which hold them. The way in which the operating system and its users interact with a given file determines the directory in which it is placed, whether that directory is mounted read-only or read-write, and the level of access each user has to that 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-used structure.

However, 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 file system structure and extended it only slightly to accommodate special files used within Red Hat Linux.

3.2. Overview of File System 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.

The FHS document is the authoritative reference to any FHS-compliant file system, but the standard leaves many areas undefined or extensible. This section is an overview of the standard and a description of the parts of the file system 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 a `/usr/` partition as read-only because it contains common executables and should not be changed by users. Since the `/usr/` directory is mounted read-only, it can be mounted from the CD-ROM or from another machine via a read-only NFS mount.

3.2.1. FHS Organization

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

3.2.1.1. The /dev/ Directory

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

3.2.1.2. The /etc/ Directory

The /etc/ directory is reserved for configuration files that are local to the machine. No binaries are to be put in /etc/. Any binaries that were once located in /etc/ should be placed into /sbin/ or possibly /bin/.

The X11/ and skel/ directories are subdirectories of the /etc/ directory:

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

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

3.2.1.3. 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 file system.

3.2.1.4. The /mnt/ Directory

The /mnt/ directory is for temporarily mounted file systems, such as CD-ROMs and floppy disks.

3.2.1.5. The /opt/ Directory

The /opt/ directory provides storage for large, static application software packages.

A package placing files in the /opt/ directory creates a directory bearing the same name as the package. This directory in turn holds files that otherwise would be scattered throughout the file system, giving 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 the /opt/ directory, then all of its files could be placed within directories inside the /opt/sample/ directory, 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 the /opt/ directory, 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.

3.2.1.6. 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 refer to Chapter 5 *The `proc` File System*.

3.2.1.7. 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
```

3.2.1.8. 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. At minimum, the following directories should be subdirectories of `/usr/`:

```
/usr
  |- bin/
  |- dict/
  |- doc/
  |- etc/
  |- games/
  |- include/
  |- kerberos/
  |- lib/
  |- libexec/
  |- local/
  |- sbin/
  |- share/
  |- src/
  |- tmp -> ../var/tmp/
  |- X11R6/
```

The `bin/` directory contains executables, `dict/` 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 the `/sbin/` directory), `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).

3.2.1.9. 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 among 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/
```

3.2.1.10. The `/var/` Directory

Since the FHS requires Linux to mount `/usr/` read-only, any programs that write log files or need `spool/` or `lock/` directories should 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."

Below are some of the directories which should be subdirectories of the `/var/` directory:

```
/var
  |- account/
  |- arpwatch/
  |- cache/
  |- crash/
  |- db/
  |- empty/
  |- ftp/
  |- gdm/
  |- kerberos/
  |- lib/
  |- local/
  |- lock/
  |- log/
  |- mail -> spool/mail/
```

```
|- mailman/
|- named/
|- nis/
|- opt/
|- preserve/
|- run/
+- spool/
     |- anacron/
     |- at/
     |- cron/
     |- fax/
     |- lpd/
     |- mail/
     |- mqueue/
     |- news/
     |- rwho/
     |- samba/
     |- slrnpull/
     |- squid/
     |- up2date/
     |- uucp/
     |- uucppublic/
     |- vbox/
     |- voice/
|- tmp/
|- tux/
|- www/
|- yp/
```

System log files such as `messages/` and `lastlog/` go in the `/var/log/` directory. The `/var/lib/rpm/` directory also contains the RPM system databases. Lock files go in the `/var/lock/` directory, 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.

3.2.2. `/usr/local/` in Red Hat Linux

In Red Hat Linux, the intended use for the `/usr/local/` directory is slightly different from that specified by the FHS. The FHS says that `/usr/local/` should be where software that is to remain safe from system software upgrades is stored. Since system upgrades from under Red Hat Linux performed safely with the `rpm` command and graphical **Package Management Tool** application, it is not necessary to protect files by putting them in `/usr/local/`. Instead, the `/usr/local/` directory is used for software that is local to the machine.

For instance, if the `/usr/` directory is mounted as a read-only NFS share from a remote host, it is still possible to install a package or program under the `/usr/local/` directory.

3.3. Special File Locations

Red Hat Linux extends the FHS structure slightly to accommodate special files.

Most files pertaining to the *Red Hat Package Manager* (*RPM*) are kept in the `/var/lib/rpm/` directory. For more information on RPM see the chapter titled *Package Management with RPM* in the *Red Hat Linux Customization Guide*.

The `/var/spool/up2date/` directory contains files used by **Red Hat Update Agent**, including RPM header information for the system. This location may also be used to temporarily store RPMs downloaded while updating the system. For more information on Red Hat Network, refer to the Red Hat Network website at https://rhn.redhat.com/.

Another location specific to Red Hat Linux is the `/etc/sysconfig/` directory. This directory stores a variety of configuration information. Many scripts that run at boot time use the files in this directory. See Chapter 4 *The `sysconfig` Directory* for more information about what is within this directory and the role these files play in the boot process.

Finally, one more directory worth noting is the `/initrd/` directory. It is empty, but is used as a critical mount point during the boot process.

Warning

Do not remove the `/initrd/` directory for any reason. Removing this directory will cause the system to fail to boot with a kernel panic error message.

Chapter 4.

The `sysconfig` Directory

The `/etc/sysconfig/` directory is where a variety of system configuration files for Red Hat Linux are stored.

This chapter outlines some of the files found in the `/etc/sysconfig/` directory, their function, and their contents. The information in this chapter 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.

4.1. Files in the `/etc/sysconfig/` Directory

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

- `amd`
- `apmd`
- `arpwatch`
- `authconfig`
- `cipe`
- `clock`
- `desktop`
- `dhcpd`
- `firstboot`
- `gpm`
- `harddisks`
- `hwconf`
- `i18n`
- `identd`
- `init`
- `ipchains`
- `iptables`
- `irda`
- `keyboard`
- `kudzu`
- `mouse`
- `named`
- `netdump`
- `network`
- `ntpd`
- `pcmcia`
- `radvd`

- rawdevices
- redhat-config-securitylevel
- redhat-config-users
- redhat-logviewer
- samba
- sendmail
- soundcard
- spamassassin
- squid
- tux
- ups
- vncservers
- xinetd

Note

If some of the files listed are not present in the /etc/sysconfig/ directory, then the corresponding program may not be installed.

4.1.1. /etc/sysconfig/amd

The /etc/sysconfig/amd file contains various parameters used by amd, which allow for the automatic mounting and unmounting of file systems.

4.1.2. /etc/sysconfig/apmd

The /etc/sysconfig/apmd file is used by apmd as a configuration for what power settings to start/stop/change on suspend or resume. It is configured to turn on or off apmd at boot time, depending on whether the hardware supports *Advanced Power Management* (*APM*) or whether or not the user has configured the system to use it. The apm daemon is a monitoring program that works with power management code within the Linux kernel. It capable of alerting users to low battery power on laptops and other power-related settings.

4.1.3. /etc/sysconfig/arpwatch

The /etc/sysconfig/arpwatch file is used to pass arguments to the arpwatch daemon at boot time. The arpwatch daemon maintains a table of Ethernet MAC addresses and their IP address pairings. For more information about what parameters are available for this file, see the arpwatch man page. By default, this file sets the owner of the arpwatch process to the user pcap.

4.1.4. `/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.

4.1.5. `/etc/sysconfig/clock`

The `/etc/sysconfig/clock` file controls the interpretation of values read from the system hardware clock.

The correct values are:

- `UTC=`<*value*>, where <*value*> is one of the following boolean values:
  - `true` or `yes` — The hardware clock is set to Universal Time.
  - `false` or `no` — The hardware clock is set to local time.
- `ARC=`<*value*>, where <*value*> is the following:
  - `true` or `yes` — The ARC console's 42-year time offset is in effect. This setting is only for ARC- or AlphaBIOS-based Alpha systems. Any other value indicates that the normal UNIX epoch is in use.
- `SRM=`<*value*>, where <*value*> is the following:
  - `true` or `yes` — The SRM console's 1900 epoch is in effect. This setting is only for SRM-based Alpha systems. Any other value indicates that the normal UNIX epoch is in use.
- `ZONE=`<*filename*> — The timezone file under `/usr/share/zoneinfo` that `/etc/localtime` is a copy of. The file contains information such as:
  `ZONE="America/New York"`

Earlier releases of Red Hat Linux used the following values (which are deprecated):

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

4.1.6. /etc/sysconfig/desktop

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

```
DESKTOP="GNOME"
```

4.1.7. /etc/sysconfig/dhcpd

The /etc/sysconfig/dhcpd file is used to pass arguments to the dhcpd daemon at boot time. The dhcpd daemon implements the Dynamic Host Configuration Protocol (DHCP) and the Internet Bootstrap Protocol (BOOTP). DHCP and BOOTP assign hostnames to machines on the network. For more information about what parameters are available in this file, see the dhcpd man page.

4.1.8. /etc/sysconfig/firstboot

Beginning with Red Hat Linux 8.0, the first time the system boots, the /sbin/init program calls the etc/rc.d/init.d/firstboot script, which in turn launches **Setup Agent**. This application allows the user to install the latest updates as well as additional applications and documentation.

The /etc/sysconfig/firstboot file tells the **Setup Agent** application not to run on subsequent reboots. To run it the next time the system boots, remove /etc/sysconfig/firstboot and execute chkconfig --level 5 firstboot on.

4.1.9. /etc/sysconfig/gpm

The /etc/sysconfig/gpm file is used to pass arguments to the gpm daemon at boot time. The gpm daemon is the mouse server which allows mouse acceleration and middle-click pasting. For more information about what parameters are available for this file, see the gpm man page. By default, it sets the mouse device to /dev/mouse.

4.1.10. /etc/sysconfig/harddisks

The /etc/sysconfig/harddisks file tunes the hard drive(s). The administrator can also use /etc/sysconfig/hardiskhd[a-h] to configure parameters for specific drives.

Warning

Do not make changes to this file without careful consideration. By changing the default values, it is possible to corrupt all of the data on the hard drive(s).

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

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

4.1.11. /etc/sysconfig/hwconf

The /etc/sysconfig/hwconf file lists all the hardware that kudzu detected on the 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 edited, devices could suddenly show up as being added or removed.

4.1.12. /etc/sysconfig/i18n

The /etc/sysconfig/i18n file sets the default language, any supported languages, and the default system font. For example:

```
LANG="en_US.UTF-8"
SUPPORTED="en_US.UTF-8:en_US:en"
SYSFONT="latarcyrheb-sun16"
```

4.1.13. /etc/sysconfig/identd

The /etc/sysconfig/identd file is used to pass arguments to the identd daemon at boot time. The identd daemon returns the username of processes with open TCP/IP connections. Some services on the network, such as FTP and IRC servers, will complain and cause slow responses if identd is not running. But in general, identd is not a required service, so if security is a concern, do not run it. For more information about what parameters are available for this file, see the identd man page. By default, the file contains no parameters.

4.1.14. /etc/sysconfig/init

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

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 via the echo -en command.

- SETCOLOR_SUCCESS=<*value*>, where <*value*> sets the color to a color indicating success via the echo -en command. The default color is set to green.
- SETCOLOR_FAILURE=<*value*>, where <*value*> sets the color to a color indicating failure via the echo -en command. The default color is set to red.
- SETCOLOR_WARNING=<*value*>, where <*value*> sets the warning color via the echo -en command. The default color is set to yellow.
- SETCOLOR_NORMAL=<*value*>, where <*value*> resets the color to "normal" via the echo -en.
- LOGLEVEL=<*value*>, where <*value*> sets the initial console logging level for the kernel. The default is 3; 8 means everything (including debugging); 1 means nothing except kernel panics. The syslogd daemon overrides this setting once started.
- 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.

4.1.15. /etc/sysconfig/ipchains

The /etc/sysconfig/ipchains file contains information used by the ipchains initialization script when setting up the ipchains service.

This file is modified by typing the command /sbin/service ipchains save when valid ipchains rules are in place. Do not manually edit this file. Instead, use the /sbin/ipchains command to configure the necessary packet filtering rules and then save the rules to this file using /sbin/service ipchains save.

Use of ipchains to set up firewall rules is not recommended as it is deprecated and may disappear from future releases of Red Hat Linux. If a firewall is necessary, use iptables instead.

4.1.16. /etc/sysconfig/iptables

Like /etc/sysconfig/ipchains, the /etc/sysconfig/iptables file stores information used by the kernel to set up packet filtering services at boot time or whenever the service is started.

Do not modify this file by hand unless familiar with how to construct iptables rules. The easiest way to add rules is to use **Security Level Configuration Tool** (redhat-config-securitylevel), the /usr/sbin/lokkit command, or the **GNOME Lokkit** application to create a firewall. Using these applications automatically edit this file at the end of the process.

Rules can be created manually by using /sbin/iptables: then type /sbin/service iptables save to add the rules to the /etc/sysconfig/iptables file.

Once this file exists, any firewall rules saved in it persists through a system reboot or a service restart.

For more information on iptables see Chapter 16 *iptables*.

4.1.17. /etc/sysconfig/irda

The /etc/sysconfig/irda file controls how infrared devices on the 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 the 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.

4.1.18. /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 /lib/kbd/keymaps/i386 and branch into different keyboard layouts from there, all labeled <*file*>.kmap.gz. The first file found beneath /lib/kbd/keymaps/i386 that matches the KEYTABLE setting is used.

4.1.19. /etc/sysconfig/kudzu

The /etc/sysconfig/kuzdu file triggers a safe probe of the system 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.

4.1.20. /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:
  - imps2 — A generic USB wheel mouse.
  - 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.

4.1.21. /etc/sysconfig/named

The /etc/sysconfig/named file is used to pass arguments to the named daemon at boot time. The named daemon is a *Domain Name System* (*DNS*) server which implements the *Berkeley Internet Name Domain* (*BIND*) version 9 distribution. This server maintains a table of which hostnames are associated with IP addresses on the network.

Currently, only the following values may be used:

- ROOTDIR="<*/some/where*>", where <*/some/where*> refers to the full directory path of a configured chroot environment under which named runs. This chroot environment must first be configured. Type info chroot for more information.
- OPTIONS="<*value*>", where <*value*> is any option listed in the man page for named except -t. In place of -t, use the ROOTDIR line above.

For more information about what parameters are available for this file, see the named man page. For detailed information on how to configure a BIND DNS server, see Chapter 12 *Berkeley Internet Name Domain (BIND)*. By default, the file contains no parameters.

4.1.22. /etc/sysconfig/netdump

The /etc/sysconfig/netdump file is the configuration file for the /etc/init.d/netdump service. The netdump service sends both oops data and memory dumps over the network. In general, netdump is not a required service; only run it if absolutely necessary. For more information about what parameters are available for this file, see the netdump man page.

4.1.23. /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.expample.com, but can be whatever hostname is necessary.

  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.

4.1.24. /etc/sysconfig/ntpd

The /etc/sysconfig/ntpd file is used to pass arguments to the ntpd daemon at boot time. The ntpd daemon sets and maintains the system clock to synchronize with an Internet standard time server. It implements version 4 of the Network Time Protocol (NTP). For more information about what parameters are available for this file, point a browser at the following file: /usr/share/doc/ntp-<*version*>/ntpd.htm (where <*version*> is the version number of ntpd). By default, this file sets the owner of the ntpd process to the user ntp.

4.1.25. /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.

4.1.26. /etc/sysconfig/radvd

The /etc/sysconfig/radvd file is used to pass arguments to the radvd daemon at boot time. The radvd daemon listens to for router requests and sends router advertisements for the IP version 6 protocol. This service allows hosts on a network to dynamically change their default routers based on these router advertisements. For more information about what parameters are available for this file, see the radvd man page. By default, this file sets the owner of the radvd process to the user radvd.

4.1.27. /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
```

4.1.28. /etc/sysconfig/redhat-config-securitylevel

The /etc/sysconfig/redhat-config-securitylevel file contains all options chosen by the user the last time the **Security Level Configuration Tool** (redhat-config-securitylevel) was run. Users should not modify this file by hand. For more information about **Security Level Configuration Tool**, see the chapter titled *Basic Firewall Configuration* in the *Red Hat Linux Customization Guide*.

4.1.29. /etc/sysconfig/redhat-config-users

The /etc/sysconfig/redhat-config-users file is the configuration file for the graphical application, **User Manager**. Under Red Hat Linux this file is used to filter out system users such as root, daemon, or lp. This file is edited by the **Preferences** => **Filter system users and groups** pull-down menu in the **User Manager** application and should not be edited by hand. For more information on using this application, see the chapter called *User and Group Configuration* in the *Red Hat Linux Customization Guide*.

4.1.30. /etc/sysconfig/redhat-logviewer

The /etc/sysconfig/redhat-logviewer file is the configuration file for the graphical, interactive log viewing application, **Log Viewer**. This file is edited by the **Edit** => **Preferences** pull-down menu in the **Log Viewer** application and should not be edited by hand. For more information on using this application, see the chapter called *Log Files* in the *Red Hat Linux Customization Guide*.

4.1.31. /etc/sysconfig/samba

The /etc/sysconfig/samba file is used to pass arguments to the smbd and the nmbd daemons at boot time. The smbd daemon offers file sharing connectivity for Windows clients on the network. The nmbd daemon offers NetBIOS over IP naming services. For more information about what parameters are available for this file, see the smbd man page. By default, this file sets smbd and nmbd to run in daemon mode.

4.1.32. /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.

4.1.33. /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. Sound card 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.

4.1.34. /etc/sysconfig/spamassassin

The /etc/sysconfig/spamassassin file is used to pass arguments to the spamd daemon (a daemonized version of Spamassassin) at boot time. Spamassassin is an email spam filter application. For a list of available options, see the spamd man page. By default, it configures spamd to run in daemon mode, create user preferences, and auto-create whitelists.

For more information about Spamassassin, see Section 11.4.2.6 *Spam Filters*.

4.1.35. /etc/sysconfig/squid

The /etc/sysconfig/squid file is used to pass arguments to the squid daemon at boot time. The squid daemon is a proxy caching server for Web client applications. For more information on configuring a `squid` proxy server, use a Web browser to open the `/usr/share/doc/squid-<version>/` directory (replace `<version>` with the `squid` version number installed on the system). By default, this file sets `squid` to start in daemon mode and sets the amount of time before it shuts itself down.

4.1.36. /etc/sysconfig/tux

The `/etc/sysconfig/tux` file is the configuration file for the Red Hat Content Accelerator (formerly known as TUX), the kernel-based Web server. For more information on configuring the Red Hat Content Accelerator, use a Web browser to open the `/usr/share/doc/tux-<version>/tux/index.html` (replace `<version>` with the version number of TUX installed on the system). The parameters available for this file are listed in `/usr/share/doc/tux-<version>/tux/parameters.html`.

4.1.37. /etc/sysconfig/ups

The `/etc/sysconfig/ups` file is used to specify information about any *Uninterruptible Power Supplies* (*UPS*) connected to the system. A UPS can be very valuable for a Red Hat Linux system because it gives 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 the system.
  - `no` — A UPS device is not connected to the system.
- `MODEL=<value>`, where `<value>` must be one of the following or set to `NONE` if no UPS is connected to the system:
  - `apcsmart` — An APC SmartUPS™ or similar device.
  - `fentonups` — A Fenton UPS™.
  - `optiups` — An OPTI-UPS™ device.
  - `bestups` — A Best Power™ UPS.
  - `genericups` — A generic brand UPS.
  - `ups-trust425+625` — 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.

4.1.38. /etc/sysconfig/vncservers

The `/etc/sysconfig/vncservers` file configures the way the *Virtual Network Computing* (*VNC*) server starts up.

VNC is a remote display system which allows users to view the desktop environment not only on the machine where it is running but across different networks on a variety of 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 using a VNC server, 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 18 *SSH Protocol* or *Red Hat Linux Customization Guide*.

4.1.39. /etc/sysconfig/xinetd

The `/etc/sysconfig/xinetd` file is used to pass arguments to the `xinetd` daemon at boot time. The `xinetd` daemon starts programs that provide Internet services when a request to the port for that service is received. For more information about the parameters available for this file, see the `xinetd` man page. For more information on the `xinetd` service, see Section 15.3 *xinetd*.

4.2. Directories in the /etc/sysconfig/ Directory

The following directories are normally found in `/etc/sysconfig/`.

- `apm-scripts/` — This directory contains the Red Hat APM suspend/resume script. Do not edit the files directly. If customization is necessary, create a file called `/etc/sysconfig/apm-scripts/apmcontinue` and it will be called at the end of the script. It is also possible to control the script by editing `/etc/sysconfig/apmd`.
- `cbq/` — This directory contains the configuration files needed to do Class Based Queuing for bandwidth management on network interfaces.
- `networking/` — This directory is used by the **Network Administration Tool** (`redhat-config-network`), and its contents should not be edited manually. For more information about configuring network interfaces using the **Network Administration Tool**, see the chapter called *Network Configuration* in the *Red Hat Linux Customization Guide*.
- `network-scripts/` — This directory contains the following network-related configuration files:
  - Network configuration files for each configured network interface, such as `ifcfg-eth0` for the `eth0` Ethernet interface.
  - Scripts used to bring up and down network interfaces, such as `ifup` and `ifdown`.
  - Scripts used to bring up and down ISDN interfaces, such as `ifup-isdn` and `ifdown-isdn`.
  - Various shared network function scripts which should not be edited directly.

  For more information on the `network-scripts` directory, see Chapter 8 *Network Interfaces*.
- `rhn/` — This directory contains the configuration files and GPG keys for the Red Hat Network. No files in this directory should be edited by hand. For more information on the Red Hat Network, see the Red Hat Network website at the following URL: https://rhn.redhat.com.

4.3. Additional Resources

This chapter is only intended as an introduction to the files in the `/etc/sysconfig/` directory. The following sources contain more comprehensive information.

4.3.1. Installed Documentation

- `/usr/share/doc/initscripts-`<*version-number*>`/sysconfig.txt` — This file contains a more authoritative listing of the files found in the `/etc/sysconfig/` directory and the configuration options available for them. The <*version-number*> in the path to this file corresponds to the version of the `initscripts` package installed.

Chapter 5. The proc File System

The Linux kernel has two primary functions: to control access to physical devices on the computer and to schedule when and how processes interact with these devices. The /proc/ directory contains a hierarchy of special files which represent the current state of the kernel — allowing applications and users to peer into the kernel's view of the system.

Within the /proc/ directory, one can find a wealth of information detailing the system hardware and any processes currently running. In addition, some of the files within the /proc/ directory tree can be manipulated by users and applications to communicate configuration changes to the kernel.

5.1. A Virtual File System

Under Linux, all data are stored as files. Most users are familiar with the two primary types of files: text and binary. But the /proc/ directory contains another type of file called a *virtual file*. It is for this reason that /proc/ is often referred to as a *virtual file system*.

These virtual files have unique qualities. Most of them are listed as zero bytes in size and yet when one is viewed, it can contain a large amount of information. In addition, most of the time and date settings on virtual files reflect the current time and date, indicative of the fact they are constantly updated.

Virtual files such as /proc/interrupts, /proc/meminfo, /proc/mounts, and /proc/partitions provide an up-to-the-moment glimpse of the system's hardware. Others, like /proc/filesystems and the /proc/sys/ directory provide system configuration information and interfaces.

For organizational purposes, files containing information on a similar topic are grouped into virtual directories and sub-directories. For instance, /proc/ide/ contains information for all physical IDE devices. Likewise, process directories contain information about each running process on the system.

5.1.1. Viewing Virtual Files

By using the cat, more, or less commands on files within the /proc/ directory, users can immediately access an enormous amount of information about the system. For example, to display the type of CPU a computer has, type cat /proc/cpuinfo to receive output similar to the following:

```
processor : 0
vendor_id : AuthenticAMD
cpu family : 5
model  : 9
model name : AMD-K6(tm) 3D+ Processor
stepping : 1
cpu MHz  : 400.919
cache size : 256 KB
fdiv_bug : no
hlt_bug  : no
f00f_bug : no
coma_bug : no
fpu  : yes
fpu_exception : yes
cpuid level : 1
wp  : yes
flags  : fpu vme de pse tsc msr mce cx8 pge mmx syscall 3dnow k6_mtrr
bogomips : 799.53
```

When viewing different virtual files in the `/proc/` file system, some of the information is easily understandable while some is not human-readable. This is in part why utilities exist to pull data from virtual files and display it in a useful way. Examples of these utilities include `lspci`, `apm`, `free`, and `top`.

Note

Some of the virtual files in the `/proc/` directory are readable only by the root user.

5.1.2. Changing Virtual Files

As a general rule, most virtual files within the `/proc/` directory are read only. However, some can be used to adjust settings in the kernel. This is especially true for files in the `/proc/sys/` subdirectory.

To change the value of a virtual file, use the `echo` command and a > symbol to redirect the new value to the file. For example, to change the hostname on the fly, type:

```
echo www.example.com > /proc/sys/kernel/hostname
```

Other files act as binary or boolean switches. Typing `cat /proc/sys/net/ipv4/ip_forward` returns either a `0` or a `1`. `0` indicates that the kernel is not forwarding network packets. Using the `echo` command to change the value of the `ip_forward` file to `1` immediately turns packet forwarding on.

Tip

Another command used to alter settings in the `/proc/sys/` subdirectory is `/sbin/sysctl`. For more information on this command, see Section 5.4 *Using the `sysctl` Command*

For a listing of some of the kernel configuration files available in the `/proc/sys/`, see Section 5.3.9 */proc/sys/*.

5.2. Top-level Files in the proc File System

Below is a list of some of the more useful virtual files in the top-level of the `/proc/` directory.

Note

In most cases, the content of the files listed in this section will not be the same on your machine. This is because much of the information is specific to the hardware on which Red Hat Linux is running.

5.2.1. /proc/apm

This file provides information about the state of the *Advanced Power Management (APM)* system and is used by the apm command. If a system with no battery is connected to an AC power source, this virtual file would look similar to this:

```
1.16 1.2 0x07 0x01 0xff 0x80 -1% -1 ?
```

Running the apm -v command on such a system results in output similar to this:

```
APM BIOS 1.2 (kernel driver 1.16)
AC on-line, no system battery
```

For systems which do not use a battery as a power source, apm is able do little more than put the machine in standby mode. The apm command is much more useful on laptops. For example, the following output is from the command cat /proc/apm on a laptop running Red Hat Linux while plugged into a power outlet:

```
1.16 1.2 0x03 0x01 0x03 0x09 100% -1 ?
```

When the same laptop is unplugged from its power source for a few minutes, the contents of the apm file change to something like this:

```
1.16 1.2 0x03 0x00 0x00 0x01 99% 1792 min
```

The apm -v command will now yield more useful data, such as the following:

```
APM BIOS 1.2 (kernel driver 1.16)
AC off-line, battery status high: 99% (1 day, 5:52)
```

5.2.2. /proc/cmdline

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

```
ro root=/dev/hda2
```

This tells us that the kernel is mounted read-only (signified by (ro)) off of the second partition on the first IDE device (/dev/hda2).

5.2.3. /proc/cpuinfo

This virtual file identifies the type of processor used by your system. The following is an example of the output typical of /proc/cpuinfo:

```
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 5
model           : 9
model name      : AMD-K6(tm) 3D+ Processor
stepping        : 1
cpu MHz         : 400.919
cache size      : 256 KB
fdiv_bug        : no
hlt_bug         : no
```

```
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr mce cx8 pge mmx syscall 3dnow k6_mtrr
bogomips        : 799.53
```

- `processor` — Provides each processor with an identifying number. On systems that have one processor, there will be only a `0`.
- `cpu family` — Authoritatively identifies the type of processor you have in the system. For an Intel-based system, place the number in front of "86" to determine the value. This is particularly helpful for those attempting to identify the architecture of an older system such as a 586, 486, or 386. Because some RPM packages are compiled for each of these particular architectures, this value also helps users determine which packages to install.
- `model name` — Displays the common name of the processor, including its project name.
- `cpu MHz` — Shows the precise speed in megahertz for the processor to the thousandth decimal point.
- `cache size` — Displays 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.

5.2.4. /proc/devices

This file displays the various character and block devices currently configured (not include devices whose modules are not loaded). Below is a sample output from this file:

```
Character devices:
  1 mem
  2 pty
  3 ttyp
  4 ttyS
  5 cua
  7 vcs
 10 misc
 14 sound
 29 fb
 36 netlink
128 ptm
129 ptm
136 pts
137 pts
162 raw
254 iscsictl

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, and is broken into two major sections: Character devices and Block devices.

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

1. Block devices have a buffer available, allowing them to order requests before dealing with them. This is important for 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 a more efficient order. Character devices do not require buffering.
2. Block devices can send and receive information in blocks of a size configured per device. Character devices send data with no preconfigured size.

For more information about devices see /usr/src/linux-2.4/Documentation/devices.txt.

5.2.5. /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
```

5.2.6. /proc/execdomains

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

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

Think of execution domains as the "personality" for a particular operating system. Because other binary formats, such as Solaris, UnixWare, and FreeBSD, can be used with Linux, programmers can change the way the operating system treats particular system calls from these binaries by changing the personality of the task. Except for the PER_LINUX execution domain, different personalities can be implemented as dynamically loadable modules.

5.2.7. /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 which contain frame buffer devices looks similar to this:

```
0 VESA VGA
```

5.2.8. /proc/filesystems

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

```
nodev rootfs
nodev bdev
nodev proc
nodev sockfs
nodev tmpfs
```

```
nodev shm
nodev pipefs
 ext2
nodev ramfs
 iso9660
nodev devpts
 ext3
nodev autofs
nodev binfmt_misc
```

The first column signifies whether the file system is mounted on a block device. Those beginning with `nodev` are not mounted on a device. The second column lists the names of the file systems supported.

The `mount` command cycles through these file systems listed here when one is not specified as an argument.

5.2.9. /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:   80448940          XT-PIC  timer
  1:     174412          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  8:          1          XT-PIC  rtc
 10:     410964          XT-PIC  eth0
 12:      60330          XT-PIC  PS/2 Mouse
 14:    1314121          XT-PIC  ide0
 15:    5195422          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 reports 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` — This is the old AT computer interrupts.

- `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.

5.2.10. /proc/iomem

This file shows you the current map of the system's memory for each physical device:

```
00000000-0009fbff : System RAM
0009fc00-0009ffff : reserved
000a0000-000bffff : Video RAM area
000c0000-000c7fff : Video ROM
000f0000-000fffff : System ROM
00100000-07ffffff : System RAM
  00100000-00291ba8 : Kernel code
  00291ba9-002e09cb : Kernel data
e0000000-e3ffffff : VIA Technologies, Inc. VT82C597 [Apollo VP3]
e4000000-e7ffffff : PCI Bus #01
  e4000000-e4003fff : Matrox Graphics, Inc. MGA G200 AGP
  e5000000-e57fffff : Matrox Graphics, Inc. MGA G200 AGP
e8000000-e8ffffff : PCI Bus #01
  e8000000-e8ffffff : Matrox Graphics, Inc. MGA G200 AGP
ea000000-ea00007f : Digital Equipment Corporation DECchip 21140 [FasterNet]
  ea000000-ea00007f : tulip
ffff0000-ffffffff : reserved
```

The first column displays the memory registers used by each of the different types of memory. The second column lists the kind of memory located within those registers. In particular, this column displays which memory registers are used by the kernel within the system RAM or, if the network interface card has multiple Ethernet ports, the memory registers assigned for each port.

5.2.11. /proc/ioports

The output of `/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
02f8-02ff : serial(auto)
0376-0376 : ide1
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
```

```
0cf8-0cff : PCI conf1
d000-dfff : PCI Bus #01
e000-e00f : VIA Technologies, Inc. Bus Master IDE
  e000-e007 : ide0
  e008-e00f : ide1
e800-e87f : Digital Equipment Corporation DECchip 21140 [FasterNet]
  e800-e87f : tulip
```

The first column gives the I/O port address range reserved for the device listed in the second column.

5.2.12. /proc/isapnp

This file lists installed *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 a 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 and their resource requirements.

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.

5.2.13. /proc/kcore

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

The contents of this file are designed to be examined by a debugger, such as gdb, and is not human readable.

Warning

Do not view the /proc/kcore virtual file. The contents of the file will scramble text output on the terminal. If this file is accidentally viewed, press [Ctrl]-[C] to stop the process then type reset to bring back the command line prompt.

5.2.14. /proc/kmsg

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

5.2.15. /proc/ksyms

This file contains the symbol definitions used by the module tools to dynamically link and bind kernel modules.

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

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

5.2.16. /proc/loadavg

This file provides a look at load average on the processor over time, as well as additional data used by uptime and other commands. A sample /proc/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.

5.2.17. /proc/locks

This file displays the files currently locked by the kernel. The contents of this file contain internal kernel debugging data and can vary tremendously, depending on the use of the system. A sample /proc/locks file for a 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 has its own line which starts with a unique number. 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 or MANDATORY. 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. 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.

5.2.18. /proc/mdstat

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

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

This file remains in the same state as seen above unless a software RAID or md device is present. In that case, view /proc/mdstat to view the current status of md*X* RAID devices.

The /proc/mdstat file below shows a system with its md0 configured as a RAID 1 device, while it is currently re-syncing the disks:

```
Personalities : [linear] [raid1]
read_ahead 1024 sectors
md0: active raid1 sda2[1] sdb2[0] 9940 blocks [2/2] [UU] resync=1% finish=12.3min
algorithm 2 [3/3] [UUU]
unused devices: <none>
```

5.2.19. /proc/meminfo

This is one of the more commonly used files in the /proc/ directory, as it reports a large amount of valuable information about the systems RAM usage.

The following sample /proc/meminfo virtual file is from a system with 256MB of RAM and 384MB of swap space:

```
        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 free, top, and ps commands. In fact, the output of the free command is even similar in appearance to the contents and structure of /proc/meminfo. But by looking directly at /proc/meminfo, more details are revealed:

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

5.2.20. /proc/misc

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

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

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

5.2.21. /proc/modules

This file displays a list of all modules loaded into the kernel. 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:

```
ide-cd                 27008   0 (autoclean)
cdrom                  28960   0 (autoclean) [ide-cd]
soundcore               4100   0 (autoclean)
```

```
agpgart                 31072   0 (unused)
binfmt_misc              5956   1
iscsi                   32672   0 (unused)
scsi_mod                94424   1 [iscsi]
autofs                  10628   0 (autoclean) (unused)
tulip                   48608   1
ext3                    60352   2
jbd                     39192   2 [ext3]
```

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 lists 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`). A module with a line containing a name listed in brackets (`[` or `]`) indicates that the module depends upon another module to be present in order to function.

5.2.22. /proc/mounts

This file provides a list of all mounts used by the system:

```
rootfs / rootfs rw 0 0
/dev/hda2 / ext3 rw 0 0
/proc /proc proc rw 0 0
/dev/hda1 /boot ext3 rw 0 0
none /dev/pts devpts rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
```

The output found here is similar to contents of `/etc/mtab`, except that `/proc/mount` is more up-to-date.

The first column specifies the device that is mounted, the second column revealing the mountpoint, and the third column tells the file system 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`.

5.2.23. /proc/mtrr

This file refers to the current Memory Type Range Registers (MTRRs) in use with the system. If the system architecture supports MTRRs, then the `/proc/mtrr` file may look similar to this:

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

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

Most of the time, this value is properly configured by default. More information on manually configuring this file, can be found online at the following URL: http://web1.linuxhq.com/kernel/v2.3/doc/mtrr.txt.html.

5.2.24. /proc/partitions

Most of the information here is of little importance to the user, except for the following columns:

- major — The major number of the device with this partition. The major number in our example (3) corresponds with the block device ide0 in /proc/devices.
- 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.

5.2.25. /proc/pci

This file contains a full listing of every PCI device on the system. Depending on the number of PCI devices, /proc/pci can be 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, this list also gives detailed IRQ information so an administrator can quickly look for conflicts.

Tip

To get a more readable version of this information, type:

```
/sbin/lspci -vb
```

5.2.26. /proc/slabinfo

This file gives information about memory usage on the *slab* level. Linux kernels greater than version 2.2 use *slab pools* to manage memory above the page level. Commonly used objects have their own slab pools. The following is a portion of a typical `/proc/slabinfo` virtual file:

```
slabinfo - version: 1.1
kmem_cache            64     68    112    2    2    1
nfs_write_data         0      0    384    0    0    1
nfs_read_data          0    160    384    0   16    1
nfs_page               0    200     96    0    5    1
ip_fib_hash           10    113     32    1    1    1
journal_head          51   7020     48    2   90    1
revoke_table           2    253     12    1    1    1
revoke_record          0      0     32    0    0    1
clip_arp_cache         0      0    128    0    0    1
ip_mrt_cache           0      0     96    0    0    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.

Note that *active* in this case means an object is in use.

5.2.27. /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  1139111 3689 234449 84378914
cpu0 1139111 3689 234449 84378914
page 2675248 8567956
swap 10022 19226
intr 93326523 85756163 174412 0 3 3 0 6 0 1 0 428620 0 60330 0 1368304 5538681
disk_io: (3,0):(1408049,445601,5349480,962448,17135856)
ctxt 27269477
btime 886490134
processes 206458
```

Some of the more popular statistics include:

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

5.2.28. /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 files in the /proc/ directory, /proc/swap provides a snapshot of every swap file name, the type of swap space, the total size, and the amount of this space that is in use (in kilobytes). The priority column is useful when multiple swap files are in use. The lower the priority, the more likely the swap file is to be used.

5.2.29. /proc/uptime

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

```
350735.47 234388.90
```

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

5.2.30. /proc/version

This file specifies the version of the Linux kernel and gcc in use, as well as the version of Red Hat Linux installed on the system:

```
Linux version 2.4.20-0.40 (user@foo.redhat.com)(gcc version 3.2.1 20021125
(Red Hat Linux 8.0 3.2.1-1)) #1 Tue Dec 3 20:50:18 EST 2002
```

This information is used for a variety of purposes, including the version data presented when a user logs in.

5.3. Directories in /proc/

Common groups of information concerning the kernel are grouped into directories and subdirectories within the /proc/ directory.

5.3.1. Process Directories

Every /proc/ directory contains a number of directories with numerical names. A listing of them may begins like this:

```
dr-xr-xr-x    3 root      root            0 Feb 13 01:28 1
dr-xr-xr-x    3 root      root            0 Feb 13 01:28 1010
dr-xr-xr-x    3 xfs       xfs             0 Feb 13 01:28 1087
dr-xr-xr-x    3 daemon    daemon          0 Feb 13 01:28 1123
dr-xr-xr-x    3 root      root            0 Feb 13 01:28 11307
```

```
dr-xr-xr-x    3 apache   apache          0 Feb 13 01:28 13660
dr-xr-xr-x    3 rpc      rpc             0 Feb 13 01:28 637
dr-xr-xr-x    3 rpcuser  rpcuser         0 Feb 13 01:28 666
```

These directories are called *process directories*, as they are named after a program's process 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.

Each process directory contains the following files:

- `cmdline` — Contains the command issued when starting the process.
- `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 symbolic 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 symbolic 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:

  ```
  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
  ```
- `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:

  ```
  08048000-08086000 r-xp 00000000 03:03 391479     /usr/sbin/sshd
  08086000-08088000 rw-p 0003e000 03:03 391479     /usr/sbin/sshd
  08088000-08095000 rwxp 00000000 00:00 0
  40000000-40013000 r-xp 00000000 03:03 293205     /lib/ld-2.2.5.so
  40013000-40014000 rw-p 00013000 03:03 293205     /lib/ld-2.2.5.so
  40031000-40038000 r-xp 00000000 03:03 293282     /lib/libpam.so.0.75
  40038000-40039000 rw-p 00006000 03:03 293282     /lib/libpam.so.0.75
  40039000-4003a000 rw-p 00000000 00:00 0
  4003a000-4003c000 r-xp 00000000 03:03 293218     /lib/libdl-2.2.5.so
  4003c000-4003d000 rw-p 00001000 03:03 293218     /lib/libdl-2.2.5.so
  ```
- `mem` — The memory held by the process. This file cannot be read by the user.
- `root` — A link to the root directory of the process.
- `stat` — The status of the process.
- `statm` — The status of the memory in use by the process. Below is a sample `/proc/statm` file:

  ```
  263 210 210 5 0 205 0
  ```

  The seven columns relate to different memory statistics for the process. From left to right, they report the following 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 that are code
5. Number of pages of data/stack
6. Number of library pages
7. Number of dirty pages

- `status` — The status of the process in a more readable form than `stat` or `statm`. Sample output for `sshd` looks similar to this:

```
Name: sshd
State: S (sleeping)
Tgid: 797
Pid: 797
PPid: 1
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 32
Groups:
VmSize:     3072 kB
VmLck:         0 kB
VmRSS:       840 kB
VmData:      104 kB
VmStk:        12 kB
VmExe:       300 kB
VmLib:      2528 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000001000
SigCgt: 0000000000014005
CapInh: 0000000000000000
CapPrm: 00000000fffffeff
CapEff: 00000000fffffeff
```

  The information in this output includes the process name and ID, the state (such as `S (sleeping)` or `R (running)`), user/group ID running the process, and detailed data regarding memory usage.

5.3.1.1. **/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.

5.3.2. **/proc/bus/**

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

The contents of the subdirectories and files available varies greatly on the precise configuration of the system. However, each of the directories for each of the bus types has at least one directory for each bus of that type. These individual bus directories, usually signified with numbers, such as `00`, contain 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:

```
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
```

The `/proc/bus/usb/` directory contains files that track the various devices on any USB buses, as well as the drivers required to use them. The `/proc/bus/usb/001/` directory contains all devices on the first USB bus. By looking at the contents of the `devices` file, one can identify the USB root hub on the motherboard:

```
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
```

5.3.3. `/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 : 01:38:43
rtc_date : 1998-02-13
rtc_epoch : 1900
alarm  : 00:00:00
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`.

5.3.4. `/proc/fs`

This directory shows which file systems are exported. If running an NFS server, typing `cat /proc/fs/nfs/exports` will display the file systems being shared and the permissions granted for those file systems. For more on sharing file system with NFS, see Chapter 9 *Network File System (NFS)*.

5.3.5. `/proc/ide/`

This directory holds 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 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`, provides additional information. The `channel` file provides the channel number, while the `model` tells you the bus type for the channel (such as `pci`).

5.3.5.1. The Device Directory

Within each IDE channel directory is a device directory. The name of the device directory corresponds to the drive letter in the `/dev/` directory. For instance, the first IDE drive on `ide0` would be `hda`.

Note

There is a symbolic link to each of these device directories in the `/proc/ide/` directory.

Each device directory contains a 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 many devices include:

- cache — The device 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
```

5.3.6. /proc/irq/

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

Each IRQ has its own directory, allowing for the individual configuration of each IRQ. 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.

For more information about the /proc/irq/ directory, consult:

```
/usr/src/linux-2.4/Documentation/filesystems/proc.txt
```

5.3.7. /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. Below is a partial listing of these virtual files:

- arp — Contains the kernel's ARP table. This file is particularly useful for connecting a 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 lists 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 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 — If ipchains are in use, this virtual file reveals any current rule.
- ip_fwnames — If ipchains are in use, this virtual file lists all firewall chain names.
- ip_masquerade — Provides a table of masquerading information under ipchains.
- ip_mr_cache — Lists the multicast routing cache.
- ip_mr_vif — Lists 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 — Lists global packet scheduler parameters.
- raw — Lists 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.
- tcp — Contains detailed TCP socket information.
- tr_rif — Lists 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.

5.3.8. /proc/scsi/

This directory is analogous to the /proc/ide/ directory, however, it is only for connected SCSI devices.

The primary file in this directory is /proc/scsi/scsi, which contains a list of every recognized SCSI device. From this listing, the type of device, as well as the model name, vendor, SCSI channel and ID data is available.

For example, if a system contains a SCSI CD-ROM, a tape drive, a hard drive, and a RAID controller, this file would 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
```

Each SCSI driver used by the system has its own directory in `/proc/scsi/`, which contains files specific to each SCSI controller using that driver. So, for the example system just addressed, `aic7xxx` and `megaraid` directories are present, as those two drivers are being utilized. The files in each of the directories typically contain an I/O address range, IRQ information, and statistics for the particular SCSI controller using that driver. Each controller can report a different type and amount of information. The Adaptec AIC-7880 Ultra SCSI host adapter's file in this 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
```

This output reveals 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. For example, this controller is communicating with the CD-ROM at 20 megabytes per second, while the tape drive is only communicating at 10 megabytes per second.

5.3.9. /proc/sys/

The /proc/sys/ directory is different from others in /proc/ because it not only provides information about the system but also allows the system administrator to immediately enable and disable kernel features.

Warning

Use caution when changing settings on a production system using the various files in the /proc/sys/ directory. Changing the wrong setting may render the kernel unstable, requiring a system reboot.

For this reason, be sure the options are valid for that file before attempting to change any value in /proc/sys/.

A good way to determine if a particular file can be configured, or if it is only designed to provide information, is to list it with the -l option at the shell prompt. If the file is writable, it may be used to configure the kernel. For example, a partial listing of /proc/sys/fs looks like this:

```
-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
```

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 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:

```
echo 1 > /proc/sys/kernel/sysrq
```

This will change the value for sysrq from 0 (off) to 1 (on).

The purpose of the System Request Key is to allow immediate input to the kernel through simple key combinations. For example, the System Request Key can be used to immediately shut down or restart a system, sync all mounted file systems, or dump important information to your console. This feature is most useful when using a development kernel or when experiencing system freezes. However, it is considered a security risk for an unattended console and is therefore turned off by default under Red Hat Linux.

Refer to /usr/src/linux-2.4/Documentation/sysrq.txt for more information on the System Request Key.

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:

```
echo 4 2 45 > /proc/sys/kernel/acct
```

Note

Any configuration changes made using the echo command will disappear when the system is restarted. To make configuration changes take effect after the system is rebooted, see Section 5.4 *Using the sysctl Command*.

The /proc/sys/ directory contains several subdirectories controlling different aspects of a running kernel.

5.3.9.1. /proc/sys/dev/

This directory provides parameters for particular devices on the system. Most systems have at least two directories, cdrom and raid. Customized kernels can have other directories, 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. 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. Use the echo command to enable or disable these features.

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 determine the acceleration of RAID devices for I/O intensive tasks, such as resyncing the disks.

5.3.9.2. /proc/sys/fs/

This directory contains an array of options and information concerning various aspects of the file system, 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 display only zeros.
- dquot-nr — Shows the maximum number of cached disk quota entries.
- file-max — 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.
- overflowgid and overflowuid — Defines the fixed group ID and user ID, respectively, for use with file systems 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.

5.3.9.3. /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 file system 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 file system to see if logging should be suspended or resumed.
- cap-bound — Controls the *capability bounding* settings, which provides a list of capabilities for any process on the system. If a capability is not listed here, then no process, no matter how privileged, can do it. The 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.

  For a valid list of values for this virtual file, consult /usr/src/linux-2.4/include/linux/capability.h. More information on capability bounding is available online at the following URL: http://lwn.net/1999/1202/kernel.php3.

- 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 — Configures the system domain name, such as example.com.
- hostname — Configures the system hostname, such as www.example.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 and 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. Any increase in msgmax would increase RAM requirements for the system.
- msgmnb — Sets the maximum number of bytes in a single message queue. The default is 16384.
- msgmni — Sets the maximum number of message queue identifiers. The default is 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 when the system experiences a kernel panic. By default, the value is set to 0, which disables automatic rebooting after a panic.
- printk — This file controls 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 — 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. However, the kernel supports much larger values than this.
- shmmni — Sets the maximum number of shared memory segments for the whole system, in bytes. 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. See Section 5.3.9 */proc/sys/* for details about the System Request Key.
- threads-max — Sets the maximum number of threads to be used by the kernel, with a default value of 2048.
- 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.

5.3.9.4. /proc/sys/net/

This directory contains subdirectories concerning various networking topics. Various configurations at the time of kernel compilation make different directories available here, such as appletalk, ethernet, ipv4, ipx, and ipv6. Within these directories, system administrators are able to adjust the network configuration on a running system.

Given the wide variety of possible networking options available with Linux, only the most common /proc/sys/net/ directories will be discussed.

The /proc/sys/net/core/ directory contains a variety of settings that control the interaction between the kernel and networking layers. The most important of these files are:

- message_burst — The amount of time in tenths of a second required to write a new warning message. This is used to prevent *Denial of Service* (*DoS*) attacks. 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 of a DoS attack is to bombard the targeted system with requests that generate errors and fill up disk partitions with log files or require all of the system's resources to handle the error logging. The settings in message_burst and message_cost are designed to be modified based on the 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 default size in bytes.
- rmem_max — Sets the receive socket buffer maximum size in bytes.
- wmem_default — Sets the send socket buffer default size in bytes.
- wmem_max — Sets the send socket buffer maximum size in bytes.

The /proc/sys/net/ipv4/ directory contains additional networking settings. Many of these settings, used in conjunction with one another, are very useful in preventing attacks on the system or using the system to act as a router.

Caution

An erroneous change to these files may affect remote connectivity to the system.

Here are some of the most important files in the /proc/sys/net/ipv4/ directory:

- icmp_destunreach_rate, icmp_echoreply_rate, icmp_paramprob_rate and icmp_timeexeed_rate — Set the maximum ICMP send packet rate, in 1/100 of a second, to hosts under certain 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 value of 0 allows the kernel to respond, while a value of 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. Setting this file to 1 enables network packet 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 the 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 incoming connection. Default of 3.
- tcp_retries2 — Sets the number of permitted re-transmissions of TCP packets. Default of 15.

The /usr/src/linux-2.4/Documentation/networking/ip-sysctl.txt file contains a complete list of files and options available in the /proc/sys/net/ipv4/ directory.

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

The /proc/sys/net/ipv4/neigh/ directory contains settings for communicating with a host directly connected to the system (called a network neighbor) and also contains different settings for systems more than one hop away.

Routing over IPV4 also has its own directory, /proc/sys/net/ipv4/route/. Unlike conf/ and neigh/, the /proc/sys/net/ipv4/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, 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.

5.3.9.5. /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.
- 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 — When set to the default value of 0 the kernel estimates the amount of memory available and fails requests that are blatantly invalid. Unfortunately, since memory is allocated using a heuristic rather than a precise algorithm, it can sometimes overload the system.

  If overcommit_memory is set to 1, then the potential for system overload is increased, but so is the performance for memory intensive tasks, such as those used by some scientific software.

  For those who desire less risk of over memory commitment, the following two options have been added. Setting overcommit_memory to 2 fails if a memory request adds up to more that half of the physical RAM, plus swap. Setting it to 3 fails if a memory request adds up to more than swap alone can hold.
- pagecache — Controls the amount of memory used by the page cache. The values in pagecache 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.

The /usr/src/linux-2.4/Documentation/sysctl/vm.txt file contains additional information on these various files.

5.3.10. /proc/sysvipc/

This directory contains 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`).

5.3.11. /proc/tty/

This directory contains information about the available and currently used *tty devices* on the system. Originally called *teletype devices*, 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 XFree86. 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 see a 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.

5.4. Using the sysctl Command

The `/sbin/sysctl` command is used to view, set, and automate kernel settings in the `/proc/sys/` directory.

To get a quick overview of all settings configurable in the `/proc/sys/` directory, type the `/sbin/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 information seen if each of the files were viewed individually. The only difference is the file location. The `/proc/sys/net/ipv4/route/min_delay` file is signified by `net.ipv4.route.min_delay`, with the directory slashes replaced by dots and the `proc.sys` portion assumed.

The sysctl command can be use in place of echo to assign values to writable files in the /proc/sys/ directory. For instance instead of using this command:

```
echo 1 > /proc/sys/kernel/sysrq
```

You can use the sysctl command:

```
sysctl -w kernel.sysrq="1"
kernel.sysrq = 1
```

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

Every time the system boots, the init program runs the /etc/rc.d/rc.sysinit script. This script contains a command to execute sysctl using /etc/sysctl.conf to dictate the values passed to the kernel. Any values added to /etc/sysctl.conf will take effect each time the system boots.

5.5. Additional Resources

Below are additional sources of information about proc file system.

5.5.1. Installed Documentation

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

- /usr/src/linux-2.4/Documentation/filesystems/proc.txt — Contains assorted, but limited, information about all aspects of the /proc/ directory.
- /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 file systems (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.
- /usr/src/linux-2.4/ — Some of 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 code.

5.5.2. Useful Websites

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

Chapter 6.

Users and Groups

Control of *users* and *groups* is a core element of Red Hat Linux system administration.

Users can be either people, meaning accounts tied to physical users, or accounts which exist for specific applications to use.

Groups are logical expressions of organization, tying users together for a common purpose. Users within the same group can read, write, or execute files owned by the group.

Each user and group have a unique numerical identification number called a *userid* (*UID*) and a *groupid* (*GID*) respectively.

When a file is created, it is assigned a user and group owner. It is also assigned separate read, write, and execute permissions for the owner, the group, and everyone else. The user and the group to which a file belongs, as well as the access permissions on the file, can be changed by the root user or, in most cases, by the creator of the file.

Proper management of users and groups, and effective management of file permissions are among the most important tasks a system administrator undertakes. For a detailed look at strategies for managing users and groups, refer to the chapter titled *Managing Accounts and Group* in the *Red Hat Linux System Administration Primer*.

6.1. User and Group Management Tools

Managing users and groups can be a tedious task, but Red Hat Linux provides tools and conventions to make their management easier.

The easiest way to manage users and groups is through the graphical application, **User Manager** (`redhat-config-users`). For more information on **User Manager**, refer to the chapter titled *User and Group Configuration* in the *Red Hat Linux Customization Guide*.

The following command line tools can also be used to manage users and groups:

- `useradd`, `usermod`, and `userdel` — Industry-standard methods of adding, deleting and modifying user accounts.
- `groupadd`, `groupmod`, and `groupdel` — Industry-standard methods of adding, deleting, and modifying user groups.
- `gpasswd` — Industry-standard method of administering the `/etc/group` file.
- `pwck`, `grpck` — Tools for the verification of the password, group, and associated shadow files.
- `pwconv`, `pwunconv` — Tools for the conversion to shadow passwords and back to standard passwords.

For an overview of users and group management, see the *Red Hat Linux System Administration Primer*. For a detailed look at command line tools for managing users and groups, see the chapter titled *User and Group Configuration* in the *Red Hat Linux Customization Guide*.

6.2. Standard Users

Table 6-1 lists the standard users configured in the `/etc/passwd` file by an "Everything" installation. The groupid (GID) in this table is the *primary group* for the user. See Section 6.3 *Standard Groups* for a listing of standard groups.

User	UID	GID	Home Directory	Shell
root	0	0	/root	/bin/bash
bin	1	1	/bin	/sbin/nologin
daemon	2	2	/sbin	/sbin/nologin
adm	3	4	/var/adm	/sbin/nologin
lp	4	7	/var/spool/lpd	/sbin/nologin
sync	5	0	/sbin	/bin/sync
shutdown	6	0	/sbin	/sbin/shutdown
halt	7	0	/sbin	/sbin/halt
mail	8	12	/var/spool/mail	/sbin/nologin
news	9	13	/var/spool/news	
uucp	10	14	/var/spool/uucp	/sbin/nologin
operator	11	0	/root	/sbin/nologin
games	12	100	/usr/games	/sbin/nologin
gopher	13	30	/usr/lib/gopher-data	/sbin/nologin
ftp	14	50	/var/ftp	/sbin/nologin
nobody	99	99	/	/sbin/nologin
rpm	37	37	/var/lib/rpm	/bin/bash
vcsa	69	69	/dev	/sbin/nologin
ntp	38	38	/etc/ntp	/sbin/nologin
canna	39	39	/var/lib/canna	/sbin/nologin
nscd	28	28	/	/bin/false
rpc	32	32	/	/sbin/nologin
postfix	89	89	/var/spool/postfix	/bin/true
named	25	25	/var/named	/bin/false
amanda	33	6	var/lib/amanda/	/bin/bash
postgres	26	26	/var/lib/pgsql	/bin/bash
sshd	74	74	/var/empty/sshd	/sbin/nologin
rpcuser	29	29	/var/lib/nfs	/sbin/nologin
nsfnobody	65534	65534	/var/lib/nfs	/sbin/nologin
pvm	24	24	/usr/share/pvm3	/bin/bash
apache	48	48	/var/www	/bin/false
xfs	43	43	/etc/X11/fs	/sbin/nologin
desktop	80	80	/var/lib/menu/kde	/sbin/nologin

User	UID	GID	Home Directory	Shell
gdm	42	42	`/var/gdm`	`/sbin/nologin`
mysql	27	27	`/var/lib/mysql`	`/bin/bash`
webalizer	67	67	`/var/www/html/usage`	`/sbin/nologin`
mailman	41	41	`/var/mailman`	`/bin/false`
mailnull	47	47	`/var/spool/mqueue`	`/sbin/nologin`
smmsp	51	51	`/var/spool/mqueue`	`/sbin/nologin`
squid	23	23	`/var/spool/squid`	`/dev/null`
ldap	55	55	`/var/lib/ldap`	`/bin/false`
netdump	34	34	`/var/crash`	`/bin/bash`
pcap	77	77	`/var/arpwatch`	`/sbin/nologin`
ident	98	98	`/`	`/sbin/nologin`
privoxy	100	101	`/etc/privoxy`	
radvd	75	75	`/`	`/bin/false`
fax	78	78	`/var/spool/fax`	`/sbin/nologin`
wnn	49	49	`/var/lib/wnn`	`/bin/bash`

Table 6-1. Standard Users

6.3. Standard Groups

Table 6-2 lists the standard groups configured by an "Everything" installation. Groups are stored under Red Hat Linux in the `/etc/group` file.

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	
wheel	10	root
mail	12	mail
news	13	news

Group	GID	Members
uucp	14	uucp
man	15	
games	20	
gopher	30	
dip	40	
ftp	50	
lock	54	
nobody	99	
users	100	
rpm	37	rpm
utmp	22	
floppy	19	
vcsa	69	
ntp	38	
canna	39	
nscd	28	
rpc	32	
postdrop	90	
postfix	89	
named	25	
postgres	26	
sshd	74	
rpcuser	29	
nfsnobody	65534	
pvm	24	
apache	48	
xfs	43	
desktop	80	
gdm	42	
mysql	27	
webalizer	67	
mailman	41	
mailnull	47	
smmsp	51	
squid	23	

Group	GID	Members
ldap	55	
netdump	34	
pcap	77	
ident	98	
privoxy	101	
radvd	75	
fax	78	
slocate	21	
wnn	49	

Table 6-2. Standard Groups

6.4. User Private Groups

Red Hat Linux uses a *user private group* (*UPG*) scheme, which makes UNIX groups easier to manage.

A UPG is created whenever a new user is added to the system. A UPG has the same name as the user for which it was created and that user is the only member of the UPG.

UPGs makes it is safe to set default permissions for a newly created file or directory which allow both the user and *that user's group* to make modifications to the file or directory.

The setting which determines what permissions are applied to a newly created file or directory is called a *umask* and is configured in the `/etc/bashrc` file. Traditionally, on UNIX systems the `umask` is set to `022`, which allows only the user who created the file or directory to make modifications. Under this scheme, all other users, *including members of the creator's group* are not allowed to make any modifications. However, under the UPG scheme, this "group protection" is not necessary since every user has their own private group.

6.4.1. Group Directories

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 project'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.

Lets say, for example, that a group of people work on files in the `/usr/lib/emacs/site-lisp/` directory. Some people are trusted to modify the directory but certainly not everyone. So first create an `emacs` group, as in the following command:

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

In order to associate the contents of the directory with the `emacs` group, type:

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

Now, it is possible to add the proper users to the group with the `gpasswd` command:

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

Allow the users to actually create files in the directory with the following command:

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

When a user creates a new file, it is assigned the group of the user's default private group. Next, set the setgid bit, which assigns everything created in the directory the same group permission as the directory itself (`emacs`). Use the following command:

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

At this point, because each user's default umask is 002, all members of the `emacs` group can create and edit files in the `/usr/lib/emacs/site-lisp/` directory without the administrator having to change file permissions every time users write new files.

6.5. Shadow Passwords

In multiuser environments it is very important to use *shadow passwords* (provided by the `shadow-utils` package). Doing so enhances the security of system authentication files. For this reason, the Red Hat Linux installation program enables shadow passwords by default.

The following is a list of advantages shadow passwords have over the old way of storing passwords on UNIX-based systems.

- Improves system security by moving encrypted password hashes from the world-readable `/etc/passwd` file to `/etc/shadow`, which is readable only by the root user.
- Stores information about password aging.
- Allows the use the `/etc/login.defs` file to enforce security policies.

Most utilities provided by the `shadow-utils` package work properly whether or not shadow passwords are enabled. However, since password aging information is stored exclusively in the `/etc/shadow` file, any commands which create or modify password aging information will not work.

Below is a list of commands which do not work without first enabling shadow passwords:

- `chage`
- `gpasswd`
- `/usr/sbin/usermod` `-e` or `-f` options
- `/usr/sbin/useradd` `-e` or `-f` options

redhat

Chapter 7.
The X Window System

While the heart of Red Hat Linux is the kernel, for many users, the face of the operating system is the graphical environment provided by the *X Window System*, also called *X*.

Various windowing environments have existed in the UNIX™ world for decades, predating many of the current mainstream operating systems. Through the years X has become the dominant graphical environment for UNIX-like operating systems.

The graphical environment for Red Hat Linux is supplied by XFree86™, an open source implementation of X. XFree86 is a large scale, rapidly developing project with hundreds of developers around the world. It features a wide degree of support for a variety of hardware devices and architectures and can run on a variety of different operating systems and platforms.

The X Window System uses a client-server architecture. The *X server* listens for connections from *X client* applications via a network or local loopback interface. The server communicates with the hardware, such as the video card, monitor, keyboard, and mouse. X client applications exist in the user-space, creating a *graphical user interface* (*GUI*) for the user and passing user requests to the X server.

7.1. XFree86

Red Hat Linux 9 uses XFree86 version 4.x as the base X Window System, which includes many cutting edge XFree86 technology enhancements such as 3D hardware acceleration support, the XRender extension for anti-aliased fonts, a modular driver based design, and support for modern video hardware and input devices.

Important

> Red Hat Linux no longer provides XFree86 version 3 server packages. Before upgrading to the latest version of Red Hat Linux, be sure the video card is compatible with XFree86 version 4 by checking the Red Hat Hardware Compatibility List located online at http://hardware.redhat.com.

The files related to XFree86 reside primarily in two locations:

`/usr/X11R6/`

: Contains X server and some client applications as well as X header files, libraries, modules, and documentation.

`/etc/X11/`

: Contains configuration files for X client and server applications. This includes configuration files for the X server itself, the older `xfs` font server, the X display managers, and many other base components.

: It is important to note that the configuration file for the newer Fontconfig-based font architecture is `/etc/fonts/fonts.conf` (which obsoletes the `/etc/X11/XftConfig` file). For more on configuring and adding fonts, see Section 7.4 *Fonts*.

Because the XFree86 server performs advanced tasks on a wide array of hardware, it requires detailed configuration. The Red Hat Linux installation program installs and configures XFree86 automatically, unless the XFree86 packages are not selected for installation. However, if the monitor or video card changes, XFree86 will need to be reconfigured. The best way to do this is to use the **X Configuration Tool** (`redhat-config-xfree86`).

To start the **X Configuration Tool** while in an active X session, go to the **Main Menu Button** (on the Panel) => **System Settings** => **Display**. After using the **X Configuration Tool** during an X session, changes will take effect after logging out and logging back in. For more about using the **X Configuration Tool** refer to the chapter titled *Audio, Video, and General Amusement* in the *Red Hat Linux Getting Started Guide*.

In some situations, reconfiguring the XFree86 server may require manually editing its configuration file, `/etc/X11/XF86Config`. For information about the structure of this file, see Section 7.3 *XFree86 Server Configuration Files*.

7.2. Desktop Environments and Window Managers

Once an XFree86 server is running, X client applications can connect to it and create a GUI for the user. A range of GUIs are possible with Red Hat Linux, from the rudimentary *Tab Window Manager* to the highly developed, interactive *GNOME* desktop environment most Red Hat Linux users are familiar with.

To create the latter, more advanced GUI two main classes of X client applications must connect to the XFree86 server: a *desktop environment* and a *window manager*.

7.2.1. Desktop Environments

A desktop environment brings together assorted X clients which, when used together, create a common graphical user environment and development platform.

Desktop environments have advanced features which allow X clients and other running processes to communicate with one another and allow all applications written to work in that environment to perform advanced tasks, such as drag and drop operations.

Red Hat Linux provides two desktop environments:

- *GNOME* — The default desktop environment for Red Hat Linux based on the GTK+ 2 graphical toolkit.
- *KDE* — An alternative desktop environment based on the Qt 3 graphical toolkit.

Both GNOME and KDE have advanced productivity applications, such as word processors, spreadsheets, and Web browsers as well as provide tools to customize the look and feel of the GUI. Additionally, if both the GTK+ 2 and the Qt libraries are present, KDE applications can run in GNOME and visa versa.

For information on the customization of the GNOME and KDE desktop environments, refer to the *Red Hat Linux Getting Started Guide*.

7.2.2. Window Managers

Window managers are X client programs which are either part of a desktop environment or, in some cases, standalone. Their primary purpose is to control the way graphical windows are positioned, resized, or moved. Window managers also control title bars, window focus behavior, and user-specified key and mouse button bindings.

Five window managers are included with Red Hat Linux:

- `kwin` — The *KWin* window manager is the default window manager for the KDE desktop environment. It is an efficient window manager which supports custom themes.

- `metacity` — The *Metacity* window manager is the default window manager for the GNOME desktop environment. It is a simple and efficient window manager which supports custom themes.
- `mwm` — The *Motif* window manager, is a basic, standalone window manager. Since it is designed to be a standalone window manager, it should not be used in conjunction with the GNOME or KDE desktop environments.
- `sawfish` — The *Sawfish* window manager is a full featured window manager which was the default for the GNOME desktop environment until the release of Red Hat Linux 8.0. It can be used either standalone or with a desktop environment.
- `twm` — The minimalist *Tab Window Manager*, which provides the most basic tool set of any of the window managers and can be used either standalone or with a desktop environment. It is installed as part of XFree86.

These window managers can be run without desktop environments to gain a better sense of their differences. To do this, type the `xinit -e` *<path-to-window-manager>* command, where *<path-to-window-manager>* is the location of the window manager binary file. The binary file can be found by typing `which` *<window-manager-name>*.

7.3. XFree86 Server Configuration Files

The XFree86 server is a single binary executable (`/usr/X11R6/bin/XFree86`) that dynamically loads any necessary X server modules at runtime from the `/usr/X11R6/lib/modules/` directory. Some of these modules are automatically loaded by the server, while others are optional and must be specified in the XFree86 server configuration file.

The XFree86 server and associated configuration files are stored in the `/etc/X11/` directory. The configuration file for the XFree86 server is `/etc/X11/XF86Config`. When Red Hat Linux is installed, the configuration files for XFree86 are created using information gathered about the system hardware during the installation process.

7.3.1. `XF86Config`

While there is rarely a need to manually edit `/etc/X11/XF86Config`, it is useful to know about the various sections and optional parameters available, especially when troubleshooting.

7.3.1.1. The Structure

The `/etc/X11/XF86Config` file is comprised of a many different sections which address specific aspects of the system hardware.

Each section begins with a `Section "`*<section-name>*`"` line (where *<section-name>* is the title for the section) and ends with an `EndSection` line. Within each of the sections, are lines containing option names and at least one option value, occasionally seen in quotes.

Lines beginning with a hash mark [#] are not read by the XFree86 server and are used for human-readable comments.

Some options within the `/etc/X11/XF86Config` file accept a boolean switch which turns the feature on or off. Acceptable boolean values are:

- `1`, `on`, `true`, or `yes` — Turns the option on.
- `0`, `off`, `false`, or `no` — Turns the option off.

The following are some of the more important sections ordered as they appear in a typical `/etc/X11/XF86Config` file. More detailed information about the XFree86 server configuration file can be found in the `XF86Config` man page.

7.3.1.2. `ServerFlags`

The optional `ServerFlags` section contains miscellaneous global XFree86 server settings. Any settings in this section may be overridden by options placed in the `ServerLayout` section (refer to Section 7.3.1.3 *`ServerLayout`* for details).

Entries within the `ServerFlags` section are on their own lines and begin with the term `Option` followed by an option enclosed in double quotation marks ["].

The following is a sample `ServerFlags` section:

```
Section "ServerFlags"
      Option "DontZap" "true"
EndSection
```

The following is a list of some of the most useful options:

- `"DontZap" "<boolean>"` — When the value of *`<boolean>`* is set to true, this setting prevents the use of the [Ctrl]-[Alt]-[Backspace] key combination to immediately terminate the XFree86 server.
- `"DontZoom" "<boolean>"` — When the value of *`<boolean>`* is set to true, this setting prevents cycling through configured video resolutions using the [Ctrl]-[Alt]-[Keypad-Plus] and [Ctrl]-[Alt]-[Keypad-Minus] key combinations.

7.3.1.3. `ServerLayout`

The `ServerLayout` section binds together the input and output devices controlled by the XFree86 server. At a minimum, this section must specify one output device and at least two input devices (a keyboard and a mouse).

The following example illustrates a typical `ServerLayout` section:

```
Section  "ServerLayout"
        Identifier     "Default Layout"
        Screen      0  "Screen0" 0 0
        InputDevice    "Mouse0" "CorePointer"
        InputDevice    "Keyboard0" "CoreKeyboard"
EndSection
```

The following entries are commonly used in the `ServerLayout` section:

- `Identifier` — Specifies a unique name for this `ServerLayout` section.
- `Screen` — Specifies the name of a `Screen` section to be used with the XFree86 server. More than one `Screen` options may be present.

  The following is an example of a typical `Screen` entry:

  ```
  Screen      0  "Screen0" 0 0
  ```

  The first number in this example `Screen` entry (`0`) indicates that the first monitor connector or *head* on the video card uses the configuration specified in the `Screen` section with the identifier `"Screen0"`.

  If the video card has more than one head, another `Screen` entry would be necessary with a different number and a different `Screen` section identifier.

  The numbers to the right of `"Screen0"` give the X and Y absolute coordinates for the upper-left corner of the screen (`0 0` by default).
- `InputDevice` — Specifies the name of an `InputDevice` section to be used with the XFree86 server.

There must be at least two `InputDevice` entires: one for the default mouse and one for the default keyboard. The options `CorePointer` and `CoreKeyboard` indicate these are the primary mouse and keyboard.

- `Option "<option-name>"` — An optional entry which specifies extra parameters for the section. Any options listed here override those listed in the `ServerFlags` section.

  Replace `<option-name>` with a valid option listed for this section in the XF86Config man page.

It is possible to create more than one `ServerLayout` section. However, the server will only read the first one to appear unless an alternate `ServerLayout` section is specified as a command line argument.

7.3.1.4. Files

The `Files` section sets paths for services vital to the XFree86 server, such as the font path.

The following example illustrates a typical `Files` section:

```
Section "Files"
        RgbPath       "/usr/X11R6/lib/X11/rgb"
        FontPath      "unix/:7100"
EndSection
```

The following entries are commonly used in the `Files` section:

- `RgbPath` — Specifies the location of the RGB color database. This database defines all valid color names in XFree86 and ties them to specific RGB values.
- `FontPath` — Specifies where the XFree86 server must connect to obtain fonts from the `xfs` font server.

  By default, the `FontPath` is `unix/:7100`. This tells the XFree86 server to obtain font information using UNIX-domain sockets for inter-process communication (IPC) on port 7100.

  See Section 7.4 *Fonts* for more information concerning XFree86 and fonts.
- `ModulePath` — An optional parameter which specifies alternate directories which store XFree86 server modules.

7.3.1.5. Module

The `Module` section specifies which modules from the `/usr/X11R6/lib/modules/` directory the XFree86 server is to load. Modules add additional functionality to the XFree86 server.

The following example illustrates a typical `Module` section:

```
Section "Module"
 Load  "dbe"
 Load  "extmod"
 Load  "fbdevhw"
 Load  "glx"
 Load  "record"
 Load  "freetype"
 Load  "type1"
 Load  "dri"
EndSection
```

7.3.1.6. `InputDevice`

Each `InputDevice` section configures one input device for the XFree86 server. Systems typically have at least two `InputDevice` sections, keyboard and mouse.

The following example illustrates a typical `InputDevice` section for a mouse:

```
Section "InputDevice"
 Identifier  "Mouse0"
 Driver       "mouse"
 Option      "Protocol" "IMPS/2"
 Option      "Device" "/dev/input/mice"
 Option      "Emulate3Buttons" "no"
EndSection
```

The following entries are commonly used in the `InputDevice` section:

- `Identifier` — Specifies a unique name for this `InputDevice` section. This is a required entry.
- `Driver` — Specifies the name of the device driver XFree86 must load for the device.
- `Option` — Specifies necessary options pertaining to the device.

  For a mouse, these options typically include:

  - `Protocol` — Specifies the protocol used by the mouse, such as `IMPS/2`.
  - `Device` — Specifies the location of the physical device.
  - `Emulate3Buttons` — Specifies whether to allow a two button mouse to act like a three button mouse when both mouse buttons are pressed simultaneously.

  Consult the XF86Config man page for a list of valid options for this section.

By default the `InputDevice` section has comments to allow users to configure additional options.

7.3.1.7. `Monitor` section

Each `Monitor` section configures one type of monitor used by the system. While one `Monitor` section is the minimum, additional instances may occur for each monitor type in use with the machine.

The best way to configure a monitor is to configure X during the installation process or by using **X Configuration Tool**. For more about using the **X Configuration Tool** refer to the chapter titled *Audio, Video, and General Amusement* in the *Red Hat Linux Getting Started Guide*.

This example illustrates a typical `Monitor` section for a monitor:

```
Section "Monitor"
 Identifier   "Monitor0"
 VendorName   "Monitor Vendor"
 ModelName    "DDC Probed Monitor - ViewSonic G773-2"
 DisplaySize  320 240
 HorizSync    30.0 - 70.0
 VertRefresh  50.0 - 180.0
EndSection
```

Warning

Be careful if manually editing values in the `Monitor` section of `/etc/X11/XF86Config`. Inappropriate values can damage or destroy the monitor. Consult the monitories documentation for the a listing of safe operating parameters.

The following are commonly entries used in the `Monitor` section:

- `Identifier` — Specifies a unique name for this `Monitor` section. This is a required entry.
- `VendorName` — An optional parameter which specifies the vendor of the monitor.
- `ModelName` — An optional parameter which specifies the monitor's model name.
- `DisplaySize` — An optional parameter which specifies, in millimeters, the physical size of the monitor's picture area.
- `HorizSync` — Specifies the range of horizontal sync frequencies compatible with the monitor in kHz. These values help the XFree86 server determine the validity of built in or specified `Modeline` entries for the monitor.
- `VertRefresh` — Specifies the range of vertical refresh range frequencies supported by the monitor, in kHz. These values help the XFree86 server determine the validity of built in or specified `Modeline` entries for the monitor.
- `Modeline` — An optional parameter which specifies additional video modes for the monitor at particular resolutions, with certain horizontal sync and vertical refresh resolutions. See the `XF86Config` man page for a more detailed explanation of `Modeline` entries.
- `Option "<option-name>"` — An optional entry which specifies extra parameters for the section. Replace *<option-name>* with a valid option listed for this section in the XF86Config man page.

7.3.1.8. `Device`

Each `Device` section configures one video card on the system. While one `Device` section is the minimum, additional instances may occur for each video card installed on the machine.

The best way to configure a video card is to configure X during the installation process or by using **X Configuration Tool**. For more about using the **X Configuration Tool** refer to the chapter titled *Audio, Video, and General Amusement* in the *Red Hat Linux Getting Started Guide*.

The following example illustrates a typical `Device` section for a video card:

```
Section "Device"
 Identifier  "Videocard0"
 Driver      "mga"
 VendorName  "Videocard vendor"
 BoardName   "Matrox Millennium G200"
 VideoRam    8192
        Option      "dpms"
EndSection
```

The following entries are commonly used in the `Device` section:

- `Identifier` — Specifies a unique name for this `Device` section. This is a required entry.
- `Driver` — Specifies which driver the XFree86 server must load in order to utilize the video card. A list of drivers can be found in `/usr/X11R6/lib/X11/Cards`, which is installed with the `hwdata` package.

- `VendorName` — An optional parameter which specifies the vendor of the video card.
- `BoardName` — An optional parameter which specifies the name of the video card.
- `VideoRam` — An optional parameter which specifies the amount of RAM available on the video card in kilobytes. This setting is only necessary for video cards the XFree86 server cannot probe to detect the amount of video RAM.
- `BusID` — An optional entry which specifies the bus location of the video card. This option is only mandatory for systems with multiple cards.
- `Screen` — An optional entry which specifies which monitor connector or head on the video card the `Device` section configures. This option is only useful for video cards with multiple heads.

  If multiple monitors are connected to different heads on the same video card, separate `Device` sections must exist and each of these sections must have a different `Screen` value.

  Values for the `Screen` entry must be an integer. The first head on the video card has a value of `0`. The value for each additional head increases this value by one.
- `Option "<option-name>"` — An optional entry which specifies extra parameters for the section. Replace *<option-name>* with a valid option listed for this section in the XF86Config man page.

  One of the more common options is `"dpms"`, which activates Service Star energy compliance for the monitor.

7.3.1.9. `Screen`

Each `Screen` section binds one video card (or video card head) to one monitor by referencing the `Device` section and the `Monitor` section for each. While one `Screen` section is the minimum, additional instances may occur for each video card and monitor combination present on the machine.

The following example illustrates a typical `Screen` section:

```
Section "Screen"
 Identifier "Screen0"
 Device     "Videocard0"
 Monitor    "Monitor0"
 DefaultDepth     16
 SubSection "Display"
  Depth     24
  Modes    "1280x1024" "1280x960" "1152x864" "1024x768" "800x600" "640x480"
 EndSubSection
 SubSection "Display"
  Depth     16
  Modes    "1152x864" "1024x768" "800x600" "640x480"
 EndSubSection
EndSection
```

The following entries are commonly used in the `Screen` section:

- `Identifier` — Specifies a unique name for this `Screen` section. This is a required entry.
- `Device` — Specifies the unique name of a `Device` section. This is a required entry.
- `Monitor` — Specifies the unique name of a `Monitor` section. This is a required entry.
- `DefaultDepth` — Specifies the default color depth in bits. In the previous example, `16`, which provides thousands of colors, is the default. Multiple `DefaultDepth` entries are permitted, but at least one must be present.

- `SubSection "Display"` — Specifies the screen modes available at a particular color depth. A `Screen` section may have multiple `Display` subsections, but there must be at least one for the color depth specified in the `DefaultDepth` entry.
- `Option "<option-name>"` — An optional entry which specifies extra parameters for the section. Replace *<option-name>* with a valid option listed for this section in the XF86Config man page.

7.3.1.10. DRI

The optional `DRI` section specifies parameters for the *Direct Rendering Infrastructure* (*DRI*). DRI is an interface which allows 3D software applications to take advantage of 3D hardware acceleration capabilities built into most modern video hardware. In addition, DRI can improve 2D performance via hardware acceleration, if supported by the video card's driver.

This section is ignored unless DRI is enabled in the `Module` section.

The following example illustrates a typical `DRI` section:

```
Section "DRI"
        Group        0
        Mode         0666
EndSection
```

Since different video cards use DRI in different ways, do not alter the values for this section without first referring to the `/usr/X11R6/lib/X11/doc/README.DRI` file.

7.4. Fonts

Red Hat Linux uses two methods to manage fonts and display under XFree86. The newer Fontconfig font subsystem simplifies font management and provides advanced display features, such as anti-aliasing. This system is used automatically for applications programmed using the Qt 3 or GTK+ 2 graphical toolkit.

For compatibility, Red Hat Linux includes the original font subsystem, called the core X font subsystem. This system, which is over 15 years old, is based around the *X Font Server* (*xfs*).

This section discusses how to configure fonts for X using both systems.

7.4.1. Fontconfig

The Fontconfig font subsystem allows applications to directly access fonts on the system and use Xft or other rendering mechanisms to render Fontconfig fonts with advanced anti-aliasing. Graphical applications can use the Xft library with Fontconfig to draw text to the screen.

Over time, the Fontconfig/Xft font subsystem will replace the core X font subsystem.

Important

> The Fontconfig font subsystem does not yet work for **OpenOffice.org** and **Abiword**, which use their own font rendering technology.

It is important to note that Fontconfig share the `/etc/fonts/fonts.conf` configuration file, which replaces the `/etc/X11/XftConfig`. The Fontconfig configuration file should not be edited by hand.

Tip

Due to the transition to the new font system, GTK+ 1.2 applications are not affected by any changes made via the **Font Preferences** dialog (accessed by selecting **Main Menu Button** [on the Panel] => **Preferences** => **Font**). For these applications, a font can be configured by adding the following lines to the file `~/.gtkrc.mine`:

```
style "user-font" {

fontset = "<font-specification>"

}

widget_class "*" style "user-font"
```

Replace <*font-specification*> with a font specification in the style used by traditional X applications, such as `-adobe-helvetica-medium-r-normal--*-120-*-*-*-*-*-*`. A full list of core fonts can be obtained by running `xlsfonts` or created interactively using `xfontsel`.

7.4.1.1. Adding Fonts to Fontconfig

Adding new fonts to the Fontconfig subsystem is a straightforward process.

1. To add fonts systemwide, copy the new fonts into the `/usr/share/fonts/local/` directory.

   To add fonts for an individual user, copy the new fonts into the `.fonts/` directory in the user's home directory.

2. Use the `fc-cache` command to update the font information cache, as in the following example:

   ```
   4fc-cache <path-to-font-directory>
   ```

   In this command, replace <*path-to-font-directory*> with the directory containing the new fonts (either `/usr/share/fonts/local/` or `~/.fonts/`).

Tip

Individual users may also install fonts graphically, by browsing to `fonts:///` in Nautilus, and dragging the new font files there.

Important

If the font filename ends with a `.gz` extension, it is compressed and cannot be used until uncompressed. To do this, use the `gunzip` command or double-click the file and drag the font to a directory in **Nautilus**.

7.4.2. Core X Font System

For compatibility, Red Hat Linux still provides the core X font subsystem, which uses the X Font Server (`xfs`) to provide fonts to X client applications.

The XFree86 server looks for a font server specified in the `FontPath` entry under the `Files` section of the `/etc/X11/XF86Config` configuration file. Refer to Section 7.3.1.4 *`Files`* for more information on the `FontPath` entry.

The XFree86 server connects to the `xfs` server on a specified port to acquire font information. For this reason, the `xfs` service must be running in order for X to start. For more on configuring services for a particular runlevel, refer to the chapter titled *Controlling Access to Services* in the *Red Hat Linux Customization Guide*.

7.4.2.1. `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.

The following is a list of the more common options:

- `alternate-servers` — Specifies 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` — Specifies an ordered list of font paths to use. A comma must follow every font path before a new font path can be started in the list.

  Use the string `:unscaled` immediately after the font path to make the unscaled fonts in that path load first. Then specify the entire path again, so that other scaled fonts will also be loaded.
- `client-limit` — Specifies the maximum number of clients the font server will service. The default is `10`.
- `clone-self` — Allows the font server to clone a new version of itself when the `client-limit` is hit. By default, this option is `on`.
- `default-point-size` — Specifies 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 a 12 point font.
- `default-resolutions` — Specifies a list of resolutions supported by the XFree86 server. Each resolution in the list must be separated by a comma.
- `deferglyphs` — Specifies whether to defer loading *glyphs* (the graphic used to visually represent a font). To disable this feature use `none`, to enable this feature for all fonts use `all`, or to turn this this feature on only for 16-bit fonts use `16`.
- `error-file` — Specifies the path and file name of a location where `xfs` errors are logged.
- `no-listen` — Prevents `xfs` from listening to particular protocols. By default, this option is set to `tcp` to prevent `xfs` from listening on TCP ports for security reasons. If using `xfs` to serve fonts over the network, remove this line.
- `port` — Specifies the TCP port that `xfs` will listen on if `no-listen` does not exist or is commented out.
- `use-syslog` — Specifies whether or not to use the system error log.

7.4.2.2. Adding Fonts to `xfs`

To add fonts to the core X font subsystem (`xfs`), follow these steps:

1. If it does not already exist, create a directory called `/usr/share/fonts/local/` using the following command as root:
   ```
   mkdir /usr/share/fonts/local/
   ```
   If creating the `/usr/share/fonts/local/` directory is necessary, it must be added to the `xfs` path using the following command as root:
   ```
   chkfontpath --add /usr/share/fonts/local/
   ```
2. Copy the new font file into the `/usr/share/fonts/local/` directory
3. Update the font information by issuing the following command as root:
   ```
   ttmkfdir -d /usr/share/fonts/local/ -o /usr/share/fonts/local/fonts.scale
   ```
4. Restart the `xfs` font server using the following command as root:
   ```
   service xfs reload
   ```

7.5. Runlevels and XFree86

In most cases, the default installation of Red Hat Linux configures a machine to boot into a graphical login environment, known as runlevel 5. It is possible, however, to boot into the text-only multi-user mode called runlevel 3 and begin an X session from there.

For more information about runlevels, refer to Section 1.4 *SysV Init Runlevels*.

This section reviews how XFree86 starts in both runlevel 3 and runlevel 5.

7.5.1. Runlevel 3

When in runlevel 3, the best way to start an X session is to log in and type `startx`. The `startx` command is a front-end to the `xinit` command which launches the XFree86 server and connects X clients applications to it. Because the user is already logged into the system at runlevel 3, `startx` does not launch a display manager or authenticate users. Refer to Section 7.5.2 *Runlevel 5* for more information about display managers.

When the `startx` command is executed, it searches for a `.xinitrc` file in the user's home directory to define the desktop environment and possibly other X client applications to run. If no `.xinitrc` file is present, it will use the system default `/etc/X11/xinit/xinitrc` file instead.

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 applications.

After setting these options, the `xinitrc` script executes all scripts located in the `/etc/X11/xinit/xinitrc.d/` directory. One important script in this directory is `xinput`, which configures settings such as the default language.

Next, the `xinitrc` script tries to execute `.Xclients` in the user's home directory and turns to `/etc/X11/xinit/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 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, trying GNOME first and then KDE followed by `twm`.

The user is returned to a text mode user session after logging out of X from runlevel 3.

7.5.2. Runlevel 5

When the system boots into runlevel 5 a special X client application, called a display manager is launched. A user must authenticate using the display manager before any desktop environments or window managers are launched.

Depending on the desktop environments installed on the system, three different display managers are available to handle user authentication.

- gdm — The default display manager for Red Hat Linux, gdm allows the user to configure language settings, shutdown, restart or log in to the system.
- kdm — KDE's display manager which allows the user to shutdown, restart or log in to the system.
- xdm — A very basic display manager which only lets the user log in to the system.

When booting into runlevel 5, the prefdm script determines the preferred display manager by referencing the /etc/sysconfig/desktop file. Refer to the /usr/share/doc/initscripts-<*version-number*>/sysconfig.txt file (where <*version-number*> is the version number of the initscripts package) for a listing of options available for this file.

Each of the display managers references 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 ownership of the console to the user. Then, the /etc/X11/xdm/Xsession script runs to accomplish many of the tasks normally performed by the xinitrc script when starting X from runlevel 3, including setting system and user resources, as well as running the scripts in the /etc/X11/xinit/xinitrc.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 (accessed by selecting **Main Menu Button** [on the Panel] => **Preferences** => **More Preferences** => **Sessions**). If the desktop environment 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.

The user is returned to the display manager after logging out of X from runlevel 5.

For more information about how display managers control user authentication, refer to the /usr/share/doc/gdm-<*version-number*>/README (where <*version-number*> is the version number for the gdm package installed) and the xdm man page.

7.6. Additional Resources

There is a large amount of detailed information available about the XFree86 server, the clients that connect to it, and the assorted desktop environments and window managers.

7.6.1. Installed Documentation

- /usr/X11R6/lib/X11/doc/README — Briefly describes the XFree86 architecture and how to get additional information about the XFree86 project as a new user.

- `/usr/X11R6/lib/X11/doc/RELNOTES` — For advanced users that want to read about the latest features available in XFree86.
- `man XF86Config` — Contains information about the XFree86 configuration files, including the meaning and syntax for the different sections within the files.
- `man 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.
- `man Xserver` — Describes the X display server.

7.6.2. 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.
- 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/ — Home of the GNOME project.
- http://www.kde.org/ — Home of the KDE desktop environment.
- http://nexp.cs.pdx.edu/fontconfig/ — Home of the Fontconfig font subsystem for XFree86.

7.6.3. Related Books

- *The Concise Guide to XFree86 for Linux* by Aron Hsiao; Que — Provides an expert's view of the operation of XFree86 on Linux systems.
- *The New XFree86* by Bill Ball; Prima Publishing — Discuses 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 available APIs.
- *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.

II. Network Services Reference

It is possible to deploy a wide variety of network services under Red Hat Linux. This part describes how network interfaces are configured as well as provides details about critical network services such as NFS, Apache HTTP Server, Sendmail, Fetchmail, Procmail, BIND, and LDAP.

Table of Contents

Chapter 8.
Network Interfaces

Under Red Hat Linux, all network communications occur between configured software *interfaces* and physical networking devices connected to the system.

The configuration files for network interfaces, and the scripts to activate and deactivate them, are located in the `/etc/sysconfig/network-scripts/` directory. Although the number and type of interface files can differ from system to system, there are three categories of files that exist in this directory:

- *Interface configuration files*
- *Interface control scripts*
- *Network function files*

The files in each of these categories work together to enable various network devices under Red Hat Linux.

This chapter will explore the relationship between these files and how they are used.

8.1. Network Configuration Files

Before delving into the interface configuration files, let us first itemize the primary configuration files used in network configuration. Understanding the role these files play in setting up the network stack can be helpful when customizing a Red Hat Linux system.

The primary network configuration files are as follows:

- `/etc/hosts` — The main purpose of this file is to resolve hostnames that cannot be resolved any other way. It can also be used to resolve hostnames on small networks with no DNS server. Regardless of the type of network the computer is on, this file should contain a line specifying the IP address of the loopback device (`127.0.0.1`) as `localhost.localdomain`. For more information, see the hosts man page.
- `/etc/resolv.conf` — This file specifies the IP addresses of DNS servers and the search domain. Unless configured to do otherwise, the network initialization scripts populate this file. For more information on this file, see the `resolv.conf` man page.
- `/etc/sysconfig/network` — Specifies routing and host information for all network interfaces. For more information on this file and the directives it accepts, see Section 4.1.23 *`/etc/sysconfig/network`*.
- `/etc/sysconfig/network-scripts/ifcfg-`<*`interface-name`*> — For each network interface on a Red Hat Linux system, there is a corresponding interface configuration script. Each of these files provide information specific to a particular network interface. See Section 8.2 *Interface Configuration Files* for more information on this type of file and the directives it accepts.

Caution

The `/etc/sysconfig/networking/` directory is used by the **Network Administration Tool** (`redhat-config-network`) and its contents should not be edited manually. For more information about configuring network interfaces using the **Network Administration Tool**, see the chapter called *Network Configuration* in the *Red Hat Linux Customization Guide*.

8.2. Interface Configuration Files

Interface configuration files control the software interfaces for individual network devices. As the system boots, it uses these files to determine what interfaces to bring up and how to configure them. These files are usually named ifcfg-<*name*>, where <*name*> refers to the name of the device that the configuration file controls.

8.2.1. Ethernet Interfaces

One of the most common interface files is ifcfg-eth0, which controls the first Ethernet *network interface card* or *NIC* in the system. In a system with multiple NICs, there are multiple ifcfg-eth<*X*> files (where <*X*> is a unique number corresponding to a specific interface). Because each device has its own configuration file, an administrator can control how each interface functions individually.

Below is a sample ifcfg-eth0 file for a system using a fixed IP address:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
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 using DHCP looks quite a bit different, because IP information is provided by the DHCP server:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

The **Network Administration Tool** (redhat-config-network) is an easy way to make changes to the various network interface configuration files (see the chapter titled *Network Configuration* in the *Red Hat Linux Customization Guide* for detailed instructions on using this tool).

However, it is also possible to edit the configuration files for a given network interface by hand.

Below is a listing of the configurable parameters in an Ethernet interface configuration file:

- 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. This directive is deprecated.
- DEVICE=<*name*>, where <*name*> is the name of the physical device (except for dynamically-allocated PPP devices where it is the *logical name*).
- DNS*{1,2}*=<*address*>, where <*address*> is a name server address to be placed in /etc/resolv.conf if the PEERDNS directive is set to yes.
- IPADDR=<*address*>, where <*address*> is the IP address.
- NETMASK=<*mask*>, where <*mask*> is the netmask value.

- `NETWORK=`<*address*>, where <*address*> is the network address. This directive is deprecated.
- `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.

- `PEERDNS=`<*answer*>, where <*answer*> is one of the following:
  - `yes` — Modify `/etc/resolv.conf` if the DNS directive is set. If using DCHP, then `yes` is the default.
  - `no` — Do not modify `/etc/resolv.conf`.
  - `SRCADDR=`<*address*>, where <*address*> is the specified source IP address for outgoing packets.
  - `USERCTL=`<*answer*>, where <*answer*> is one of the following:
    - `yes` — Non-root users are allowed to control this device.
    - `no` — Non-root users are not allowed to control this device.

8.2.2. Dialup Interfaces

If connecting to the Internet via a dialup connection, a configuration file is necessary for the interface.

PPP interface files are named using the following format `ifcfg-ppp`<*X*> (where <*X*> is a unique number corresponding to a specific interface).

The PPP interface configuration file is created automatically when `wvdial`, the **Network Administration Tool** or **Kppp** is used to create a dialup account. The *Red Hat Linux Getting Started Guide* contains instructions for using these GUI-based dialup connection tools. It is also possible to create and edit this file manually.

The following is a typical `ifcfg-ppp0` file:

```
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
```

Serial Line Internet Protocol (SLIP) is another dialup interface, although it is used less frequently. SLIP files have interface configuration file names such as `ifcfg-sl0`.

Other options, not already discussed, that may be used in these files include:

- `DEFROUTE=`<*answer*>, where <*answer*> is one of the following:

  - yes — Set this interface as the default route.
  - no — Do not set this 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 the number of seconds of idle activity before the interface will disconnect itself.
- INITSTRING=<*string*>, where <*string*> is the initialization 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.
- MODEMPORT=<*device*>, where <*device*> is the name of the serial device 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 information. In some dialup situations, setting this to a value of 576 will result in fewer packets dropped and a slight improvement to 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 — Modify the 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 modified.

- 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. This file contains the phone number to be dialed and other important information for the interface.

8.2.3. Other Interfaces

Other common interface configuration files that use these options include the following:

- `ifcfg-lo` — A local *loopback interface* is often used in testing, as well as being used in 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.

> **Warning**
>
> Never edit the loopback interface script, `/etc/sysconfig/network-scripts/ifcfg-lo`, by hand. Doing so can prevent the system from operating correctly.

- `ifcfg-irlan0` — An *infrared interface* allows information between devices, such as a laptop and a printer, to flow over an infrared link. This works in a similar way to an Ethernet device except that it commonly occurs over a peer-to-peer connection.
- `ifcfg-plip0` — A *Parallel Line Interface Protocol (PLIP)* connection works much the same way, except that it utilizes a parallel port.
- `ifcfg-tr0` — *Token Ring* topologies are not as common on *Local Area Networks* (*LANs*) as they once were, having been eclipsed by Ethernet.

8.2.4. Alias and Clone Files

Two lesser-used types of interface configuration files found in the `/etc/sysconfig/network-scripts/` directory are *alias* and *clone* files.

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, an `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 should follow this naming convention, 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 set to `no` if it is not specified, users cannot bring this interface up and down. To give users this ability, create a clone by copying `ifcfg-eth0` to `ifcfg-eth0-user` and add the following line:

```
USERCTL=yes
```

When a user brings up the `eth0` interface with the `ifup eth0-user` command, the configuration options from `ifcfg-eth0` and `ifcfg-eth0-user` are combined. 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 graphical **Network Administration Tool**. For more on using this tool, see the chapter called *Network Configuration* in the *Red Hat Linux Customization Guide*.

8.3. Interface Control Scripts

The interface control scripts activate and deactivated system interfaces. There are two primary interface control scripts, `/sbin/ifdown` and `/sbin/ifup`, that call on control scripts located in the `/etc/sysconfig/network-scripts/` directory.

The `ifdown` and `ifup` interface scripts are symbolic links to scripts in the `/sbin/` directory. When either of these scripts are called, they require the value of the interface to be specified, such as:

```
ifup eth0
Determining IP information for eth0... done.
```

At that point, the `/etc/rc.d/init.d/functions` and `/etc/sysconfig/network-scripts/network-functions` files are used to perform a variety of tasks. See Section 8.4 *Network Function Files* for more information.

After verifying that an interface has been specified and that the user executing the request is allowed to control the interface, the correct script brings the interface up or down. The following are common interface control scripts:

- `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 using 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 an interface is brought up or down.
- `ifdown-ppp` and `ifup-ppp` — Used to bring a PPP interface up or down.
- `ifup-routes` — Adds static routes for a 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 a SLIP interface up or down.

Warning

Removing or modifying any scripts in the `/etc/sysconfig/network-scripts/` directory can cause interface connections to act strangely or fail. Only advanced users should modify scripts related to a network interface.

The easiest way to manipulate all network scripts simultaneously is to use the `/sbin/service` command on the network service (`/etc/rc.d/init.d/network`), as illustrated the following command:

```
/sbin/service network <action>
```

In this example *<action>* can be either `start`, `stop`, or `restart`.

To view a list of configured devices and currently active network interfaces, use the following command:

```
/sbin/service/network status
```

8.4. Network Function Files

Red Hat Linux makes use of several files that contain important common functions used to bring interfaces up and down. Rather than forcing each interface control file to contain these functions, they are grouped together in a few files that are called upon when necessary.

The `/etc/sysconfig/network-scripts/network-functions` file contains the most commonly used IPv4 functions, which are useful to many interface control scripts. These functions include contacting running programs that have requested information about changes in an interface's status, setting hostnames, finding a gateway device, seeing if a particular device is down, and adding a default route.

As the functions required for IPv6 interfaces are different than IPv4 interfaces, a `network-functions-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 the `network-functions` 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 be found in this file.

8.5. Additional Resources

The following resources contain more information concerning network interfaces and can be found in the following locations.

8.5.1. Installed Documentation

- `/usr/share/doc/initscripts-<`*`version`*`>/sysconfig.txt` — A comprehensive guide to available options for network configuration files, including IPv6 options not covered in this chapter.
- `/usr/share/doc/iproute-<`*`version`*`>/ip-cref.ps` — This Postscript™ file contains a wealth of information about the `ip` command, which can be used to manipulate routing tables, among other things. Use the **ghostview** or **kghostview** application to view this file.

Chapter 9.

Network File System (NFS)

NFS (Network File System) allows hosts to mount partitions on a remote system and use them as though they are local file systems. This allows the system administrator to store resources in a central location on the network, providing authorized users continuous access to them.

Two versions of NFS are currently in use. NFS version 2 (NFSv2), which has been around for several years, is widely supported by various operating systems. NFS version 3 (NFSv3) 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 that 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 chapter titled *Network File System (NFS)* in the *Red Hat Linux Customization Guide*.

9.1. Methodology

Linux uses a combination of kernel-level support and continuously running daemon processes to provide NFS file sharing, however, NFS support must be enabled in the Linux kernel in order 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`, the following processes ensure that a given 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 file system.
- `rpc.nfsd` — The process that implements the user-space components 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 use.
- `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 of 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 and should only be used if the server environment requires them.

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 is a random value stored on the server's side and is passed along with RPC requests from the client. The NFS server can be restarted without affecting the clients and the cookie will remain intact.

NFS only performs authentication when a client system attempts to mount a remote file system. To limit access, the NFS server first employs TCP wrappers. TCP wrappers reads the `/etc/hosts.allow` and `/etc/hosts.deny` files to determine if a particular client should be permitted or prevented access to the NFS server. For more information on configuring access controls with TCP wrappers, see Chapter 15 *TCP Wrappers and `xinetd`*.

After the client is granted access by TCP wrappers, the NFS server refers to its configuration file, `/etc/exports`, to determine whether the client can mount any of the exported file systems. 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 client, not a user. Exported file systems can be a accessed by any users on the remote machine.

When configuring the `/etc/exports` file, be very careful when granting read-write permissions (`rw`) for an exported file system.

9.1.1. NFS and `portmap`

NFS relies upon remote procedure calls (RPC) to function. The `portmap` service 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 TCP wrappers' hosts access files (`/etc/hosts.allow` and `/etc/hosts.deny`) to control which remote systems are permitted to use RPC-based services on the server. See Chapter 15 *TCP Wrappers and `xinetd`* for more information. Access control rules for `portmap` will affect all RPC-based services. Alternatively, it is possible to 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 for these rules.

9.1.1.1. Trouble shooting NFS with `portmap`

As `portmap` provides the coordination between RPC services and the port numbers used to communicate with them, it is useful to be able to view the status of 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`, use the `rpcinfo -p` command:

```
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
```

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 this output, it is apparent that various NFS services are 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 (`/sbin/service nfs restart`) will cause those service to correctly register with `portmap` and begin working.

9.2. NFS Server Configuration Files

Configuring a system to share files and directories using NFS is a simple process. Every file system being exported to remote users via NFS, as well as the access rights relating to those file systems, 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 file system by an authorized host.

The `exportfs` command allows the root user to selectively export or unexport directories without restarting the NFS service. When `exportfs` is passed the proper options, the file systems to be exported are written to `/var/lib/nfs/xtab`. Since `rpc.mountd` refers to the `xtab` file when deciding access privileges to a file system, changes to the list of exported file systems 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 file system shares must be written in the same way they are specified in `/etc/exports`. This option is used to test an exported file system before adding it permanently to the list of file systems to be exported.
- `-i` — Ignore `/etc/exports`; only options given from the command line are used to define exported file systems.
- `-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 file systems 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 file systems.

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.

9.2.1. /etc/exports

The /etc/exports file controls which file systems are exported to remote hosts and specifies options. Blank lines are ignored, comments can be made by starting a line with the hash mark (#), and long lines can be wrapped with a backslash (\). Each exported file system should be on its own individual line. Lists of authorized hosts placed after an exported file system 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 use it:

```
/some/directory bob.example.com
/another/exported/directory 192.168.0.3
```

After re-exporting /etc/exports with the /sbin/service nfs reload command, the bob.example.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 file system will not be able to change it. To allow hosts to make changes to the file system, the rw (read-write) option must be specified.
- 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 file system 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. The no_wdelay option turns off this feature, but is only available when using the sync option.
- root_squash — Strips root users connected remotely from having root privileges by assigning them the nobody userid. 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 case, a special user account can be created for remote NFS users to share and specify (anonuid=<*uid-value*>,anongid=<*gid-value*>), where <*uid-value*> is the userid 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 file system 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 hostnames, use the following methods:

- *single host* — Where one particular host is specified with a fully qualified domain name, hostname, or IP address.
- *wildcards* — Where a * or ? character is used to take into account a grouping of fully qualified domain names that match a particular string of letters. Wildcards are not to be used with IP addresses; however, they may accidently work if reverse DNS lookups fail.

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 `*.example.com` as wildcard will allow sales.example.com to access an exported file system, but not bob.sales.example.com. To match both possibilities, as well as sam.corp.example.com, you would have to provide `*.example.com *.*.example.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 file system 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 file system, where users can be added and removed from an NIS group without affecting `/etc/exports`.

Warning

The format of the `/etc/exports` file is very precise, particularly in regards to use of the space character. Remember to always separate exported file systems from hosts and hosts from one another with a space character. However, there should be no other space characters in the file except on comment lines.

For example, the following two lines do not mean the same thing:

```
/home bob.example.com(rw)
/home bob.example.com (rw)
```

The first line allows only users from `bob.example.com` read-write access to the `/home` directory. The second line allows users from `bob.example.com` to mount the directory read-only (the default), but the rest of the world can mount it read-write.

9.3. NFS Client Configuration Files

Any NFS share made available by a server can be mounted using various methods. The share can be manually mounted, using the `mount` command. However, this requires that the root user type the `mount` command every time the system restarts. Two methods of configuring NFS shares to be mounted automatically at boot time include modifying the `/etc/fstab` or using the `autofs` service.

9.3.1. /etc/fstab

Placing a properly formatted line in the `/etc/fstab` file has the same effect as manually mounting the exported file system. The `/etc/fstab` file is read by the `/etc/rc.d/init.d/netfs` script at system startup and any NFS shares listed there will be mounted.

A sample `/etc/fstab` line to mount an NFS export looks like the following:

```
<server>:</path/of/dir> </local/mnt/point> nfs <options> 0 0
```

The <*server-host*> corresponds to hostname, IP address, or fully qualified domain name of the server exporting the file system.

The <*/path/of/directory*> is the path to the exported directory.

The <*/local/mount/point*> specifies where on the local file system 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 file system being mounted.

The <*options*> area specifies mount options for the file system. For example, if the options area states `rw,suid`, the exported file system will be mounted read-write and the user and groupid set by the server will be used. Note that parentheses are not to be used here. For more mount options, see Section 9.3.3 *Common NFS Mount Options*.

9.3.2. autofs

One drawback to using `/etc/fstab` is that, regardless of how infrequently a user may access the NFS mounted file system, the system must dedicate resources to keep that mount in place. This is not a problem with one or two mounts, but when the 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 `automount` utility, which will mount and unmount NFS file systems automatically, saving resources.

The `autofs` script, located in the `/etc/rc.d/init.d/` directory, 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.

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*> element of this line indicates the location of the mount on the local file system. 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-file-system>
```

The <*directory*> refers to the directory within the mount point where the exported file system should be mounted. Much like a standard `mount` command, the host exporting the file system, as well as the file system being exported, are required in the <*host*>:<*exported-file system*> section. To specify particular options to be used when mounting the exported file system, place them in the <*mount-options*> section, separated by commas. For NFS mounts that use `autofs`, place `-fstype=nfs` in the <*mount-options*> section.

While `autofs` configuration files can be used for a variety of mounts to many types of devices and file systems, 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.example.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.example.com system within its exported `/home/` file system. 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 9.3.3 *Common NFS Mount Options*.

9.3.3. Common NFS Mount Options

Beyond mounting a file system 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 used 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 file system is unavailable, or if it should report an error (`soft`).

  If `hard` is specified, the user cannot terminate the process waiting for the NFS communication to resume unless `intr` option is also specified.

  If `soft`, is specified, the user 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 file system. This is useful if the system is mounting a non-Linux file system via NFS that contains incompatible binaries.
- `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 listed on the `mount` man page, including options for mounting non-NFS file systems.

9.4. Securing NFS

NFS works well for sharing entire file systems 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 file system 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 file systems on a server or mounting them on a client. Doing so will minimize NFS security risks and better protect data on the server.

9.4.1. Host Access

NFS controls who can mount an exported file system based on the host making the mount request, not the user that will actually use the file system. Hosts must be given explicit rights to mount the exported file system. Access control is not possible for users, other than file and directory permissions. In other words, once a file system is exported via NFS, any user on any remote host connected to the NFS server can access the shared data. To limit the potential risks, administrators can only allow read-only access or squashing users to a common user and groupid. But these solutions may prevent the NFS share from being used in the way it was originally intended.

Additionally, if an attacker gains control of the DNS server used by the system exporting the NFS file system, 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 exporting NFS shares as the scope of the wildcard may encompass more systems than intended.

For more information on securing NFS, refer to the chapter titled *Server Security* in the *Red Hat Linux Security Guide*.

9.4.2. File Permissions

Once the NFS file system is mounted read-write by a remote host, the only protection each shared file has is its permissions. If two users that share the same userid value mount the same NFS file system, 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. For more on NFS and userid conflicts, refer to the chapter titled *Managing Accounts and Groups* in the *Red Hat Linux System Administration Primer*.

The default behavior when exporting a file system via NFS is to use *root squashing*. This sets the userid of anyone accessing the NFS share as the root user on their local machine to a value of the server's nobody account. Never turn off root squashing.

If exporting an NFS share read-only, consider using the `all_squash` option, which makes every user accessing the exported file system take the userid of the nobody user.

9.5. 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 or mounting NFS shares. Consult the following sources for more information.

9.5.1. Installed Documentation

- `/usr/share/doc/nfs-utils-`<*version-number*>`/` — Replace <*version-number*> with the version number of the NFS package. This directory contains a wealth of information about the NFS implementation for Linux, including a look at various NFS configurations and their impact on file transfer performance.
- `man mount` — Contains a comprehensive look at mount options for both NFS server and client configurations.

- `man fstab` — Gives details for the format of the `/etc/fstab` file used to mount file systems at system boot.
- `man nfs` — Provides detail on NFS-specific file system export and mount options.
- `man exports` — Shows common options used in the `/etc/exports` file when exporting NFS file systems.

9.5.2. 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 file systems and shows, in detail, how NFS communication occurs.

Chapter 10.

Apache HTTP Server

The Apache HTTP Server is a robust, commercial-grade open source Web server developed by the Apache Software Foundation (http://www.apache.org). Red Hat Linux includes the Apache HTTP Server version 2.0 as well as a number of server modules designed to enhance its functionality.

The default configuration file installed with the Apache HTTP Server works without alteration for most situations. This chapter outlines many of the Apache HTTP Server configuration file (`/etc/httpd/conf/httpd.conf`) to aid those who require a custom configuration or need to convert a configuration file from the older Apache HTTP Server 1.3 format.

Warning

> If using the graphical **HTTP Configuration Tool** (`redhat-config-httpd`), *do not* hand edit the Apache HTTP Server's configuration file as the **HTTP Configuration Tool** regenerates this file whenever it is used.
>
> For more information about the **HTTP Configuration Tool**, please refer to the chapter titled *Apache HTTP Server Configuration* in the *Red Hat Linux Customization Guide*.

10.1. Apache HTTP Server 2.0

There are important differences between the Apache HTTP Server version 2.0 and version 1.3 (version 1.3 shipped with Red Hat Linux 7.3 and earlier). This section reviews some of the features of Apache HTTP Server 2.0 and outlines important changes. For instructions on migrating a version 1.3 configuration file to the 2.0 format, refer to Section 10.2 *Migrating Apache HTTP Server 1.3 Configuration Files*.

10.1.1. Features of Apache HTTP Server 2.0

The arrival of Apache HTTP Server 2.0 brings with it a number of new features. Among them are the following:

- *New Apache API* — Modules utilize a new, more powerful set of Application Programming Interfaces (APIs).

  Important

  > Modules built for Apache HTTP Server 1.3 will not work without being ported to the new API. If unsure whether or not a particular module has been ported, consult the developer *before* upgrading.

- *Filtering* — Modules can act as content filters. Refer to Section 10.2.4 *Modules and Apache HTTP Server 2.0* for more on how filtering works.
- *IPv6 Support* — The next generation IP addressing format is supported.
- *Simplified Directives* — A number of confusing directives have been removed while others have been simplified. See Section 10.5 *Configuration Directives in `httpd.conf`* for more information about specific directives.

- *Multilingual Error Responses* — When using *Server Side Include* (*SSI*) documents, customizable error response pages can be delivered in multiple languages.
- *Multiprotocol Support* — Multiple protocols are supported.

A more complete list complete list of changes can be found online at http://httpd.apache.org/docs-2.0/.

10.1.2. Packaging Changes in Apache HTTP Server 2.0

Starting with Red Hat Linux 8.0, the Apache HTTP Server packages were renamed. Also, some related packages were renamed, deprecated, or incorporated into other packages.

Below is a list of the packaging changes:

- The `apache`, `apache-devil` and `apache-manual` packages were renamed `httpd`, `httpd-devel` and `httpd-manual` respectively.
- The `mod_dav` package were incorporated into the `httpd` package.
- The `mod_put` and `mod_roaming` packages were removed, since their functionality is a subset of that provided by `mod_dav`.
- The `mod_auth_any` and `mod_bandwidth` packages were removed.
- The version number for the `mod_ssl` package is now synchronized with the `httpd` package. This means that the `mod_ssl` package for Apache HTTP Server 2.0 has a *lower* version number than `mod_ssl` package for Apache HTTP Server 1.3.

10.1.3. File System Changes in Apache HTTP Server 2.0

The following changes to the file system layout occur when upgrading to Apache HTTP Server 2.0:

- *A new configuration directory, `/etc/httpd/conf.d/`, has been added.* — This new directory is used to store configuration files for individually packaged modules, such as `mod_ssl`, `mod_perl`, and `php`. The server is instructed to load configuration files from this location by the directive `Include conf.d/*.conf` within the Apache HTTP Server configuration file, `/etc/httpd/conf/httpd.conf`.

  Important

  It is vital that this line be inserted when migrating an existing configuration.

- *The `ab` and `logresolve` programs have been moved.* — These utility programs have been moved from the `/usr/sbin/` directory and into the `/usr/bin/` directory. This causes scripts with absolute paths for these binaries to fail.
- *The `dbmmanage` command has been replaced.* — The `dbmmanage` command has been replaced with `htdbm`. See Section 10.2.4.4 *The `mod_auth_dbm` and `mod_auth_db` Modules* for more information.
- *The `logrotate` configuration file has been renamed.* — The `logrotate` configuration file has been renamed from `/etc/logrotate.d/apache` to `/etc/logrotate.d/httpd`.

The next section outlines how to migrate an Apache HTTP Server 1.3 configuration to the new 2.0 format.

10.2. Migrating Apache HTTP Server 1.3 Configuration Files

If upgrading from Red Hat Linux 7.3 or earlier upon which the Apache HTTP Server was already installed, then the new stock configuration file for the Apache HTTP Server 2.0 package is installed as `/etc/httpd/conf/httpd.conf.rpmnew` and the original version 1.3 `httpd.conf` is not touched. It is, of course, entirely up to you whether you use the new configuration file and migrate the old settings to it or use the existing file as a base and modify it to suit; however, some parts of the file have changed more than others and a mixed approach is generally the best. The stock configuration files for both version 1.3 and version 2.0 are divided into three sections. The goal of this guide is to suggest what is hopefully the easiest route.

If the `/etc/httpd/conf/httpd.conf` is a modified version of the default Red Hat Linux version and a saved a copy of the original is available, it may be easiest to invoke the `diff` command, as in the following example:

```
diff -u httpd.conf.orig httpd.conf | less
```

This command highlights any modifications made. If a copy of the original file is not available, extract it from an RPM package using the `rpm2cpio` and `cpio` commands, as in the following example:

```
rpm2cpio apache-<version-number>.i386.rpm | cpio -i --make
```

In the above command, replace *<version-number>* with the version number for the `apache` package.

Finally, it is useful to know that the Apache HTTP Server has a testing mode to check for configuration errors. To use access it, type the following command:

```
apachectl configtest
```

10.2.1. Global Environment Configuration

The global environment section of the configuration file contains directives which affect the overall operation of the Apache HTTP Server, such as the number of concurrent requests it can handle and the locations of the various files. This section requires a large number of changes compared with the others and it is therefore recommended to base this section on the Apache HTTP Server 2.0 configuration file and migrate the old settings into it.

10.2.1.1. Selecting Which Interfaces and Ports To Bind To

The `BindAddress` and `Port` directives no longer exist; their functionality is now provided by a more flexible `Listen` directive.

If `Port 80` was set in the 1.3 version configuration file, change it to `Listen 80` in the 2.0 configuration file. If `Port` was set to some value *other than 80*, then append the port number to the contents of the `ServerName` directive.

For example, the following is a sample Apache HTTP Server 1.3 directive:

```
Port 123
ServerName www.example.com
```

To migrate this setting to Apache HTTP Server 2.0, use the following structure:

```
Listen 123
ServerName www.example.com:123
```

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- `http://httpd.apache.org/docs-2.0/mod/mpm_common.html#listen`
- `http://httpd.apache.org/docs-2.0/mod/core.html#servername`

10.2.1.2. Server-pool Size Regulation

In Apache HTTP Server 2.0, the responsibility for accepting requests and dispatching child-processes to handle them has been abstracted into a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server. There are three MPM modules that ship with version 2.0: `prefork`, `worker`, and `perchild`.

The original Apache HTTP Server 1.3 behavior has been moved into the `prefork` MPM. Currently only the `prefork` MPM is available on Red Hat Linux, although the other MPMs may be made available at a later date.

The `prefork` MPM accepts the same directives as Apache HTTP Server 1.3, so the following directives may be migrated directly:

- `StartServers`
- `MinSpareServers`
- `MaxSpareServers`
- `MaxClients`
- `MaxRequestsPerChild`

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mpm.html

10.2.1.3. Dynamic Shared Object (DSO) Support

There are many changes required here, and it is highly recommended that anyone trying to modify an Apache HTTP Server 1.3 configuration to suit version 2.0 (as opposed to migrating the changes into the version 2.0 configuration) copy this section from the stock Red Hat Linux Apache HTTP Server 2.0 configuration file.

Those who do not want to copy the section from the stock Apache HTTP Server 2.0 configuration should note the following:

- The `AddModule` and `ClearModuleList` directives no longer exist. These directives where used to ensure that modules could be enabled in the correct order. The Apache HTTP Server 2.0 API allows modules to specify their ordering, eliminating the need for these two directives.
- The order of the `LoadModule` lines is no longer relevant.
- Many modules have been added, removed, renamed, split up, or incorporated with each other.
- `LoadModule` lines for modules packaged in their own RPMs (`mod_ssl`, `php`, `mod_perl`, and the like) are no longer necessary as they can be found in the relevant file in the `/etc/httpd/conf.d/` directory.
- The various `HAVE_XXX` definitions are no longer defined.

Important

If modifying the original file, please note that it is of paramount importance that the httpd.conf contains the following directive:

```
Include conf.d/*.conf
```

Omission of this directive will result in the failure of all modules packaged in their own RPMs (such as mod_perl, php, and mod_ssl).

10.2.1.4. Other Global Environment Changes

The following directives have been removed from Apache HTTP Server 2.0's configuration:

- *ServerType* — The Apache HTTP Server can only be run as ServerType standalone making this directive irrelevant.
- *AccessConfig* and *ResourceConfig* — These directives have been removed since they mirror the functionality of the Include directive. If the AccessConfig and ResourceConfig directives are set then replace them with Include directives.

  To ensure that the files are read in the order implied by the older directives the Include directives should be placed at the end of httpd.conf, with the one corresponding to ResourceConfig preceding the one corresponding to AccessConfig. If using the default values, include them explicitly as conf/srm.conf and conf/access.conf files.

10.2.2. Main Server Configuration

The main server configuration section of the configuration file sets up the main server, which responds to any requests that are not handled by a <VirtualHost> definition. Values here also provide defaults for any <VirtualHost> containers defined.

The directives used in this section have changed little between Apache HTTP Server 1.3 and version 2.0. If the main server configuration is heavily customized it may be easier to modify the existing configuration file to suit Apache HTTP Server 2.0. Users with only lightly customized main server sections should migrate their changes into the default 2.0 configuration.

10.2.2.1. UserDir Mapping

The UserDir directive is used to enable URLs such as http://example.com/~bob/ to map to a subdirectory within the home directory of the user bob, such as /home/bob/public_html. A side-effect of this feature allows a potential attacker to determine whether a given username is present on the system. For this reason, the default configuration for Apache HTTP Server 2.0 disables this directive.

To enable UserDir mapping, change the directive in httpd.conf from:

```
UserDir disable
```

to the following:

```
UserDir public_html
```

For more on this topic, refer to the following documentation on the Apache Software Foundation's website, http://httpd.apache.org/docs-2.0/mod/mod_userdir.html#userdir.

10.2.2.2. Logging

The following logging directives have been removed:

- `AgentLog`
- `RefererLog`
- `RefererIgnore`

However, agent and referrer logs are still available using the `CustomLog` and `LogFormat` directives.

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/mod_log_config.html#customlog
- http://httpd.apache.org/docs-2.0/mod/mod_log_config.html#logformat

10.2.2.3. Directory Indexing

The deprecated `FancyIndexing` directive has now been removed. The same functionality is available through the `FancyIndexing` *option* within the `IndexOptions` directive.

The new `VersionSort` option to the `IndexOptions` directive causes files containing version numbers to be sorted in a more natural way. For example, `httpd-2.0.6.tar` appears before `httpd-2.0.36.tar` in a directory index page.

The defaults for the `ReadmeName` and `HeaderName` directives have changed from `README` and `HEADER` to `README.html` and `HEADER.html`.

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/mod_autoindex.html#indexoptions
- http://httpd.apache.org/docs-2.0/mod/mod_autoindex.html#readmename
- http://httpd.apache.org/docs-2.0/mod/mod_autoindex.html#headername

10.2.2.4. Content Negotiation

The `CacheNegotiatedDocs` directive now takes the argument `on` or `off`. Existing instances of `CacheNegotiatedDocs` should be replaced with `CacheNegotiatedDocs on`.

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/mod_negotiation.html#cachenegotiateddocs

10.2.2.5. Error Documents

To use a hard-coded message with the `ErrorDocument` directive, the message should be enclosed in a pair of double quotation marks ["], rather than just preceded by a double quotation mark as required in Apache HTTP Server 1.3.

To migrate an `ErrorDocument` setting to Apache HTTP Server 2.0, use the following structure:

```
ErrorDocument 404 "The document was not found"
```

Note the trailing double quote in the previous example `ErrorDocument` directive.

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/core.html#errordocument

10.2.3. Virtual Hosts Configuration

The contents of all `<VirtualHost>` containers should be migrated in the same way as the main server section as described in Section 10.2.2 *Main Server Configuration*.

Important

Note that SSL/TLS virtual host configuration has been moved out of the main server configuration file and into `/etc/httpd/conf.d/ssl.conf`.

For more on this topic, refer to the chapter titled *Apache HTTP Secure Server Configuration* in the *Red Hat Linux Customization Guide* and the documentation online at the following URL:

- http://httpd.apache.org/docs-2.0/vhosts/

10.2.4. Modules and Apache HTTP Server 2.0

In Apache HTTP Server 2.0, the module system has been changed to allow modules to be chained together or combined in new and interesting ways. *Common Gateway Interface* (*CGI*) scripts, for example, can generate server-parsed HTML documents which can then be processed by `mod_include`. This opens up a tremendous number of possibilities with regards to how modules can be combined to achieve a specific goal.

The way this works is that each request is served by exactly one *handler* module followed by zero or more *filter* modules.

Under Apache HTTP Server 1.3, for example, a PHP script would be handled in its entirety by the PHP module. Under Apache HTTP Server 2.0, the request is initially *handled* by the core module — which serves static files — and is then *filtered* by the PHP module.

Exactly how to use this, and all the other new features of Apache HTTP Server 2.0 for that matter, is beyond the scope of this document; however, the change has ramifications if the `PATH_INFO` directive is used for a document which is handled by a module that is now implemented as a filter, as each contains trailing path information after the true file name. The core module, which initially handles the request, does not by default understand `PATH_INFO` and will return `404 Not Found` errors for requests that contain such information. As an alternative, use the `AcceptPathInfo` directive to coerce the core module into accepting requests with `PATH_INFO`.

The following is an example of this directive:

```
AcceptPathInfo on
```

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/core.html#acceptpathinfo

- http://httpd.apache.org/docs-2.0/handler.html
- http://httpd.apache.org/docs-2.0/filter.html

10.2.4.1. The mod_ssl Module

The configuration for mod_ssl has been moved from httpd.conf into the file /etc/httpd/conf.d/ssl.conf. For this file to be loaded, and hence for mod_ssl to work, the statement Include conf.d/*.conf must be in the httpd.conf as described in Section 10.2.1.3 *Dynamic Shared Object (DSO) Support*.

ServerName directives in SSL virtual hosts must explicitly specify the port number.

For example, the following is a sample Apache HTTP Server 1.3 directive:

```
<VirtualHost _default_:443>
    # General setup for the virtual host
    ServerName ssl.example.name
    ...
</VirtualHost>
```

To migrate this setting to Apache HTTP Server 2.0, use the following structure:

```
<VirtualHost _default_:443>
    # General setup for the virtual host
    ServerName ssl.host.name:443
    ...
</VirtualHost>
```

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/mod_ssl.html
- http://httpd.apache.org/docs-2.0/vhosts/

10.2.4.2. The mod_proxy Module

Proxy access control statements are now placed inside a <Proxy> block rather than a <Directory proxy:>.

The caching functionality of the old mod_proxy has been split out into the following three modules:

- mod_cache
- mod_disk_cache
- mod_file_cache

These generally use the same or similar directives as the older versions of the mod_proxy module.

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/mod_proxy.html

10.2.4.3. The `mod_include` Module

The `mod_include` module is now implemented as a filter and is therefore enabled differently. Refer to Section 10.2.4 *Modules and Apache HTTP Server 2.0* for more about filters.

For example, the following is a sample Apache HTTP Server 1.3 directive:

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

To migrate this setting to Apache HTTP Server 2.0, use the following structure:

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

Note that just as before, the `Options +Includes` directive is still required for the `<Directory>` container or in a `.htaccess` file.

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/mod_include.html

10.2.4.4. The `mod_auth_dbm` and `mod_auth_db` Modules

Apache HTTP Server 1.3 supported two authentication modules, `mod_auth_db` and `mod_auth_dbm`, which used Berkeley Databases and DBM databases respectively. These modules have been combined into a single module named `mod_auth_dbm` in Apache HTTP Server 2.0, which can access several different database formats. To migrate from `mod_auth_db`, configuration files should be adjusted by replacing `AuthDBUserFile` and `AuthDBGroupFile` with the `mod_auth_dbm` equivalents: `AuthDBMUserFile` and `AuthDBMGroupFile`. Also, the directive `AuthDBMType DB` must be added to indicate the type of database file in use.

The following example shows a sample `mod_auth_db` configuration for Apache HTTP Server 1.3:

```
<Location /private/>
  AuthType Basic
  AuthName "My Private Files"
  AuthDBUserFile /var/www/authdb
  require valid-user
</Location>
```

To migrate this setting to version 2.0 of Apache HTTP Server, use the following structure:

```
<Location /private/>
  AuthType Basic
  AuthName "My Private Files"
  AuthDBMUserFile /var/www/authdb
  AuthDBMType DB
  require valid-user
</Location>
```

Note that the `AuthDBMUserFile` directive can also be used in `.htaccess` files.

The `dbmmanage` Perl script, used to manipulate user name and password databases, has been replaced by `htdbm` in Apache HTTP Server 2.0. The `htdbm` program offers equivalent functionality and like `mod_auth_dbm` can operate a variety of database formats; the `-T` option can be used on the command line to specify the format to use.

Table 10-1 shows how to migrate from a DBM-format database to `htdbm` format using `dbmmanage`.

Action	dbmmanage command (1.3)	Equivalent htdbm command (2.0)
Add user to database (using given password)	`dbmmanage authdb add username password`	`htdbm -b -TDB authdb username password`
Add user to database (prompts for password)	`dbmmanage authdb adduser username`	`htdbm -TDB authdb username`
Remove user from database	`dbmmanage authdb delete username`	`htdbm -x -TDB authdb username`
List users in database	`dbmmanage authdb view`	`htdbm -l -TDB authdb`
Verify a password	`dbmmanage authdb check username`	`htdbm -v -TDB authdb username`

Table 10-1. Migrating from `dbmmanage` to `htdbm`

The `-m` and `-s` options work with both `dbmmanage` and `htdbm`, enabling the use of the MD5 or SHA1 algorithms for hashing passwords, respectively.

When creating a new database with `htdbm`, the `-c` option must be used.

For more on this topic, refer to the following documentation on the Apache Software Foundation's website:

- http://httpd.apache.org/docs-2.0/mod/mod_auth_dbm.html

10.2.4.5. The `mod_perl` Module

The configuration for `mod_perl` has been moved from `httpd.conf` into the file `/etc/httpd/conf.d/perl.conf`. For this file to be loaded, and hence for `mod_perl`to work, the statement `Include conf.d/*.conf` must be included in the `httpd.conf` as described in Section 10.2.1.3 *Dynamic Shared Object (DSO) Support*.

Occurences of `Apache::` in the `httpd.conf` must be replaced with `ModPerl::`. Additionally, the manner in which handlers are registered has been changed.

This is a sample Apache HTTP Server 1.3 `mod_perl` configuration:

```
<Directory /var/www/perl>
    SetHandler perl-script
    PerlHandler Apache::Registry
    Options +ExecCGI
</Directory>
```

This is the equivalent `mod_perl` for Apache HTTP Server 2.0:

```
<Directory /var/www/perl>
    SetHandler perl-script
    PerlModule ModPerl::Registry
    PerlHandler ModPerl::Registry::handler
    Options +ExecCGI
</Directory>
```

Most modules for `mod_perl` 1.x should work without modification with `mod_perl` 2.x. XS modules require recompilation and may possibly require minor Makefile modifications.

10.2.4.6. The `mod_python` Module

The configuration for `mod_python` has been moved from `httpd.conf` into the file `/etc/httpd/conf.d/python.conf`. For this file to be loaded, and hence for `mod_python` to work, the statement `Include conf.d/*.conf` must be in the `httpd.conf` as described in Section 10.2.1.3 *Dynamic Shared Object (DSO) Support*.

10.2.4.7. PHP

The configuration for PHP has been moved from `httpd.conf` into the file `/etc/httpd/conf.d/php.conf`. For this file to be loaded, the statement `Include conf.d/*.conf` must be in the `httpd.conf` as described in Section 10.2.1.3 *Dynamic Shared Object (DSO) Support*.

The PHP is now implemented as a filter and must therefore be enabled in a different manor. See Section 10.2.4 *Modules and Apache HTTP Server 2.0* for more about filters.

Under Apache HTTP Server 1.3, PHP was implemented using the following directives:

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Under Apache HTTP Server 2.0, use the following directives instead:

```
<Files *.php>
    SetOutputFilter PHP
    SetInputFilter PHP
</Files>
```

In PHP 4.2.0 and later the default set of predefined variables which are available in the global scope has changed. Individual input and server variables are, by default, no longer placed directly into the global scope. This change may cause scripts to break. Revert to the old behavior by setting `register_globals` to `On` in the file `/etc/php.ini`.

For more on this topic, refer to the following URL for details concerning the global scope changes:

- http://www.php.net/release_4_1_0.php

10.3. After Installation

After installing the `httpd` package, the Apache HTTP Server's documentation is available by installing the `httpd-manual` package and pointing a Web browser to http://localhost/manual/ or browse the Apache documentation available online at http://httpd.apache.org/docs-2.0/.

The Apache HTTP Server's documentation contains a full list and complete descriptions of all configuration options. For the convenience, this chapter provides short descriptions of the configuration directives used by Apache HTTP Server 2.0.

The version of the Apache HTTP Server 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 looking through the configuration files, 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 configured in the `/etc/httpd/conf.d/ssl.conf` file. For more information about virtual hosts, see Section 10.8 *Virtual Hosts*. For information on configuring a secure server virtual host, Section 10.8.1 *Setting Up Virtual Hosts*. For information on setting up an Apache HTTP Secure Server see the chapter titled *Apache HTTP Secure Server Configuration* in the *Red Hat Linux Customization Guide*.

Note

Red Hat, Inc. does not ship FrontPage extensions as the Microsoft™ license prohibits the inclusion of these extensions in a third party product. More information about FrontPage extensions and the Apache HTTP Server can be found online at the following URL: http://www.rtr.com/fpsupport/.

10.4. Starting and Stopping `httpd`

The `httpd` RPM installs the `/etc/rc.d/init.d/httpd` script, which can be accessed using the `/sbin/service` command.

To start the server, as root type:

```
/sbin/service httpd start
```

To stop the server, as root type:

```
/sbin/service httpd stop
```

The `restart` option is a shorthand way of stopping and then starting the Apache HTTP Server.

To restart the server, as root type:

```
/sbin/service httpd restart
```

Note

If running the Apache HTTP Server as a secure server, it is necessary to type the server password whenever using the `start` or `restart` options.

After editing the `httpd.conf` file, however, it is not necessary to explicitly stop and start the server. Instead, use the `reload` option.

To reload the server configuration file, as root type:

```
/sbin/service httpd reload
```

Note

If running the Apache HTTP Server as a secure server, the server password is *not* required when using the `reload` option.

By default, the `httpd` service will *not* start automatically at boot time. To configure the `httpd` service to start up at boot time using an initscript utility, such as `/sbin/chkconfig`, **/sbin/ntsysv**, or the **Services Configuration Tool** program. Refer to the chapter titled *Controlling Access to Services* in *Red Hat Linux Customization Guide* for more information regarding these tools.

Note

If running the Apache HTTP Server as a secure server, the secure server's password is required after the machine boots, unless a specific type of server key file is present.

For information about setting up an Apache HTTP Secure Server see the chapter titled *Apache HTTP Secure Server Configuration* in the *Red Hat Linux Customization Guide*.

10.5. Configuration Directives in `httpd.conf`

The Apache HTTP Server configuration file is `/etc/httpd/conf/httpd.conf`. The `httpd.conf` file is well-commented and mostly self-explanatory. Its default configuration works for most situations; however, it is a good idea to become familiar some of the more important configuration options.

Warning

With the release of Apache HTTP Server 2.0, many configuration options have changed. If migrating a version 1.3 configuration file to the 2.0 format, refer to Section 10.2 *Migrating Apache HTTP Server 1.3 Configuration Files*.

10.5.1. General Configuration Tips

If configuring the Apache HTTP Server, edit `/etc/httpd/conf/httpd.conf` and then either reload, restart, or stop and start the `httpd` process as outlined in Section 10.4 *Starting and Stopping `httpd`*.

Before editing `httpd.conf`, first make a copy the original file. Creating a backup makes it easier to recover from mistakes made while editing the configuration file.

If a mistake is made and the Web server does not work correctly, first review recently edited passages in `httpd.conf` to verify there are no typos.

Next look in the Web server's error log, `/var/log/httpd/error_log`. The error log may not be easy to interpret, depending on the level of experience. If experiencing problems, however, the last entries in the error log should provide useful information about what happened.

Next are a list of short descriptions for many of the directives included in `httpd.conf`. These descriptions are not exhaustive. For more information, refer to the Apache documentation provided in HTML format at http://localhost/manual/ or online at the following URL: http://httpd.apache.org/docs-2.0/.

For more information about `mod_ssl` directives, refer to the documentation included in HTML format at http://localhost/mod/mod_ssl.html or online at the following URL: http://httpd.apache.org/docs-2.0/mod/mod_ssl.html.

10.5.2. `ServerRoot`

The `ServerRoot` is the top-level directory which contains the server's files. Both the secure server and the non-secure server set the `ServerRoot` directive is set to `"/etc/httpd"`.

10.5.3. 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 /etc/httpd/logs/apache_runtime_status is only used as a fall back.

10.5.4. PidFile

PidFile names the file where the server records its process ID (PID). By default the PID is set in /var/run/httpd.pid.

10.5.5. Timeout

Timeout defines, in seconds, the amount of time that the server will wait for receipts and transmissions during communications. Specifically, Timeout defines how long the 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 by default, which is appropriate for most situations.

10.5.6. KeepAlive

KeepAlive sets whether the server will allow more than one request per connection and can be used to prevent any one client from consuming too much of the server's resources.

By default Keepalive is set to off. If Keepalive is set to on and the server becomes very busy, the server can quickly spawn the maximum number of child processes. In this situation, the server will slow down significantly. If Keepalive is enabled, it is a good idea to set the the KeepAliveTimeout low (refer to Section 10.5.8 *KeepAliveTimeout* for more information about the KeepAliveTimeout directive) and monitor the /var/log/httpd/error_log log file on the server. This log reports when the server is running out of child processes.

10.5.7. MaxKeepAliveRequests

This directive sets the maximum number of requests allowed per persistent connection. The Apache Project recommends a high setting, which improves the server's performance. MaxKeepAliveRequests is set to 100 by default, which should be appropriate for most situations.

10.5.8. KeepAliveTimeout

KeepAliveTimeout sets the number of seconds the server will wait after a request has been served before it closes the connection. Once the server receives a request, the Timeout directive applies instead. KeepAliveTimeout is set to 15 seconds by default.

10.5.9. MinSpareServers and MaxSpareServers

The Apache HTTP 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.

The default MinSpareServers value is 5; the default MaxSpareServers value is 20. These default settings should be appropriate in most situations. Be careful not to increase the MinSpareServers to a large number as doing so will create a heavy processing load on the server even when traffic is light.

10.5.10. StartServers

StartServers sets how many server processes are created upon startup. Since the Web server dynamically kills and creates server processes based on traffic load, it is not necessary to change this parameter. The Web server is set to start eight server processes at startup.

10.5.11. MaxClients

MaxClients sets a limit on the total number of server processes, or simultaneously connected clients, that can run at one time. The main purpose of this directive is to keep a runaway Apache HTTP Server from crashing the operating system. For busy servers this value should be set to a high value. The server's default is set to 150. It is not recommended this the value for the MaxClients exceed 256.

10.5.12. 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 the server is 1000.

10.5.13. Listen

The Listen command identifies the ports on which the Web server will accept incoming requests. By default, the Apache HTTP Server is set to listen to port 80 for non-secure Web communications and (in the /etc/httpd/conf.d/ssl.conf which defines any secure servers) to port 443 for secure Web communications.

If the Apache HTTP Server is configured to listen to a port under 1024, the root user to start it. For port 1024 and above, httpd can be started as a regular user.

The Listen directive can also be used to specify particular IP addresses over which the server will accept connections.

10.5.14. Include

Include allows other configuration files to be included at runtime.

The path to these configuration files can be absolute or relative to the ServerRoot.

Important

For the server to use individually packaged modules, such as mod_ssl, mod_perl, and php, the following directive must be in Section 1: Global Environment of httpd.conf:

```
Include conf.d/*.conf
```

10.5.15. LoadModule

LoadModule is used to load in Dynamic Shared Object (DSO) modules. More information on the Apache HTTP Server's DSO support, including exactly how to use the LoadModule directive, can be found in Section 10.7 *Adding Modules*. Note, the load order of the modules is *no longer important* with Apache HTTP Server 2.0. See Section 10.2.1.3 *Dynamic Shared Object (DSO) Support* for more information about Apache HTTP Server 2.0 DSO support.

10.5.16. ExtendedStatus

The ExtendedStatus directive controls whether Apache generates basic (off) or detailed server status information (on), when the server-status handler is called. The Server-status handler is called using Location tags. More information on calling server-status is included in Section 10.5.63 *Location*.

10.5.17. 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 the 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 the secure server. <IfDefine HAVE_SSL> tags also surround the LoadModule and AddModule directives for the ssl_module.

10.5.18. User

The User directive sets the user name of the server process and determines what files the server is allowed to access. Any files inaccessible to this user are also inaccessible to clients connecting to the Apache HTTP Server.

By default User is set to apache.

Note

For security reasons, the Apache HTTP Server will refuse to run as the root user.

10.5.19. Group

Specifies the group name of the Apache HTTP Server processes.

By default Group is set to apache.

10.5.20. ServerAdmin

Set the ServerAdmin directive to the email address of the Web server 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.

By default, ServerAdmin is set to root@localhost.

A common way to set up ServerAdmin is to set it to webmaster@example.com. Then alias webmaster to the person responsible for the Web server in /etc/aliases and run /usr/bin/newaliases.

10.5.21. ServerName

Use ServerName to set a hostname and port number (matching the Listen directive) for the server. The ServerName does not need to match the machine's actual hostname. For example, the Web server may be www.example.com but the server's hostname is actually foo.example.com. The value specified in ServerName must be a valid Domain Name Service (DNS) name that can be resolved by the system — do not make something up.

The following is a sample ServerName directive:

```
ServerName www.example.com:80
```

When specifying a ServerName, be sure the IP address and server name pair are included in the /etc/hosts file.

10.5.22. UseCanonicalName

When set to on, this directive configures the Apache HTTP Server to references itself using the value specified in the ServerName and Port directives. When UseCanonicalName is set to off, the server will instead use the value used by the requesting client when referring to itself.

UseCanonicalName is set to off by default.

10.5.23. DocumentRoot

The DocumentRoot is the directory which contains most of the HTML files which is served in response to requests. The default DocumentRoot for both the non-secure and secure Web servers is the /var/www/html directory. For example, the server might receive a request for the following document:

```
http://example.com/foo.html
```

The server looks for the following file in the default directory:

```
/var/www/html/foo.html
```

To change the DocumentRoot so that it is not shared by the secure and the non-secure Web servers, see Section 10.8 *Virtual Hosts*.

10.5.24. Directory

<Directory /path/to/directory> and </Directory> tags create what is referred to as a *container* and are used to enclose a group of configuration directives meant to apply only to a par-

ticular directory and its subdirectories. Any directive which is applicable to a directory may be used within `<Directory>` tags.

By default, very restrictive parameters are applied to the root directory (`/`), using the `Options` (see Section 10.5.25 *`Options`*) and `AllowOverride` (see Section 10.5.26 *`AllowOverride`*) directives. Under this configuration, any directory on the system which needs more permissive settings has to be explicitly given those settings.

In the default configuration, another `Directory` container is configured for the `DocumentRoot` which assigns less rigid parameters to the directory tree so that the Apache HTTP Server can access the files residing there.

The `Directory` container can be also be used to configure additional `cgi-bin` directories for server-side applications outside of the directory specified in the `ScriptAlias` directive (refer to Section 10.5.44 *`ScriptAlias`* for more information about the `ScriptAlias` directive).

To accomplish this, the `Directory` container must set the `ExecCGI` option for that directory.

For example, if CGI scripts are located in `/home/my_cgi_directory`, add the following `Directory` container to the `httpd.conf` file:

```
<Directory /home/my_cgi_directory>
    Options +ExecCGI
</Directory>
```

Next, the `AddHandler` directive must be uncommented to identify files with the `.cgi` extension as CGI scripts. See Section 10.5.59 *`AddHandler`* for instructions on setting `AddHandler`.

For this to work, permissions for CGI scripts, and the entire path to the scripts, must be set to 0755.

10.5.25. `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 the `DocumentRoot` directory, `Options` is set to include `Indexes` and `FollowSymLinks`. `Indexes` permits the server to generate a directory listing for a directory if no `DirectoryIndex` (for example, `index.html`) is specified. `FollowSymLinks` allows the server to follow symbolic links in that directory.

Note

> `Options` statements from the main server configuration section needs to be replicated to each `VirtualHost` containers individually. Refer to Section 10.5.69 *`VirtualHost`* for more information about `VirtualHost` containers.

10.5.26. `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.

10.5.27. `Order`

The `Order` directive controls the order in which `allow` and `deny` directives are evaluated. The server is configured to evaluate the `Allow` directives before the `Deny` directives for the `DocumentRoot` directory.

10.5.28. `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, and so on. The `DocumentRoot` directory is configured to `Allow` requests from `all`, meaning everyone has access.

10.5.29. `Deny`

`Deny` works just like `Allow`, except it specifies who is denied access. The `DocumentRoot` is not configured to `Deny` requests from anyone by default.

10.5.30. `UserDir`

`UserDir` is the name of the subdirectory within each user's home directory where they should place personal HTML files which are served by the Web server. This directive is set to `disable` by default.

The name for the subdirectory is set to `public_html` in the default configuration. For example, the server might receive the following request:

http://*example.com*/~*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 vary).

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 also work). Files that will be served in users' `public_html` directories must be set to at least 0644.

10.5.31. `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://*example*/*this_directory*/, they get either the `DirectoryIndex` page if it exists or a server-generated directory list. The default for `DirectoryIndex` is `index.html` and the `index.html.var` type map. The server tries to find any one of these files, and returns the first one it finds. If it does not find any of these files and `Options Indexes` is set for that directory, the server generates and returns a listing, in HTML format, of the subdirectories and files within the directory, unless the directory listing feature is turned off.

10.5.32. **AccessFileName**

AccessFileName names the file which the server should use for access control information in each directory. The default is .htaccess.

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.

10.5.33. **CacheNegotiatedDocs**

By default, the 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 CacheNegotiatedDocs is set to on, disables the function and allowing proxy servers to cache documents.

10.5.34. **TypesConfig**

TypesConfig names the file which sets the default list of MIME type mappings (file name extensions 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 10.5.58 *AddType*.

10.5.35. **DefaultType**

DefaultType sets a default content type for the Web server to use for documents whose MIME types cannot be determined. The default is text/plain.

10.5.36. **IfModule**

<IfModule> and </IfModule> tags create a conditional container which are only activated if the specified module is loaded. 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. Or, if an exclamation point [!] appears before the module name, the directives are processed only if the module specified in the <IfModule> tag is *not* loaded.

For more information about Apache HTTP Server modules, refer to Section 10.7 *Adding Modules*.

10.5.37. **HostnameLookups**

HostnameLookups can be set to on, off or double. If HostnameLookups set to on, the server automatically resolves the IP address for each connection. Resolving the IP address means that the server makes one or more connections to a DNS server, adding processing overhead. If HostnameLookups is set to double, the server performs a double-reverse DNS look up adding even more processing overhead.

To conserve resources on the server, HostnameLookups set to off by default.

If hostnames are required in server log files, consider running one of the many log analyzer tools that perform the DNS lookups more efficiently and in bulk when rotating the Web server log files.

10.5.38. ErrorLog

ErrorLog specifies the file where server errors are logged. By default, this directive is set to /var/log/httpd/error_log.

10.5.39. LogLevel

LogLevel sets how verbose the error messages in the error logs are. LogLevel can be set (from least verbose to most verbose) to emerg, alert, crit, error, warn, notice, info or debug. The default LogLevel is warn.

10.5.40. LogFormat

The LogFormat directive configures the format of the various Web server log files. The actual LogFormat used depends on the settings given in the CustomLog directive (see Section 10.5.41 *CustomLog*).

The following are the format options if the CustomLog directive is set to combined:

%h (remote host's IP address or hostname)

Lists the remote IP address of the requesting client. If HostnameLookups is set to on, the client hostname is recorded unless it is not available from DNS.

%l (rfc931)

Not used. A hyphen [-] appears in the log file for this field.

%u (authenticated user)

If authentication was required, lists the user name of the user is recorded. Usually, this is not used, so a hyphen [-] appears in the log file for this field.

%t (date)

Lists the date and time of the request.

%r (request string)

Lists the request string exactly as it came from the browser or client.

%s (status)

Lists the HTTP status code which was returned to the client host.

%b (bytes)

Lists the size of the document.

%\"%{Referer}i\" (referrer)

Lists the URL of the webpage which referred the client host to Web server.

%\"%{User-Agent}i\" (user-agent)

Lists the type of Web browser making the request.

10.5.41. CustomLog

CustomLog identifies the log file and the log file format. By default, the log is recorded to the /var/log/httpd/access_log file.

The default CustomLog format is combined. The following illustrates the combined log file format:

```
remotehost rfc931 user date "request" status bytes referrer user-agent
```

10.5.42. ServerSignature

The ServerSignature directive adds a line containing the Apache HTTP Server server version and the ServerName to any server-generated documents, such as error messages sent back to clients. ServerSignature is set to on by default.

It to also be set to off or to EMail. EMail, adds a mailto:ServerAdmin HTML tag to the signature line of auto-generated responses.

10.5.43. Alias

The Alias setting allows directories outside the DocumentRoot directory to be accessible. Any URL ending in the alias automatically resolves to the alias' path. By default, one alias for an icons directory is already set up. An icons directory can be accessed by the Web server, but the directory is not in the DocumentRoot.

10.5.44. ScriptAlias

The ScriptAlias directive defines where CGI scripts are located. Generally, it is not good practice to leave CGI scripts within the DocumentRoot, where they can potentially be viewed as text documents. For this reason, a special directory outside of the DocumentRoot directory containing server-side executables and scripts is designated by the ScriptAlias directive. This directory is known as a cgi-bin and set to /var/www/cgi-bin/ by default.

It is possible to establish directories for storing executables outside of the cgi-bin directory. For instructions on doing so, refer to Section 10.5.59 *AddHandler* and Section 10.5.24 *Directory*.

10.5.45. Redirect

When a webpage is moved, Redirect can be used to map the file location to a new URL. The format is as follows:

```
Redirect /<old-path>/<file-name> http://<current-domain>/<current-path>/<file-name>
```

In this example, replace *<old-path>* with the old path information for *<file-name>* and *<current-domain>* and *<current-path>* with the current domain and path information for *<file-name>*.

In this example, any requests for *<file-name>* at the old location is automatically redirected to the new location.

For more advanced redirection techniques, use the mod_rewrite module included with the Apache HTTP Server. For more information about configuring the mod_rewrite module, refer to the Apache Software Foundation documentation online at http://httpd.apache.org/docs-2.0/mod/mod_rewrite.html.

10.5.46. IndexOptions

IndexOptions controls the appearance of server generated directing listings, by adding icons, file descriptions, and so on. If Options Indexes is set (see Section 10.5.25 *Options*), the Web server generates a directory listing when the Web server receives an HTTP request for a directory without an index.

First, the Web server looks in the requested directory for a file matching the names listed in the DirectoryIndex directive (usually, index.html). If an index.html file is not found, Apache HTTP Server creates an HTML directory listing of the requested directory. The appearance of this directory listing is controlled, in part, by the IndexOptions directive.

The default configuration turns on FancyIndexing. This means that a user can re-sort a directory listing by clicking on column headers. Another click on the same header will switch from ascending to descending order. FancyIndexing also shows different icons for different files, based upon file extensions.

The AddDescription option, when used in conjunction with FancyIndexing, presents a short description for the file in server generated directory listings.

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 webpages; IconsAreLinks, for making the icons act as part of the HTML link anchor along with the filename and others.

10.5.47. AddIconByEncoding

This directive names icons which are displayed by files with MIME encoding in server generated directory listings. For example, by default, the Web server shows the compressed.gif icon next to MIME encoded x-compress and x-gzip files in server generated directory listings.

10.5.48. AddIconByType

This directive names icons which are displayed next to files with MIME types in server generated directory listings. For example, the server shows the icon text.gif next to files with a mime-type of text, in server generated directory listings.

10.5.49. AddIcon

AddIcon specifies which icon to show in server generated directory listings for files with certain extensions. For example, the Web server is set to show the icon binary.gif for files with .bin or .exe extensions.

10.5.50. DefaultIcon

DefaultIcon specifies the icon displayed in server generated directory listings for files which have no other icon specified. The unknown.gif image file is the default.

10.5.51. AddDescription

When using FancyIndexing as an IndexOptions parameter, the AddDescription directive can be used to display user-specified descriptions for certain files or file types in a server generated directory listings. The AddDescription directive supports listing specific files, wildcard expressions, or file extensions.

10.5.52. ReadmeName

ReadmeName names the file which, if it exists in the directory, is appended to the end of server generated directory listings. The Web server first tries to include the file as an HTML document and then try to include it as plain text. By default, ReadmeName is set to README.html.

10.5.53. HeaderName

HeaderName names the file which, if it exists in the directory, is prepended to the start of server generated directory listings. Like ReadmeName, the server tries to include it as an HTML document if possible or in plain text if not.

10.5.54. IndexIgnore

IndexIgnore lists file extensions, partial file names, wildcard expressions or full filenames. The Web server will not include any files which match any of those parameters in server generated directory listings.

10.5.55. AddEncoding

AddEncoding names filename extensions which should specify a particular encoding type. AddEncoding can also be used to instruct some browsers to uncompress certain files as they are downloaded.

10.5.56. AddLanguage

AddLanguage associates file name extensions with specific languages. This directive is useful for Apache HTTP Servers which serve content in multiple languages based on the client Web browser's language settings.

10.5.57. LanguagePriority

LanguagePriority sets precedence for different languages in case the client Web browser has no language preference set.

10.5.58. AddType

Use the AddType directive to define MIME type and file extension pairs. For example, using PHP4, use the AddType directive to make the Web server recognize with PHP file extensions (.php4, .php3 .phtml .php) as PHP MIME types. The following directive tells the Apache HTTP Server to recognize the .shtml file extension:

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

10.5.59. AddHandler

AddHandler maps file extensions to specific handlers. For example, the cgi-script handler can be matched with the extension .cgi to automatically treat a file ending with .cgi as a CGI script. The following is a sample AddHandler directive for the .cgi extension.

```
AddHandler cgi-script .cgi
```

This directive enables CGIs outside of the `cgi-bin` to function in any directory on the server which has the `ExecCGI` option within the directories container. Refer to Section 10.5.24 *`Directory`* for more information about setting the `ExecCGI` option for a directory.

In addition to CGI scripts, the `AddHandler` directive is used to process server-parsed HTML and image-map files.

10.5.60. `Action`

`Action` specifies a MIME content type and CGI script pair, so that whenever a file of that media type is requested, a particular CGI script is executed.

10.5.61. `ErrorDocument`

The `ErrorDocument` directive associates an HTTP response code with a message or a URL to be sent back to the client. By default, the Web server outputs a simple and usually cryptic error message when an error occurs. The `ErrorDocument` directive forces the Web server to instead output a customized message or redirects the client to a local or external URL.

Important

In order to be valid, the message *must* be enclosed in a pair of double quotes ["].

10.5.62. `BrowserMatch`

The `BrowserMatch` directive allows the server to define environment variables and take appropriate actions based on the User-Agent HTTP header field — which identifies the client's Web browser type. By default, the 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.

10.5.63. `Location`

The `<Location>` and `</Location>` tags create a container in which access control based on URL can be specified.

For instance, to allow people connecting from within the server's domain to see status reports, use the following directives:

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow Deny from all
    Allow from <.example.com>
</Location>
```

Replace `<.example.com>` with the second-level domain name for the Web server.

To provide server configuration reports (including installed modules and configuration directives) to requests from inside the domain, use the following directives:

```
<Location /server-info>
    SetHandler server-info
    Order deny,allow
    Deny from all
    Allow from <.example.com>
</Location>
```

Again, replace <*.example.com*> with the second-level domain name for the Web server.

10.5.64. `ProxyRequests`

To configure the Apache HTTP Server to function as a proxy server, remove the hash marks from the beginning of the `<IfModule mod_proxy.c>` line to load the `mod_proxy` module and set the `ProxyRequests` directive to `On`.

10.5.65. `Proxy`

`<Proxy *>` and `</Proxy>` tags create a container which encloses a group of configuration directives meant to apply only to the proxy server. Many directives which are applicable to a directory may be used within `<Proxy>` tags.

10.5.66. `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 shows the hostname if `ProxyVia` is set to `On`, shows the hostname and the Apache HTTP Server version for `Full`, passes along any Via: lines unchanged for `Off`, and Via: lines are removed for `Block`.

10.5.67. Cache Directives

A number of commented cache directives are supplied by the default Apache HTTP Server configuration file. In most cases, uncommenting these lines by removing the hash mark [#] from the beginning of the line is sufficient. The following, however, is a list of some of the more important cache-related directives.

- `CacheRoot` — Specifies the name of the directory containing cached files. The default `CacheRoot` is the `/var/httpd/proxy/` directory.
- `CacheSize` — Specifies how much space the cache can use in kilobytes. The default `CacheSize` is `5` KB.
- `CacheGcInterval` — Specifies the number of hours which must pass before files in the cache are deleted. The default for `CacheGcInterval` is `4` hours.
- `CacheMaxExpire` — Specifies how long HTML documents are retained (without a reload from the originating Web server) in the cache. The default is `24` hours.
- `CacheLastModifiedFactor` — Specifies 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 `CacheLastModifiedFactor` 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` — Specifies 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 `1` hour.
- `NoCache` — Specifies a list of hosts whose content is not cached.

10.5.68. NameVirtualHost

The NameVirtualHost directive associates an IP address and port number, if necessary, for any name-based virtual hosts. Name-based virtual hosting allows one Apache HTTP Server to serve different domains without using multiple IP addresses.

Note

Name-based virtual hosts *only* work with non-secure HTTP connections. If using virtual hosts with a secure server, use IP address-based virtual hosts instead.

To enable name-based virtual hosting, uncomment the NameVirtualHost configuration directive and add the correct IP address. Then add more VirtualHost containers for each virtual host.

10.5.69. VirtualHost

<VirtualHost> and </VirtualHost> tags create a container outlining the characteristics of a virtual host. The <VirtualHost> container accepts most configuration directives.

A set of commented VirtualHost container is provided in httpd.conf, which illustrates the minimum set of configuration directives necessary for each virtual host. Refer to Section 10.8 *Virtual Hosts* for more information about virtual hosts.

Note

All SSL virtual host containers have been moved into the file /etc/httpd/conf.d/ssl.conf.

10.5.70. SSL Configuration Directives

The SSL directives in /etc/httpd/conf.d/ssl.conf file can be configured to enable secure Web communications using SSL and TLS.

10.5.70.1. SetEnvIf

SetEnvIf sets environment variables based on the headers of incoming secure connections. In the supplied /etc/httpd/conf.d/ssl.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.

For more information on SSL directives, direct a Web browser to either of the following addresses:

- http://localhost/manual/mod/mod_ssl.html
- http://httpd.apache.org/docs-2.0/mod/mod_ssl.html

For information about setting up an Apache HTTP Secure Server see the chapter titled *Apache HTTP Secure Server Configuration* in the *Red Hat Linux Customization Guide*.

Note

In most cases, the SSL directives are configured appropriately as installed. Be cautious when altering Apache HTTP Secure Server directives as misconfiguration can lead to security vulnerabilities.

10.6. Default Modules

The Apache HTTP Server is distributed with a number of modules. By default the following modules are installed and enabled with the `httpd` package on Red Hat Linux:

```
mod_access
mod_auth
mod_auth_anon
mod_auth_dbm
mod_auth_digest
mod_include
mod_log_config
mod_env
mod_mime_magic
mod_cern_meta
mod_expires
mod_headers
mod_usertrack
mod_unique_id
mod_setenvif
mod_mime
mod_dav
mod_status
mod_autoindex
mod_asis
mod_info
mod_cgi
mod_dav_fs
mod_vhost_alias
mod_negotiation
mod_dir
mod_imap
mod_actions
mod_speling
mod_userdir
mod_alias
mod_rewrite
mod_proxy
mod_proxy_ftp
mod_proxy_http
mod_proxy_connect
```

Additionally, the following modules are available by installing additional packages:

```
mod_auth_mysql
mod_auth_pgsql
mod_perl
mod_python
mod_ssl
php
squirrelmail
```

10.7. Adding Modules

The Apache HTTP Server supports *Dynamically Shared Objects* (*DSO*s) or modules, which can easily be loaded at runtime as necessary.

The Apache Project provides complete DSO Documentation at online http://httpd.apache.org/docs-2.0/dso.html. Or if the `http-manual` package is installed, documentation about DSOs can be found at http://localhost/manual/mod/.

For the Apache HTTP Server to use a DSO, it must be specified in a `LoadModule` directive within in `/etc/httpd/conf/httpd.conf`; if the module is provided by a separate package, the line must appear within the modules configuration file in the `/etc/httpd/conf.d/` directory. Refer to Section 10.5.15 *`LoadModule`* for more about the `LoadModule` directive.

If adding or deleting modules from `http.conf`, Apache HTTP Server must be reloaded or restarted, as covered in Section 10.4 *Starting and Stopping `httpd`*.

If creating a new module, first install the `httpd-devel` package as it contains the include files, the header files, as well as the *APache eXtenSion* (`/usr/sbin/apxs`) application, which uses the include files and the header files to compile DSOs.

After writing a module, use `/usr/sbin/apxs` to compile the module sources outside the Apache source tree. For more information about using the `/usr/sbin/apxs` command, refer to the the Apache documentation online at http://httpd.apache.org/docs-2.0/dso.html and the `apxs`, man page.

Once compiled, put the module in the `/usr/lib/httpd/` directory. Then add a `LoadModule` line to the `httpd.conf`, using the following structure:

```
LoadModule <module-name> <path/to/module.so>
```

In the above example, change *`<module-name>`* to the name of the module and *`<path/to/module.so>`* to the path to the DSO.

10.8. Virtual Hosts

The Apache HTTP Server's built in virtual hosting allows the server to serve different information based on which IP address, hostname, or port is being requested. A complete guide to using virtual hosts is available online at http://httpd.apache.org/docs-2.0/vhosts/.

10.8.1. Setting Up Virtual Hosts

To create a name-based virtual host, it is best use the virtual host container provided in `httpd.conf` as an example.

The virtual host example read as follows:

```
#NameVirtualHost *
#
#<VirtualHost  *>
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example.com
#    ErrorLog logs/dummy-host.example.com-error_log
#    CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>
```

To activate name-based virtual hosting, uncomment the `NameVirtualHost` line by removing the hash mark (`#`) and replace the asterisk (`*`) with the IP address assigned to the machine.

Next, configure a virtual host, by uncommenting and customizing the `<VirtualHost>` container.

On the <VirtualHost> line, change the asterisk (*) to the server's IP address. Change the ServerName to a *valid* DNS name assigned to the machine, and configure the other directives as necessary.

The <VirtualHost> container is highly customizable and accepts almost every directive available within the main server configuration.

Tip

If configuring a virtual host to listen on a non-default port, that port must be added to the Listen directive in the global settings section of the /etc/httpd/conf/http.conf file.

To activate a newly created virtual host the Apache HTTP Server must be reloaded or restarted. Refer to Section 10.4 *Starting and Stopping* `httpd` for instructions on doing this.

Comprehensive information about creating and configuring both name-based and IP address-based virtual hosts is provided online at http://httpd.apache.org/docs-2.0/vhosts/.

10.8.2. The Secure Web Server Virtual Host

By default, the Apache HTTP Server is configured as both a non-secure and a secure server. Both the non-secure and secure servers use the same IP address and host name, but listen on different ports: 80 and 443 respectively. This enables both non-secure and secure communications to take place simultaneously.

One aspect of SSL enhanced HTTP transmissions are that they are more resource intensive than the standard HTTP protocol, so a secure server cannot serve as many pages per second. For this reason it is often a good idea to minimize the information available from the secure server, especially on a high traffic Web site.

Important

Do not use name-based virtual hosts in conjunction with a secure Web server as the SSL handshake occurs before the HTTP request identifies the appropriate name-based virtual host. Name-based virtual hosts only work with the non-secure Web server.

The configuration directives for the secure server are contained within virtual host tags in the /etc/httpd/conf.d/ssl.conf file.

By default, both the secure and the non-secure Web servers share the same DocumentRoot. It is recommended that the DocumentRoot be different for the secure Web server.

To stop the non-secure Web server from accepting connection comment the line in httpd.conf which reads Listen 80 by placing a hash mark at the beginning of the line. When finished it will look like the following example:

```
#Listen 80
```

For more information on configuring an SSL enhanced Web server, refer to the chapter titled *Apache HTTP Secure Server Configuration* in the *Red Hat Linux Customization Guide*. For advanced configuration tips, refer to the Apache Software Foundation documentation available online at the following URLs:

- http://httpd.apache.org/docs-2.0/ssl/.

- http://httpd.apache.org/docs-2.0/vhosts/

10.9. Additional Resources

To learn more about the Apache HTTP Server, refer to the following resources.

10.9.1. Useful Websites

- http://httpd.apache.org — The official website for the Apache HTTP Server with documentation on all the directives and default modules.
- http://www.modssl.org — The official website for `mod_ssl`.
- http://www.apacheweek.com — A comprehensive online online weekly about all things Apache.

10.9.2. 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 the Apache HTTP Server at compilation, configuration, and run time. This book is available online at http://www.apacheref.com/.
- *Professional Apache* by Peter Wainwright; Wrox Press Ltd — *Professional Apache* is from Wrox Press Ltd's "Programmer to Programmer" series and is aimed at both experienced and novice Web server administrators.
- *Administering Apache* by Mark Allan Arnold; Osborne Media Group — This book is targeted at Internet Service Providers who aim to provide more secure services.
- *Apache Server Unleashed* by Richard Bowen, et al; SAMS BOOKS — An encyclopedic source for the Apache HTTP Server.
- *Apache Pocket Reference* by Andrew Ford, Gigi Estabrook; O'Reilly — This is the latest addition to the O'Reilly Pocket Reference series.

Chapter 11. Email

The birth of electronic mail (*email*) occurred in the early 1960s. The mailbox was a file in a user's home directory that was readable only by that user. Primitive mail applications appended new text messages to the bottom of the file, and the user had to wade through the constantly growing file to find any particular message. This system was only capable of sending messages to users on the same system.

The first actual network transfer of an electronic mail message file took place in 1971 when a computer engineer named Ray Tomlinson sent a test message between two machines via ARPANET — the precursor to the Internet. Communication via email soon became very popular, comprising 75 percent of ARPANET's traffic in less than two years.

Today, email system based on standardized network protocols have evolved into some of the most widely used services on the Internet. Red Hat Linux offers many advanced applications to serve and access email.

This chapter reviews modern email protocols in use today and some of the programs designed to send and receive email.

11.1. Email Protocols

Today, email is delivered using a client/server architecture. An email message is created using mail client program. This program then sends the message to a server. The server then forwards the massage to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

11.1.1. Mail Transport Protocols

Mail delivery from a client application to the server, and from an originating server to the destination server is handled by the *Simple Mail Transfer Protocol* (*SMTP*) .

11.1.1.1. SMTP

The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. In order to send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Red Hat Linux, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail.

One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or *spam* possible. Modern SMTP servers attempt to minimize this behavior by allowing only known hosts access to the SMTP server. Those servers that do not impose such restrictions are called *open relay* servers.

Red Hat Linux uses Sendmail (`/usr/sbin/sendmail`) as its default SMTP program. However, a simpler mail server application called Postfix (`/usr/sbin/postfix`) is also available.

11.1.2. Mail Access Protocols

There are two primary protocols used by email client applications to retrieve email from mail servers: the *Post Office Protocol* (*POP*) and the *Internet Message Access Protocol* (*IMAP*).

Unlike SMTP, both of these protocols require connecting clients to authenticate using a username and password. By default, passwords for both protocols are passed over the network unencrypted.

11.1.2.1. POP

The default POP server under Red Hat Linux is `/usr/sbin/ipop3d` and is provided by the `imap` package. When using a POP server, email messages are downloaded by email client applications. By default, most POP email clients are automatically configured to delete the message on the email server after it has been successfully transferred, however this setting usually can be changed.

POP is fully compatible with important Internet messaging standards, such as *Multipurpose Internet Mail Extensions* (*MIME*), which allow for email attachments.

POP works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections, POP requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments.

The most current version of the standard POP protocol is POP3.

There are, however a variety of lesser-used POP protocol variants:

- *APOP* — POP3 with MDS authentication. An encoded hash of the user's password is sent from the email client to the server rather then sending an unencrypted password.
- *KPOP* — POP3 with Kerberos authentication. See Chapter 17 *Kerberos* for more information.
- *RPOP* — POP3 with RPOP authentication. This uses a per-user ID, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so RPOP is no more secure than standard POP.

For added security, it is possible to use *Secure Socket Layer* (*SSL*) encryption for client authentication and data transfer sessions. This can be enabled by using the `ipop3s` service, or by using the `/usr/sbin/stunnel` program. Refer to Section 11.5.1 *Securing Communication* for more information.

11.1.2.2. IMAP

The default IMAP server under Red Hat Linux is `/usr/sbin/imapd` and is provided by the `imap` package. When using an IMAP mail server, email messages remain on the server where users can read or delete them. IMAP also allows client applications to create, rename, or delete mail directories on the server to organize and store email.

IMAP is particularly useful for those who access their email using multiple machines. The protocol is also convenient for users connecting to the mail server via a slow connection, because only the email header information is downloaded for messages until opened, saving bandwidth. The user also has the ability to delete messages without viewing or downloading them.

For convenience, IMAP client applications are capable of caching copies of messages locally, so the user can browse previously read messages when not directly connected to the IMAP server.

IMAP, like POP, is fully compatible with important Internet messaging standards, such as MIME, which allow for email attachments.

For added security, it is possible to use *SSL* encryption for client authentication and data transfer sessions. This can be enabled by using the `imaps` service, or by using the `/usr/sbin/stunnel` program. Refer to Section 11.5.1 *Securing Communication* for more information.

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 online at http://www.imap.org/products/longlist.htm.

11.2. Email Program Classifications

In general, all email applications fall into at least one of three classifications. Each classification plays 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 one is important for ensuring that email arrives at the correct destination.

11.2.1. Mail Transfer Agent

A *Mail Transfer Agent* (*MTA*) transfers email messages between hosts using SMTP. A message may involve several MTAs as it moves to its intended destination.

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 is quite complicated. In addition, due to problems from spam, use of a particular MTA is usually restricted by the MTA's configuration or by the lack of access to the MTA's network.

Many modern email client programs can act as an MTA when sending email. However, this action should not be confused with the role of a true MTA. The sole reason email client programs are capable of sending out email (like an MTA) is because the host running the application does not have its own MTA. This is particularly true for email client programs on non-Unix-based operating systems. However, these client programs only send outbound messages to an MTA they are authorized to use and do not directly deliver the message to the intended recipient's email server.

Since Red Hat Linux installs two MTAs, Sendmail and Postfix, email client programs are often not required to act as an MTA. Red Hat Linux also includes a special purpose MTA called Fetchmail.

For more information on Sendmail and Fetchmail, refer to Section 11.3 *Mail Transport Agents*.

11.2.2. Mail Delivery Agent

A *Mail Delivery Agent* (*MDA*) is invoked by the MTA to file incoming email in the proper user's mailbox. In many cases, the MDA is actually a *Local Delivery Agent* (*LDA*), such as `mail` or Procmail.

Any program that actually handles a message for delivery to the point where it can be read by an email client application can be considered an MDA. For this reason, some MTAs (such as Sendmail and Postfix) can fill the role of an MDA when they append new email messages to a local user's mail spool file. In general, MDAs do not transport messages between systems nor do they provide a user interface; MDAs distribute and sort messages on the local machine for an email client application to access.

11.2.3. Mail User Agent

A *Mail User Agent* (*MUA*) is synonymous with an email client application. An MUA is a program that, at the very least, allows a user to read and compose email messages. Many MUAs are capable of retrieving messages via the POP or IMAP protocols, setting up mailboxes to store messages, and sending outbound messages to an MTA.

MUAs may be graphical, such as **Mozilla Mail**, or have a very simple, text-based interface, such as `mutt` or `pine`.

11.3. Mail Transport Agents

Red Hat Linux includes two primary MTAs, Sendmail and Postfix. Sendmail is configured as the default MTA, although it is easy to switch the default MTA to Postfix.

Tip

For information about how to switch the default MTA from Sendmail to Postfix, see the chapter called *Mail Transport Agent (MTA) Configuration* in the *Red Hat Linux Customization Guide*.

Red Hat Linux also includes a special purpose MTA called Fetchmail, which is used to deliver email from a remote MTA to a local MTA.

This section details Sendmail and Fetchmail.

11.3.1. Sendmail

Sendmail's core purpose, like other MTAs, is to safely transfer email among hosts, usually using the SMTP protocol. However, Sendmail is highly configurable, allowing control over almost every aspect of how email is handled, including the protocol used. Many system administrators elect to use Sendmail as their MTA due to its power and scalability.

11.3.1.1. Purpose and Limitations

It is important to be aware of what Sendmail is and what it can do as opposed to what it is not. In these days of monolithic applications that fulfill multiple roles, Sendmail may seem like the only application needed to run an email server within an organization. Technically, this is true, as Sendmail can spool mail to each users' directory and deliver outbound mail for users. However, 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, 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 have been dedicated to helping explain everything that can be done and how to fix things that go wrong. Refer to the Section 11.6 *Additional Resources* for a list of Sendmail resources.

This section reviews the files installed with Sendmail by default and reviews basic configuration changes, including how to stop unwanted email (spam) and how to extend Sendmail with the *Lightweight Directory Access Protocol (LDAP)*.

11.3.1.2. The Default Sendmail Installation

The Sendmail executable is `/usr/sbin/sendmail`.

Sendmail's lengthy and detailed configuration file is `/etc/mail/sendmail.cf`. Avoid editing the `sendmail.cf` file directly. Instead, to make configuration changes to Sendmail, edit the

`/etc/mail/sendmail.mc` file, back up the original `/etc/mail/sendmail.cf`, and then use the included `m4` macro processor to create a new `/etc/mail/sendmail.cf`. More information on configuring Sendmail can be found in Section 11.3.1.3 *Common Sendmail Configuration Changes*.

Various Sendmail configuration files are installed in the `/etc/mail/` directory including:

- `access` — Specifies which systems can use Sendmail for outbound email.
- `domaintable` — Specifies domain name mapping.
- `local-host-names` — Specifies aliases for the host.
- `mailertable` — Specifies instructions that override routing for particular domains.
- `virtusertable` — Specifies 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 made to these configurations in their database files, run the command:

```
makemap hash /etc/mail/<name> < /etc/mail/<name>
```

Where <*name*> is replaced with the name of the configuration file to convert.

For example, to have all emails addressed to the `example.com` domain delivered to <`bob@other-example.com`>, add the following line to the `virtusertable` file:

```
@example.com     bob@other-example.com
```

To finalize the change, the `virtusertable.db` file must be updated using the following command as root:

```
makemap hash /etc/mail/virtusertable < /etc/mail/virtusertable
```

This creates a new `virtusertable.db` file containing the new configuration.

11.3.1.3. Common Sendmail Configuration Changes

When altering the Sendmail configuration file, it is best generate an entirely new `/etc/mail/sendmail.cf` file instead of editing an existing one.

Caution

Before changing the `sendmail.cf` file, it is a good idea to backup a working version of the file.

To add the desired functionality to Sendmail, edit the `/etc/mail/sendmail.mc` file. When finished, use the `m4` macro processor to generate a new `sendmail.cf` by executing the `m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf` command. After creating a new `/etc/mail/sendmail.cf` file, restart Sendmail to make it take effect. The easiest way to do this is to type the `/sbin/service sendmail restart` command as root.

By default, the `m4` macro processor is installed with Sendmail but is part of the `m4` package.

Important

The default `sendmail.cf` file does not allow Sendmail to accept network connections from any host other than the local computer. To configure Sendmail as a server for other clients, edit `/etc/mail/sendmail.mc` and change `DAEMON_OPTIONS` to also listen on network devices or comment out this option all together. Then regenerate `/etc/mail/sendmail.cf` by running:

```
m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
```

This configuration should work for most SMTP-only sites. It will *not* work for UUCP (UNIX to UNIX Copy) sites; you must generate a new `sendmail.cf` if using UUCP mail transfers.

Consult the `/usr/share/sendmail-cf/README` file before editing any files in the directories under the `/usr/share/sendmail-cf` directory, as they can affect the future configuration of `/etc/mail/sendmail.cf` files.

11.3.1.4. Masquerading

One common Sendmail configuration is to have a single machine act as a mail gateway for all machines on the network. For instance, a company may want to have a machine called `mail.bigcorp.com` that handles all of their email and assigns a consistent return address to all out going mail.

In this situation, the Sendmail server must masquerade the machine names on the company network so that their return address is user@bigcorp.com instead of user@devel.bigcorp.com.

To do this, add the following lines to `/etc/mail/sendmail.mc`:

```
FEATURE(always_add_domain)dnl
FEATURE(`masquerade_entire_domain')
FEATURE(`masquerade_envelope')
FEATURE(`allmasquerade')
MASQUERADE_AS(`bigcorp.com.')
MASQUERADE_DOMAIN(`bigcorp.com.')
MASQUERADE_AS(bigcorp.com)
```

After generating a new `sendmail.cf` using `m4`, this configuration will make all mail from inside the network appear as if it were sent from `bigcorp.com`.

11.3.1.5. Stopping Spam

Email spam can be defined as unnecessary and unwanted email received by a user who never requested the communication. It is a disruptive, costly, and widespread abuse of Internet communication standards.

Sendmail makes it relatively easy to block new spamming techniques being employed to send junk email. It even blocks many of the more usual spamming methods by default.

For example, forwarding of SMTP messages, also called relaying, has been disabled by default since Sendmail version 8.9. Before this change occurred, Sendmail would direct the mail host (`x.org`) to accept messages from one party (`y.com`) and send them to a different party (`z.net`). Now, however, Sendmail must be configured to permit any domain to relay mail through the server. To configure relay domains, edit the `/etc/mail/relay-domains` file and restart Sendmail.

However, many times users are bombarded with spam from other servers throughout the Internet. In these instances, Sendmail's access control features available through the `/etc/mail/access` file can be used to prevent connections from unwanted hosts. The following example illustrates how this file can be used to both block and specifically allow access to the Sendmail server:

```
badspammer.com       ERROR:550 "Go away and do not spam us anymore"
tux.badspammer.com   OK
10.0                 RELAY
```

This example states that any email sent from `badspammer.com` is blocked with a 550 RFC-821 compliant error code, with a message sent back to the spammer. 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 the mail server.

Because `/etc/mail/access.db` is a database, use `makemap` to activate any changes. Do this using the following command as root:

```
makemap hash /etc/mail/access < /etc/mail/access
```

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 information and examples.

Since Sendmail calls the Procmail MDA when delivering mail, it is also possible to use a spam filtering program, such as SpamAssassin to identify and file spam for users. Refer to Section 11.4.2.6 *Spam Filters* for more about using SpamAssassin.

11.3.1.6. 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, an LDAP server can be used to look up a particular email address from a common corporate directory by the 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, LDAP abstracts the mail routing level from Sendmail and its separate configuration files to a powerful LDAP cluster that can be leveraged by many different applications.

The current version of Sendmail contains support for LDAP. To extend the Sendmail server using LDAP, first get an LDAP server, such as **OpenLDAP**, running and properly configured. Then edit the `/etc/mail/sendmail.mc` to include the following:

```
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 the implementation of LDAP, especially when configuring 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 the `/etc/mail/sendmail.cf` file by running `m4` and restarting Sendmail. See Section 11.3.1.3 *Common Sendmail Configuration Changes* for instructions on doing this.

For more information on LDAP, see Chapter 13 *Lightweight Directory Access Protocol (LDAP)*.

11.3.2. Fetchmail

Fetchmail is an MTA which retrieves email from remote servers and delivers it to the local MTA. 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 connects and quickly downloads all of the email messages to the mail spool file using any number of protocols, including the POP3 and IMAP. It can even forward email messages to an SMTP server, if necessary.

Fetchmail is configured for each user through the use of a `.fetchmailrc` file in the user's home directory.

Using preferences in the `.fetchmailrc` file, Fetchmail checks for email on a remote server and pulls it off. It then attempts 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 used to filter the email and place it in a mailbox so that it can be read by an MUA.

11.3.2.1. Fetchmail Configuration Options

Although it is possible to pass all necessary options on the command line to check for email on a remote server when executing Fetchmail, using a `.fetchmailrc` file is much easier. All of the configuration options go in the `.fetchmailrc` file and it is possible to 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 for a specific email server, such as the port to check or number of seconds to wait before timing out. These options affect every user option used 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 appear 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 checking that email server. Like server options, multiple user options may be specified for use with a particular server as well as 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 this server's hostname is specified when Fetchmail is invoked. The `skip` option sets up test configurations in `.fetchmailrc` and only checks that server when specifically desired, without affecting any currently working configurations.

A sample `.fetchmailrc` file looks like this:

```
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, 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 user 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

> Users are not required to place their password in the `.fetchmailrc` file. Omitting the `with password '<password>'` section causes Fetchmail to ask for a password when it is launched.

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 option in detail, but the most common ones are listed here.

11.3.2.2. Global Options

Each global option should be placed on a single line after a `set` action.

- `daemon` *<seconds>* — Specifies daemon-mode, where Fetchmail stays in the background and polls for mail at the interval specified.
- `postmaster` — Specifies a local user to send mail to in case of delivery problems.
- `syslog` — Specifies the log file for errors and status messages. By default, this is `/var/log/maillog`.

11.3.2.3. Server Options

Server options must be placed 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, `password` authentication is used, but some protocols support other types of authentication, including `kerberos_v5`, `kerberos_v4`, and `ssh`. If the `any` authentication type is used, Fetchmail first tries methods that do not require a password, then methods that mask the password, and finally attempts to send the password unencrypted to authenticate to the server.
- `interval` *<number>* — Polls the specified server every *<number>* of times that it checks for email on all configured servers. This option is generally used for email servers where the user rarely receives messages.
- `port` *<port-number>* — Overrides the default port number for a specified protocol.
- `proto` *<protocol>* — Specifies a specific protocol, such as `pop3` or `imap`, to check for messages on this server.
- `timeout` *<seconds>* — Specifies the interval of server inactivity after which Fetchmail gives up on a connection attempt. If this value is not set, a default of `300` seconds is assumed.

11.3.2.4. 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 defined options must 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 retrieved before stopping.
- `flush` — Tells Fetchmail to delete all previously viewed messages in the queue before retrieving new messages.
- `limit` <*max-number-bytes*> — Specifies that only messages below a particular size may be retrieved. This option is useful with slow network links, when a large message takes too long to download.
- `password` '<*password*>' — Specifies the password to be used for this user.
- `preconnect` "<*command*>" — Executes the specified command before retrieving messages for this user.
- `postconnect` "<*command*>" — Executes the specified command after retrieving messages for this user.
- `ssl` — Activates SSL encryption.
- `user` "<*username*>" — Sets the username used by Fetchmail to retrieve messages. *This option should be listed before any other user options.*

11.3.2.5. Fetchmail Command Options

Most 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 used whenever Fetchmail is run.

However, there may be times when it is desirable to run the `fetchmail` command with other options for a particular purpose. It is possible to issue command options to temporarily override a `.fetchmailrc` setting that is causing an error, as any options specified at the command line override configuration file options.

11.3.2.6. Informational or Debugging Options

Certain options used after the `fetchmail` command can supply 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` — Displays detailed version information, lists its global options, and shows 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.

11.3.2.7. 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 only downloads 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 to leave them on the remote email server.
- `--quit` — Quits the Fetchmail daemon process.

More commands and `.fetchmailrc` options can be found in the `fetchmail` man page.

11.4. Mail Delivery Agents

Red Hat Linux includes two primary MDAs, Procmail and `mail`. Both of the applications are considered Local Delivery Agents and both move email from the MTA's spool file into the user's mailbox. However, Procmail provides a robust filtering system.

This section details only Procmail. For information on the `mail` command, consult its man page.

Procmail delivers and filters email as it is placed in the mail spool file of the localhost. It is powerful, gentle on system resources, and widely used. Procmail can play a critical role in delivering email to be read by email client applications.

Procmail can be invoked in several different ways. Whenever an MTA places an email into the mail spool file, Procmail is launched. Procmail then filters and files the email so the MUA can find it, and quits. Alternatively, the MUA can be configured to execute Procmail any time a message is received so that messages are moved into their correct mailboxes. By default, the presence of a `.procmailrc` file in the user's home directory will invoke Procmail whenever an MTA receives a new message.

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 `/etc/procmailrc` 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, construct a `.procmailrc` file with particular environment variables and rules for certain types of messages.

In most configurations, the decision as to whether Procmail starts and attempts to filter the email is based on the existence of a user's `.procmailrc` file. To disable Procmail, but save the work on the `.procmailrc` file, move it to a similar file's name using the `mv ~/.procmailrc ~/.procmailrcSAVE` command. When ready to test Procmail again, change the name of the file back to `.procmailrc`. Procmail will begin working again immediately.

11.4.1. 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 at the beginning of `.procmailrc` in the following format:

```
<env-variable>="<value>"
```

In this example, *<env-variable>* is the name of the variable and *<value>* defines the variable.

Many environment variables are not used by most Procmail users, and many of the more important environment variables are already defined by a default value. Most of the time, the following variables will be used:

- `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 breaks up the 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, lines in a user's `.procmailrc` file may look like this:

  ```
  MAILDIR=$HOME/Msgs
  INCLUDERC=$MAILDIR/lists.rc
  INCLUDERC=$MAILDIR/spam.rc
  ```

  If the user would like to turn off Procmail filtering of their email lists but leave spam control in place, they could comment out the first `INCLUDERC` line with a hash mark 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 assumes that the lockfile is old and can be deleted. The default is 1024 seconds.
- `LOGFILE` — The location and file to contain any Procmail information 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 more information. This option is useful for debugging.

Other important environmental variables are pulled from the shell, such as `LOGNAME`, which is the login name; `HOME`, which is the location of the home directory; and `SHELL`, which is the default shell.

A comprehensive explanation of all environments variables, as well as their default values, is available in the `procmailrc` man page.

11.4.2. 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 happening.

A thorough explanation of regular expressions is beyond the scope of this section. 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 depend upon an understanding of Procmail recipe structure. Introductory information specific to basic regular expression rules can be found in the `grep` man page.

A Procmail recipe takes the following form:

```
: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 is created for this message. If a lockfile is to be created, 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 matches 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.

11.4.2.1. Delivering vs. Non-Delivering Recipes

The action used if the recipe matches a particular message determines whether the 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 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.

11.4.2.2. 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 is only used if the 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` — Parses the body of the message and looks for matching conditions.
- `b` — Uses the body in any resulting action, such as writing the message to a file or forwarding it. This is the default behavior.
- `c` — Generates 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` — While similar to the `A` flag, 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 if this recipe is checked only when 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 reports whether or not it was successful before considering the message filtered.

  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.

11.4.2.3. 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. Specify a local lockfile by placing a colon (`:`) after any flags on a recipe's first line. This creates a local lockfile based on the destination file name plus whatever has been set in the `LOCKEXT` global environment variable.

Alternatively, specify the name of the local lockfile to be used with this recipe after the colon.

11.4.2.4. 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:

- `!` — In the condition line, this character 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 specified number of bytes.

The following characters are used to perform special actions:

- `!` — In the action line, this character tells Procmail to forward the message to the specified email addresses
- `$` — Refers to a variable set earlier in the `rc` file. This is often used to set a common mailbox that is 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, Procmail assumes that the action line is specifying a mailbox where the message should be written.

11.4.2.5. Recipe Examples

Procmail is an extremely flexible program, which matches 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.

A basic recipe may not even contain conditions, as is illustrated in the following example:

```
: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 file name and the `LOCKEXT` to name it. No condition is specified, so every message matches this recipe and, therefore, is 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 messages in this file.

This basic recipe could be placed 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 this example.

```
: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 that are matched to `/dev/null`, are permanently deleted. If the recipe conditions inadvertently catch unintended messages, they disappear without a trace, making it hard to troubleshoot the rule.

> A better solution is to point the recipe's action to a special mailbox, which can be checked from time to time in order to look for *false positives*, or messages that inadvertently matched the conditions. Once satisfied that no messages are accidentally being matched, 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 to prevent manual sorting. The following recipe grabs email sent from a particular mailing list and puts it in the correct folder.

```
: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 the 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.

Consult the many Procmail online resources available from Section 11.6 *Additional Resources* to see more detailed and powerful recipes.

11.4.2.6. Spam Filters

Because it is called by Sendmail, Postfix, and Fetchmail upon receiving new emails, Procmail can be used as a powerful tool for combating spam.

This is particularly true when Procmail is used in conjunction with SpamAssassin. When used together, these two applications can quickly identify spam emails, and sort or destroy them.

SpamAssassin uses header analysis, text analysis, blacklists, and a spam-tracking database to quickly and accurately identify and tag spam.

The easiest way for a local user to use SpamAssassin is to place the following line near the top of the `~/.procmailrc` file:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-default.rc
```

The `/etc/mail/spamassassin/spamassassin-default.rc` contains a simple Procmail rule that activates SpamAssassin for all incoming email. If an email is determined to be spam, it is tagged in the header as such and the title is prepended with the following pattern:

```
*****SPAM*****
```

The message body of the email is also prepended with a running tally of what elements caused it to be diagnosed as spam.

To file email tagged as spam, a rule similar to the following can be used:

```
:0 Hw
* ^X-Spam-Status: Yes
spam
```

This rule files all email tagged in the header as spam into a mailbox called `spam`.

Since SpamAssassin is a Perl script, it may be necessary on busy servers to use the binary SpamAssassin daemon (`spamd`) and client application (`spamc`). Configuring SpamAssassin this way, however, requires root access to the host.

To start the `spamd` daemon, type the following command as root:

```
/sbin/service spamassassin start
```

In order for the SpamAssassin daemon to start when the system is booted, use an initscript utility, such as the **Services Configuration Tool** (`redhat-config-services`), to turn on the `spamassassin` service. Refer to Section 1.4.2 *Runlevel Utilities* for more information about initscript utilities.

To configure Procmail to use the SpamAssassin client application instead of the Perl script, place the following line near the top of the `~/.procmailrc` file or for a system-wide configuration, place it in `/etc/procmailrc`:

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

11.5. Mail User Agents

There are scores of mail programs available under Red Hat Linux. There are full-featured, graphical email client programs, such as **Mozilla Mail** or **Ximian Evolution**, as well as text-based email programs such as **mutt** and **pine**.

For instructions on using these applications, refer to the chapter titled *Email Applications* in the *Red Hat Linux Getting Started Guide*.

The remainder of this section focuses on securing communication between the client and server.

11.5.1. Securing Communication

Popular MUAs included with Red Hat Linux, such as **Mozilla Mail**, `mutt`, and `pine`, offer SSL-encrypted email sessions.

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 by users on the network. Additionally, since the standard POP and IMAP protocols pass authentication information unencrypted, it is possible for an attacker to gain access to user accounts by collecting usernames and passwords as they are passed over the network.

11.5.1.1. Secure Email Clients

Most Linux MUAs designed to check email on remote servers support SSL encryption. In order to use SSL when retrieving email, it must be enabled on both the email client and server.

SSL is easy to enable on the client-side, often done with the click of a button in the MUA's configuration window or via an option in the MUA's configuration file. Secure IMAP and POP have known port numbers (993 and 995, respectively) that the MUA will use to authenticate and download messages.

11.5.1.2. Securing Email Client Communications

Offering SSL encryption to IMAP and POP users on the email server is a simple matter.

First, create an SSL certificate. This can be done two ways: by applying to a *Certificate Authority* (*CA*) for an SSL certificate or by creating a self-signed certificate.

Caution

> Self-signed certificates should be used for testing purposes only. Any server used in a production environment should use an SSL certificate granted by a CA.

To create a self-signed SSL certificate for IMAP, change to the /usr/share/ssl/certs/ directory and type the following command as root:

```
make imapd.pem
```

Answer all of the questions to complete the process.

To create a self-signed SSL certificate for POP, change to the /usr/share/ssl/certs/ directory, and type the following command as root:

```
make ipop3d.pem
```

Again, answer all of the questions to complete the process.

Once finished, use the /sbin/service command to start the appropriate daemon (imaps or pop3s). Then, set the imaps or the pop3s service to start at the proper runlevels using an initscript utility, such as **Services Configuration Tool** (redhat-config-services). Refer to Section 1.4.2 *Runlevel Utilities* for more information about initscript utilities.

Alternatively, the stunnel command can be used as an SSL encryption wrapper around the standard, non-secure daemons, imapd or pop3d.

The stunnel program uses external OpenSSL libraries included with Red Hat Linux to provide strong cryptography and protect the connections. It is best to apply to a *Certificate Authority* (*CA*) for an SSL certificate, but it is also possible to create a self-signed certificate.

To create a self-signed SSL certificate, change to the /usr/share/ssl/certs/ directory, and type the following command:

```
make stunnel.pem
```

Again, answer all of the questions to complete the process.

Once the certificate is generated, it is possible to use the stunnel command to start the imapd mail daemon using the following command:

```
/usr/sbin/stunnel -d 993 -l /usr/sbin/imapd imapd
```

Once this command is issued, it is possible to open an IMAP email client and connect to the email server using SSL encryption.

To start the pop3d using the stunnel command, type the following command:

```
/usr/sbin/stunnel -d 993 -l /usr/sbin/pop3d pop3d
```

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.

11.6. Additional Resources

The following is a list of additional documentation about email applications.

11.6.1. Installed Documentation

- Information on configuring 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 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.
- /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, the following 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.
  - /usr/share/doc/spamassassin-<*version-number*>/ — This directory contains a large amount of information pertaining to SpamAssassin. Replace <*version-number*> with the version number of the `spamassassin` package.

11.6.2. 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 the 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 a 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, offers troubleshooting tips, details about file locking, and the use of wildcard characters.
- http://www.uwasa.fi/~ts/info/proctips.html — Contains dozens of tips that make using Procmail much easier. Includes instructions on how to test `.procmailrc` files and use Procmail scoring to decide if a particular action should be taken.
- http://www.spamassassin.org/ — The official site of the SpamAssassin project.

11.6.3. 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 using 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.

Chapter 12. Berkeley Internet Name Domain (BIND)

On most modern networks, including the Internet, users locate other computers by name. This frees users from the daunting task of remembering the numerical network address of network resources. The most effective way to configure a network to allow such name-based connections is to set up a *Domain Name Service* (*DNS*) or *nameserver*, which resolves hostnames on the network to numerical addresses and vice versa.

This chapter reviews the nameserver included in Red Hat Linux, *Berkeley Internet Name Domain* (*BIND*) DNS server, with an emphasis on the structure of its configuration files and how it may be administered both locally or remotely.

For instructions on configuring BIND using the graphical **Bind Configuration Tool** (`redhat-config-bind`), refer to the chapter called *BIND Configuration* in the *Red Hat Linux Customization Guide*.

Warning

> If you use the **Bind Configuration Tool**, do not manually edit any BIND configuration files as all changes will be overwritten the next time you use the **Bind Configuration Tool**.

12.1. Introduction to DNS

When hosts on a network connect to one another via a hostname, also called a *fully qualified domain name (FQDN)*, DNS is used to associate the names of machines to the IP address for the host.

Use of DNS and FQDNs also has advantages for system administrators, allowing the flexibility to change the IP address for a host without effecting name-based queries to the machine. Conversely, administrators can shuffle which machines handle a name-based query.

DNS is normally implemented using centralized servers that are authoritative for some domains and refer to other DNS servers for other domains.

When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the FQDN based on its resolver library, which may contain authoritative information about the host requested or cached data from an earlier query. If the nameserver does not already have the answer in its resolver library, it queries other nameservers, called *root nameservers*, to determine which nameservers are authoritative for the FQDN in question. Then, with that information, it queries the authoritative nameservers to determine the IP address of the requested host. If performing a reverse lookup, the same procedure is used, except the query is made with an unknown IP address rather than a name.

12.1.1. Nameserver Zones

On the Internet, the FQDN of a host can be broken down into different sections. These sections are organized into a hierarchy much like a tree, with a main trunk, primary branches, secondary branches, and so forth. Consider the following FQDN:

```
bob.sales.example.com
```

When looking at how a FQDN is resolved to find the IP address that relates to a particular system, read the name from right to left, with each level of the hierarchy divided by periods (`.`). In this example, `com` defines the *top level domain* for this FQDN. The name `example` is a sub-domain under `com`, while `sales` is a sub-domain under `example`. The name furthest to the left, `bob`, identifies a specific machine.

Except for the hostname, each section is a called a *zone*, which defines a specific *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 how the namespace is organized.

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 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 how the nameserver is configured.

12.1.2. Nameserver Types

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* — Answers queries from other nameservers concerning namespaces for which it is considered an authority. However, slave nameservers get their namespace information from master nameservers.
- *caching-only* — Offers name to IP resolution services but is not authoritative for any zones. Answers for all resolutions are cached in memory for a fixed period of time, which is specified by the retrieved zone record.
- *forwarding* — Forwards requests to a specific list of nameservers for name resolution. If none of the specified nameservers can perform the resolution, 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 resolutions for others.

12.1.3. BIND as a Nameserver

BIND name performs name resolution services through the `/usr/sbin/named` daemon. BIND also includes an administration utility called `/usr/sbin/rndc`. More information about `rndc` can be found in Section 12.4 *Using `rndc`*.

BIND stores its configuration files in the following two places:

- `/etc/named.conf` — The configuration file for the `named` daemon.
- `/var/named/` directory — The `named` working directory which stores zone, statistic, and cache files.

The next few sections review the BIND configuration files in more detail.

12.2. `/etc/named.conf`

The `named.conf` file is a collection of statements using nested options surrounded by opening and closing ellipse characters, `{ }`. Administrators must be careful when editing `named.conf` to avoid syntactical errors as many seemingly minor errors will prevent the `named` service from starting.

Warning

> Do *not* 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.

A typical `named.conf` file is organized similar to the following example:

```
<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>;
};
```

12.2.1. Common Statement Types

The following types of statements are commonly used in `/etc/named.conf`:

12.2.1.1. `acl` Statement

The `acl` statement (or access control statement) defines groups of hosts which can then be permitted or denied access to the nameserver.

An `acl` statement takes the following form:

```
acl <acl-name> {
    <match-element>;
    [<match-element>; ...]
};
```

In this statement, replace *<acl-name>* with the name of the access control list and replace *<match-element>* with a semi-colon separated list of IP addresses. Most of the time, an individual IP address or IP network notation (such as `10.0.1.0/24`) is used to identify the IP addresses within the `acl` statement.

The following access control lists are already defined as keywords to simplify configuration:

- `any` — Matches every IP address.
- `localhost` — Matches any IP address in use by the local system.

- `localnets` — Matches any IP address on any network to which the local system is connected.
- `none` — Matches no IP addresses.

When used in conjunction with other statements (such as the `options` statement), `acl` statements can be very useful in preventing the misuse of a BIND nameserver.

The following example defines two access control lists and uses an `options` statement to define how they are treated by the nameserver:

```
  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 example contains two access control lists, `black-hats` and `red-hats`. Hosts in the `black-hats` list are denied access to the nameserver, while hosts in the `red-hats` list are given normal access.

12.2.1.2. `include` Statement

The `include` statement allows files to be included in a `named.conf`. This way sensitive configuration data (such as `keys`) can be placed in a separate file with restrictive permissions.

An `include` statement takes the following form:

```
include  "<file-name>"
```

In this statement, *<file-name>* is replaced with an absolute path to a file.

12.2.1.3. `options` Statement

The `options` statement defines global server configuration options and sets defaults for other statements. It can be used to specify the location of the `named` working directory, the types of queries allowed, and much more.

The `options` statement takes the following form:

```
options {
        <option>;
 [<option>; ...]
};
```

In this statement, the *<option>* directives are replaced with a valid option.

The following are commonly used options:

- `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`, this option applies to recursive queries. By default, all hosts are allowed to perform recursive queries on the nameserver.
- `blackhole` — Specifies which hosts are not allowed to query the server.
- `directory` — Changes the `named` working directory to something other than the default value, `/var/named/`.
- `forward` — Controls forwarding behavior of a `forwarders` directive.

  The following options are accepted:

  - `first` — Specifies that the namservers specified in the `forwarders` directive be queried before `named` attempts to resolve the name itself.
  - `only` — Specifies that `named` not attempt name resolution itself in the event queries to namservers specified in the `forwarders` directive fail.

- `forwarders` — Specifies a list of valid IP addresses for nameservers where requests should be forwarded for resolution.
- `listen-on` — Specifies the network interface on which `named` listens for queries. By default, all interfaces are used.

  In this way, if the DNS server is also a gateway, BIND can be configured to only answer queries that originate from one of the networks.

  A `listen-on` directive might look like this:

  ```
  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. It accepts the following options:

  - `yes` — Notifies slave servers.
  - `no` — Does not notify slave servers.
  - `explicit` — Only notifies slave servers specified in an `also-notify` list within a zone statement.

- `pid-file` — Specifies the location of the process ID file created by `named`.
- `statistics-file` — Specifies an alternate location for statistics files. By default, `named` statistics are saved to the `/var/named/named.stats` file.

Dozens of other options are also available, many of which rely upon one another to work properly. See the *BIND 9 Administrator Reference Manual* in Section 12.7.1 *Installed Documentation* and the man page for `bind.conf` for more details.

12.2.1.4. zone Statement

A zone statement defines the characteristics of a zone such as the location of its configuration file and zone-specific options. This statement can be used to override the global options statements.

A zone statement takes the following form:

```
zone <zone-name> <zone-class> {
     <zone-options>;
     [<zone-options>; ...]
};
```

In this statement, <*zone-name*> is the name of the zone, <*zone-class*> is the optional class of the zone, and <*zone-options*> is a list of options characterizing the zone.

The <*zone-name*> attribute for the zone statement is particularly important, as it is the default value assigned for the $ORIGIN directive used within the corresponding zone file located in the /var/named/ directory. The named daemon appends the name of the zone to any non-fully qualified domain name listed in the zone file.

For example, if a zone statement defines the namespace for example.com, use example.com as the <*zone-name*> so it is placed at the end of hostnames within the example.com zone file.

For more information about zone files, see Section 12.3 *Zone Files*.

The most common zone statement options include the following:

- 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.

  Be careful when allowing hosts to update information about their zone. Do not enable this option unless the host specified is completely trusted. In general, it better to have an administrator manually update the records for a zone and reload the named service.
- file — Specifies the name of the file in the named working directory that contains the zone's configuration data.
- masters — The masters option lists the IP addresses from which to request authoritative zone information. Used only if the zone is defined as type slave.
- notify — Controls whether named notifies the slave servers when a zone is updated. It accepts the following options:
  - yes — Notifies slave servers.
  - no — Does not notify slave servers.
  - explicit — Only notifies slave servers specified in an also-notify list within a zone statement.
- type — Defines the type of zone.

  Below is a list of valid options:
  - forward — Forwards all requests for information about this zone to other nameservers.
  - hint — A special type of zone used to point to the root nameservers which resolve queries when a zone is not otherwise known. No configuration beyond the default is necessary with a hint zone.

- `master` — Designates the nameserver as authoritative for this zone. A zone should be set as the `master` if the zone's configuration files reside on the system.
- `slave` — Designates the nameserver as a slave server for this zone. Also specifies the IP address of the master nameserver for the zone.

- `zone-statistics` — Configures `named` to keep statistics concerning this zone, writing them to either the default location (`/var/named/named.stats`) or the file listed the `statistics-file` option in the `server` statement. See Section 12.2.2 *Other Statement Types* for more information about the `server` statement.

12.2.1.5. Sample `zone` Statements

Most changes to the `/etc/named.conf` file of a master or slave nameserver involves adding, modifying, or deleting `zone` statements. While these `zone` statements can contain many options, most nameservers require only a small subset to function efficiently. The following `zone` statements are very basic examples illustrating a master-slave nameserver relationship.

The following is an example of a `zone` statement for the primary nameserver hosting `example.com` (`192.168.0.1`):

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-update { none; };
};
```

In the statement, the zone is identified as `example.com`, the type is set to `master`, and the `named` service is instructed to read the `/var/named/example.com.zone` file. It also tells `named` not to allow by any other hosts to update.

A slave server's `zone` statement for `example.com` looks slightly different from the previous example. For a slave server, the type is set to `slave` and in place of the `allow-update` line is a directive telling `named` the IP address of the master server.

A slave server's `zone` statement for example.com may look like this:

```
zone "example.com" {
  type slave;
  file "example.com.zone";
  masters { 192.168.0.1; };
};
```

This `zone` statement configures `named` on the slave server to look for the master server at the `192.168.0.1` IP address for information about the `example.com` zone. The information the slave server receives from the master server is saved to the `/var/named/example.com.zone` file.

12.2.2. Other Statement Types

The following is a list of lesser used statement types available within `named.conf`

- `controls` — Configures various security requirements necessary to use the `rndc` command to administer the `named` service.

  See Section 12.4.1 *Configuring `/etc/named.conf`* to see how the `controls` statement should look, including various options that may only be used with 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 Section 12.4.2 *Configuring `/etc/rndc.conf`* for instruction on how to write 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 to 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`. This occurs because 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* in Section 12.7.1 *Installed Documentation*.

- `server` — Defines particular options that affect how `named` should act toward remote nameservers, especially regarding notifications and zone transfers.

  The `transfer-format` option controls whether one resource record is sent 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 secure DNS (DNSSEC). See Section 12.5.3 *Security* for more information concerning BIND security.

- `view "<view-name>"` — Creates special views depending upon the host contacting to 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 can only 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 `options` 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 12.5.2 *Multiple Views* for more information about the `view` statement.

12.2.3. Comment Tags

The following is a list of valid comment tags used within `named.conf`:

- `//` — When placed at the beginning of a line, that line is ignored by `named`.
- `#` — When placed at the beginning of a line, that line is ignored by `named`.
- `/*` and `*/` — When text is enclose in these tags, the block of text is ignored by `named`.

12.3. Zone Files

Zone files contain information about a particular namespace and are stored in the `named` working directory, `/var/named/` by default. 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 perform tasks or apply special settings to the zone. *Resource records* define the parameters of the zone and assign identities to individual hosts. Directives are optional, but resource records are required to provide nameservice to a zone.

All directives and resource records should go on their own individual lines.

Comments can be placed after semicolon characters (`;`) in zone files.

12.3.1. Zone File Directives

Directives begin with the dollar sign character (`$`) followed by the name of the directive. They usually appear at the top of the zone file.

The following are commonly used directives:

- `$INCLUDE` — Configures `named` to include another zone file in this zone file at the place where the directive appears. This allows additional zone settings to be stored apart from the main zone file.
- `$ORIGIN` — Appends the domain name to unqualified records, such as those with the hostname and nothing more.

  For example, a zone file may contains the following line:

  ```
  $ORIGIN example.com
  ```

  Any names used in resource records that do not end in a trailing period (`.`) will have `example.com` appended to them.

  Note

  The use of the `$ORIGIN` directive is unnecessary if the zone is specified in `/etc/named.conf` because the zone name is used as the value for the `$ORIGIN` directive by default.

- `$TTL` — Sets the default *Time to Live (TTL)* value for the zone. This is the length of time, in seconds, a zone resource record is valid. Each resource record can contain its own TTL value, which overrides this directive.

  Increasing this value allows remote nameservers to cache the zone information for a longer period of time, reducing the number of queries for the zone and lengthening the amount of time required to proliferate resource record changes.

12.3.2. Zone File Resource Records

The primary component of a zone file is its resource records.

There are many types of zone file resource records. The following are used most frequently:

- `A` — Address record, which specifies an IP address to assign to a name, as in this example:

  ```
  <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 is the target for all non-FQDN requests.

Consider the following `A` record examples for the `example.com` zone file:

```
                IN      A       10.0.1.3
server1         IN      A       10.0.1.5
```

Requests for `example.com` are pointed to 10.0.1.3, while requests for `server1.example.com` are pointed to 10.0.1.5.

- `CNAME` — Canonical name record, maps one name to another. This type of record is also known as an alias record.

  The next example tells `named` that any requests sent to the <*alias-name*> will point to the host, <*real-name*>. `CNAME` records are most commonly used to point to services that use a common naming scheme, such as `www` for Web servers.

  ```
  <alias-name>     IN     CNAME       <real-name>
  ```

  In the following example, an `A` record binds a hostname to an IP address, while a `CNAME` record points the commonly used `www` hostname to it.

  ```
  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.

  ```
          IN      MX      <preference-value>   <email-server-name>
  ```

  In this example, the <*preference-value*> allows numerical ranking of the email servers for a namespace, giving preference to some email systems over others. The `MX` resource record with the lowest <*preference-value*> is preferred over the others. However, multiple email servers can possess the same value to distribute email traffic evenly among them.

  The <*email-server-name*> may be a hostname or FQDN.

  ```
          IN      MX      10      mail.example.com.
          IN      MX      20      mail2.example.com.
  ```

  In this example, the first `mail.example.com` email server is preferred to the `mail2.example.com` email server when receiving email destined for the `example.com` domain.

- `NS` — NameServer record, which announces the authoritative nameservers for a particular zone.

  This is an example of an `NS` record:

  ```
          IN      NS      <nameserver-name>
  ```

  The <*nameserver-name*> should be a FQDN.

  Next, two nameservers are listed as authoritative for the domain. It is not important whether these nameservers are slaves or if one is a master; they are both still considered authoritative.

  ```
          IN      NS      dns1.example.com.
          IN      NS      dns2.example.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 Section 12.3.4 *Reverse Name Resolution Zone Files* for more examples of `PTR` records in use.

- `SOA` — Start Of Authority record, proclaims important authoritative information about a namespace to the nameserver.

  Located after the directives, an `SOA` resource record is the first resource record in a zone file.

  The following example shows the basic structure of an `SOA` record:

  ```
  @       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 (in seconds).

With BIND, all times refer to seconds. However, it is possible to use abbreviations when specifying units of time other than seconds, such as minutes (`M`), hours (`H`), days (`D`), and weeks (`W`). The table in Table 12-1 shows an amount of time in seconds and the equivalent time in another format.

Seconds	Other Time Units
60	1M
1800	30M
3600	1H
10800	3H
21600	6H
43200	12H
86400	1D
259200	3D
604800	1W
31536000	365D

Table 12-1. Seconds compared to other time units

The following example illustrates the form an `SOA` resource record might take when it is configured with real values.

```
@      IN     SOA    dns1.example.com.     hostmaster.example.com. (
                     2001062501 ; serial
                     21600      ; refresh after 6 hours
                     3600       ; retry after 1 hour
                     604800     ; expire after 1 week
                     86400 )    ; minimum TTL of 1 day
```

12.3.3. Example Zone File

Seen individually, directives and resource records can be difficult to grasp. However, when placed together in a single file, they become easier to understand.

The following example shows a very basic zone file.

```
$ORIGIN example.com
$TTL 86400
@      IN     SOA    dns1.example.com.     hostmaster.example.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.example.com.
       IN     NS     dns2.example.com.

       IN     MX     10     mail.example.com.
       IN     MX     20     mail2.example.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 as `dns1.example.com` and `dns2.example.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 period (`.`), the `$ORIGIN` domain is placed after them, expanding them to `server1.example.com` and `server2.example.com`. Through the related `A` resource records, their IP addresses can be determined.

FTP and Web services, available at the standard `ftp.example.com` and `www.example.com` names, are pointed at the appropriate servers using `CNAME` records.

12.3.4. 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 fully qualified domain name.

A `PTR` record looks similar to this:

```
<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 the follow example, IP addresses `10.0.1.20` through `10.0.1.25` are pointed to corresponding FQDNs.

```
$ORIGIN 1.0.10.in-addr.arpa
$TTL 86400
@      IN     SOA    dns1.example.com.     hostmaster.example.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.example.com.
        IN     NS     dns2.example.com.

20      IN     PTR    alice.example.com.
21      IN     PTR    betty.example.com.
22      IN     PTR    charlie.example.com.
23      IN     PTR    doug.example.com.
24      IN     PTR    ernest.example.com.
25      IN     PTR    fanny.example.com.
```

This zone file would be called into service with a `zone` statement in the `named.conf` file which looks similar to the following:

```
zone "1.0.10.in-addr.arpa" IN {
  type master;
  file "example.com.rr.zone";
  allow-update { none; };
};
```

There is very little difference between this example and a standard `zone` statement, except for the zone name. Note that a reverse name resolution zone requires the first three blocks of the IP address reversed followed by `.in-addr.arpa`. This allows the single block of IP numbers used in the reverse name resolution zone file to be associated with the zone.

12.4. Using **rndc**

BIND includes a utility called `rndc` which allows command line administration of the `named` daemon from the localhost or from a remote host.

In order to prevent unauthorized access to the `named` daemon, BIND uses a shared secret key method is used to grant privileges to hosts. This means an identical key must be present in both `/etc/named.conf` and the `rndc` configuration file, `/etc/rndc.conf`

12.4.1. Configuring **/etc/named.conf**

In order for `rndc` to connect to a `named` service, there must a `controls` statement in the BIND server's `/etc/named.conf` file.

The `controls` statement below shown in the following example allows `rndc` to connect from the localhost.

```
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. The next example illustrates a sample `key` statement.

```
key "<key-name>" {
  algorithm hmac-md5;
  secret "<key-value>";
};
```

In this case, the <*key-value*> is a HMAC-MD5 key. Use the following command to generate HMAC-MD5 keys:

```
dnssec-keygen -a hmac-md5 -b <bit-length> -n HOST <key-file-name>
```

A key with at least a 256-bit length is a good idea. The actual key that should be placed in the <*key-value*> area can be found in the <*key-file-name*>.

Caution

Because `/etc/named.conf` is world-readable, it is a good idea to place the `key` statement in a separate file readable only by root and then use an `include` statement to reference it, as in the following example:

```
include "/etc/rndc.key";
```

12.4.2. Configuring `/etc/rndc.conf`

The `key` is the most important 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 match the keys specified in the target server's `/etc/named.conf`, add the following lines to `/etc/rndc.conf`.

```
options {
  default-server  localhost;
  default-key     "<key-name>";
};
```

This command sets a global default key. However the `rndc` command can also use different keys for different servers, as in the following example:

```
server localhost {
  key  "<key-name>";
};
```

Caution

Make sure that only the root user can read or write to the `/etc/rndc.conf` file.

12.4.3. Command Line Options

An `rndc` command takes the following form:

```
rndc <options> <command> <command-options>
```

When executing `rndc` on a properly configured localhost, the following commands are available:

- `halt` — Stops the `named` service immediately.
- `querylog` — Logs all queries made to this nameserver.
- `refresh` — Refreshes the nameserver's database.
- `reload` — Reloads the zone files but keeps all other previously cached responses. This command also allows changes to zone files without losing all stored name resolutions.

  If changes only affected a specific zone, reload only one zone by adding the name of the zone after the `reload` command.
- `stats` — Dumps the current `named` statistics to the `/var/named/named.stats` file.
- `stop` — Stops the server gracefully, saving any dynamic update and *Incremental Zone Transfers* (*IXFR*) data before exiting.

Occasionally, it may be necessary 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 port number to use for the `rndc` connection other than port 953, the default.
- `-s` <*server*> — Tells `rndc` to send the command to a server other than the `default-server` specified in its configuration file.
- `-y` <*key-name*> — Specifies a key other than the `default-key` option in the `/etc/rndc.conf` file.

Additional information about these options can be found in the `rndc` man page.

12.5. Advanced Features of BIND

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 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.

All of the features mentioned here are discussed in greater detail in the *BIND 9 Administrator Reference Manual*. See Section 12.7.1 *Installed Documentation* for more information.

12.5.1. 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 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 when using *dynamic updating* to make changes to master zone records. If manually editing zone files to make changes, AXFR is used. More information on dynamic updating is available in the *BIND 9 Administrator Reference Manual*. See Section 12.7.1 *Installed Documentation* for more information.

12.5.2. Multiple Views

Through the use of the `view` statement in `named.conf`, BIND can present different information depending on who is making the request.

This is primarily used to deny sensitive DNS entries from clients outside of the local network, while allowing queries from clients inside the local network.

The `view` statement uses the `match-clients` option to match IP addresses or entire networks and give them special options and zone data.

12.5.3. 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*, this feature allows a transfer from master to slave is authorized only after verifying that a shared secret key exists on the master and slave servers.

  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 supports *TKEY*, which is another shared secret key method of authorizing zone transfers.

12.5.4. IP version 6

BIND version 9 can provide nameservice in IP version 6 (IPv6) environments through the use of `A6` zone records.

If the network environment includes both IPv4 and IPv6 hosts, use the `lwresd` lightweight resolver daemon on all network clients. This daemon is a very efficient, caching-only nameserver which understands the new `A6` and `DNAME` records used under IPv6. See the `lwresd` man page for more information.

12.6. Common Mistakes to Avoid

It is very common for beginners to make mistakes when editing BIND configuration files. Be sure to avoid the following issues:

- *Take care to increment the serial number when editing a zone file.*

  If the serial number is not incremented, the master nameserver may have the correct, new information, but the slave nameservers will never be notified of the change or attempt to refresh their data of that zone.

- *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 periods (`.`) in zone files after all FQDNs and omit them on hostnames.*

  A period at the end of a domain name denotes a fully qualified domain name. If the period is omitted, then `named` appends the name of the zone or the `$ORIGIN` value to complete it.

- *If a firewall is blocking connections from the `named` program to other nameservers, edit its configuration file.*

  By default, BIND version 9 uses random ports above 1024 to query other nameservers. Some firewalls, however, expect all nameservers to communicate using only port 53. Force `named` to use port 53 by adding the following line to the `options` statement of `/etc/named.conf`:

  ```
  query-source address * port 53;
  ```

12.7. Additional Resources

The following sources of information will provide additional resources regarding BIND.

12.7.1. 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 reviews 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 in this directory.

- `man named` — Explores assorted arguments that can be used to control the BIND nameserver daemon.

- `man named.conf` — A comprehensive list of options available within the `named` configuration file.
- `man rndc` — Explains the different options available when using the `rndc` command to control a BIND nameserver.
- `man rndc.conf` — A comprehensive list of options available within the `rndc` configuration file.

12.7.2. Useful Websites

- http://www.isc.org/products/BIND — The home page of the BIND project containing information about current releases as well as 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.

12.7.3. 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.

Chapter 13. Lightweight Directory Access Protocol (LDAP)

Lightweight Directory Access Protocol (*LDAP*) is a set of open protocols used to access centrally stored information over a network. It is based on the *X.500* standard for directory sharing, but is less complex and resource intensive. For this reason, LDAP is sometimes referred to as "*X.500 Lite*."

Like X.500, LDAP organizes information in a hierarchal manner using directories. These directories can store a variety of information and can even be used in a manner similar to Network Information Service (NIS), enabling anyone to access their account from any machine on the LDAP enabled network.

In many cases, however, LDAP is used simply as a virtual phone directory, allowing users to easily access contact information for other users. But LDAP is more flexible than a traditional phone directory, because it is capable of referring a querent to other LDAP servers throughout the world, providing an ad-hoc global repository of information. Currently, however, LDAP is more commonly used within individual organizations, like universities, government departments, and private companies.

LDAP is a client-server system. The server can use a variety of databases to store a directory, each optimized for quick and copious read operations. When an LDAP client application connects to an LDAP server, it can either query a directory or attempt to modify it. In the event of a query, the server either answers the query or, if it can not answer locally, it can refer the querent to an LDAP server which does have the answer. If the client application is attempting to modify information an LDAP directory, the server verifies that the user has permission to make the change and then adds or updates the information.

This chapter will refer to the configuration and use of OpenLDAP 2.0, an open source implementation of the LDAPv2 and LDAPv3 protocols.

13.1. Why Use LDAP?

The main benefit of using LDAP is that information for an entire organization can be consolidated into a central repository. For example, rather than managing user lists for each group within an organization, you can use LDAP as a central directory accessible from anywhere on the network. And because LDAP supports Secure Sockets Layer (SSL) and Transport Layer Security (TLS), sensitive data can be protected from prying eyes.

LDAP also supports a number of back-end databases in which to store directories. This allows administrators the flexibility to deploy the database best suited for the type of information the server is to disseminate. Because LDAP also has a well-defined client Application Programming Interface (API), the number of LDAP-enabled applications are numerous and increasing in quantity and quality.

On the negative side, LDAP can be difficult to configure.

13.1.1. OpenLDAP 2.0 Feature Enhancements

OpenLDAP 2.0 includes a number of important features.

- *LDAPv3 Support* — OpenLDAP 2.0 supports Simple Authentication and Security Layer (SASL), Transport Layer Security (TLS), and Secure Sockets Layer (SSL), among other improvements. Many of the changes in the protocol since LDAPv2 are designed to make LDAP more secure.
- *IPv6 Support* — OpenLDAP supports the next generation Internet Protocol version 6.
- *LDAP Over IPC* — OpenLDAP can communicate within a system using interprocess communication (IPC). This enhances security by obviating the need to communicate over a network.

- *Updated C API* — Improves the way programmers can connect to and use LDAP directory servers.
- *LDIFv1 Support* — Provides 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.

13.2. LDAP Terminology

Any discussion of LDAP requires a basic understanding of a set of LDAP-specific terms:

- *entry* — An entry is a single unit within an LDAP directory. Each entry is identified by its unique *Distinguished Name (DN)*.
- *attributes* — Attributes are pieces of information directly associated with an entry. For example, an organization could be represented as an LDAP entry. Attributes associated with the organization might be its fax number, its address, and so on. People can also be represented as entries in the LDAP directory. Common attributes for people include the person's telephone number and email address.

  Some attributes are required, while other attributes are optional. An *objectclass* definition sets which attributes are required and which are not for each entry. Objectclass definitions are found in various schema files, located in the `/etc/openldap/schema/` directory. For more information on LDAP schema, see Section 13.5 *The `/etc/openldap/schema/` Directory*.
- *LDIF* — The *LDAP Data Interchange Format* (LDIF) is an ASCII text representation of LDAP entries. Files used for importing data to LDAP servers must be in LDIF format. An LDIF entry looks similar to the following example:

```
[<id>]
dn: <distinguished name>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
```

  Each entry can contain as many <*attrtype*>: <*attrvalue*> pairs as needed. A blank line indicates the end of an entry.

  Caution

  All <*attrtype*> and <*attrvalue*> pairs *must* be defined in a corresponding schema file to use this information.

  Any value enclosed within a "<" and a ">" is a variable and can be set whenever a new LDAP entry is created. This rule does not apply, however, to <*id*>. The <*id*> is a number determined by the application you use to edit the entry.

  Note

  You should never need to edit an LDIF entry by hand. Instead use an LDAP client application, such as the ones listed in Section 13.3 *OpenLDAP Daemons and Utilities*.

13.3. OpenLDAP Daemons and Utilities

The suite of OpenLDAP libraries and tools is spread out over the following packages:

- `openldap` — Contains the libraries necessary to run the OpenLDAP server and client applications.
- `openldap-clients` — Contains command line tools for viewing and modifying directories on an LDAP server.
- `openldap-servers` — Contains the servers and other utilities necessary to configure and run an LDAP server.

There are two servers contained in the `openldap-servers` package: the *Standalone LDAP Daemon* (`/usr/sbin/slapd`) and the *Standalone LDAP Update Replication Daemon* (`/usr/sbin/slurpd`).

The `slapd` daemon is the standalone LDAP server while the `slurpd` daemon is used to synchronize changes from one LDAP server to other LDAP servers on the network. The `slurpd` daemon is only used when dealing with multiple LDAP servers.

To perform administrative tasks, the `openldap-servers` package installs the following utilities into the `/usr/sbin/` directory:

- `slapadd` — Adds entries from an LDIF file to an LDAP directory. For example, the command `/usr/sbin/slapadd -l ldif-input` will read in the LDIF file, *`ldif-input`*, containing the new entries.
- `slapcat` — Pulls entries out of an LDAP directory in the default format — Berkeley DB — and saves them in an LDIF file. For example, the command `/usr/sbin/slapcat -l ldif-output` will output an LDIF file called *`ldif-output`* containing the entries from the LDAP directory.
- `slapindex` — Re-indexes the `slapd` directory based on the current content.
- `slappasswd` — Generates an encrypted user password value for use with `ldapmodify` or the `rootpw` value in the `slapd` configuration file, `/etc/openldap/slapd.conf`. Execute the `/usr/sbin/slappasswd` command to create the password.

Warning

Be sure to stop `slapd` by issuing `/usr/sbin/service slapd stop` before using `slapadd`, `slapcat` or `slapindex`. Otherwise, the integrity of the LDAP directory is at risk.

For more information about how to use these utilities, see their respective man pages.

The `openldap-clients` package installs tools into `/usr/bin/` which are used to add, modify, and delete entries in an LDAP directory. These tools include the following:

- `ldapmodify` — Modifies entries in an LDAP directory, accepting input via a file or standard input.
- `ldapadd` — Adds entries to your directory by 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 by accepting input via user input at the terminal or via a file.

With the exception of `ldapsearch`, each of these utilities is more easily used by referencing a file containing the changes to be made rather than typing a command for each entry you wish to change in an LDAP directory. The format of such a file is outlined in each application's man page.

13.3.1. NSS, PAM, and LDAP

In addition to the OpenLDAP packages, Red Hat Linux includes a package called `nss_ldap` which enhances LDAP's ability to integrate into both Linux and other UNIX environments.

The `nss_ldap` package provides the following modules:

- `/lib/libnss_ldap-`<*glibc-version*>`.so`
- `/lib/security/pam_ldap.so`

The `libnss_ldap-`<*glibc-version*>`.so` module allows applications to look up users, groups, hosts, and other information using an LDAP directory via glibc's *Nameservice Switch* (NSS) interface. NSS allows applications to authenticate using LDAP in conjunction with the *Network Information Service* (NIS) name service and flat authentication files.

The `pam_ldap` module allows PAM-aware applications to authenticate users using information stored in an LDAP directory. PAM-aware applications include console login, POP and IMAP mail servers, and Samba. By deploying an LDAP server on your network, all of these applications can authenticate using the same user ID and password combination, greatly simplifying administration.

13.3.2. PHP4, the Apache HTTP Server, and LDAP

Red Hat Linux includes a package containing an LDAP module for the PHP server-side scripting language.

The `php-ldap` package adds LDAP support to the PHP4 HTML-embedded scripting language via the `/usr/lib/php4/ldap.so` module. This module allows PHP4 scripts to access information stored in an LDAP directory.

Important

> Red Hat Linux no longer ships with the `auth_ldap` package. This package provided LDAP support for versions 1.3 and earlier of the Apache HTTP Server. See the Apache Software Foundation website at http://www.apache.org/ for details on the status of this module.

13.3.3. LDAP Client Applications

There are graphical LDAP clients available which support creating and modifying directories, but they do not ship with Red Hat Linux. One such application is **LDAP Browser/Editor** — A Java-based tool available online at http://www.iit.edu/~gawojar/ldap.

Most other LDAP clients access directories as read-only, using them to reference, but not alter, organization-wide information. Some examples of such applications are Mozilla-based Web browsers, Sendmail, **Balsa**, **Pine**, **Evolution**, and **Gnome Meeting**.

13.4. OpenLDAP Configuration Files

OpenLDAP configuration files are installed into the `/etc/openldap/` directory. The following is a brief list highlighting the most important directories and files:

- `/etc/openldap/ldap.conf` — This is the configuration file for all *client* applications which use the OpenLDAP libraries such as `ldapsearch`, `ldapadd`, Sendmail, **Pine**, **Balsa**, **Evolution**, and **Gnome Meeting**.
- `/etc/openldap/slapd.conf` — This is the configuration file for the `slapd` daemon. See Section 13.6.1 *Editing `/etc/openldap/slapd.conf`* for more information about this file.
- `/etc/openldap/schema/` directory — This subdirectory contains the schema used by the `slapd` daemon. See Section 13.5 *The `/etc/openldap/schema/` Directory* for more information about this directory.

Note

If the `nss_ldap` package is installed, it will create a file named `/etc/ldap.conf`. This file is used by the PAM and NSS modules supplied by the `nss_ldap` package. See Section 13.7 *Configuring Your System to Authenticate Using OpenLDAP* for more information about this configuration file.

13.5. The `/etc/openldap/schema/` Directory

The `/etc/openldap/schema/` directory holds 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, define new attribute types and object classes within the `local.schema` file. Many organizations use existing attribute types from the schema files installed by default and add new object classes to the `local.schema` file.

Extending schema 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.

13.6. OpenLDAP Setup Overview

This section provides a quick overview for installing and configuring an OpenLDAP directory. For more details, refer to the following URLs:

- http://www.openldap.org/doc/admin/quickstart.html — The *Quick-Start Guide* on the OpenLDAP website.
- http://www.redhat.com/mirrors/LDP/HOWTO/LDAP-HOWTO.html — The *LDAP Linux HOWTO* from the Linux Documentation Project, mirrored on Red Hat's website.

The basic steps for creating an LDAP server are as follows:

1. Install the `openldap`, `openldap-servers`, and `openldap-clients` RPMs.
2. Edit the `/etc/openldap/slapd.conf` file to reference your LDAP domain and server. Refer to Section 13.6.1 *Editing `/etc/openldap/slapd.conf`* for more information on how to edit this file.
3. Start `slapd` with the command:
   `/sbin/service/ldap start`

   After you have configured LDAP correctly, you can use `chkconfig`, `ntsysv`, or the **Services Configuration Tool** to configure LDAP to start at boot time. For more information about configuring services, refer to the chapter titled *Controlling Access to Services* in the *Red Hat Linux Customization Guide*.
4. Add entries to your LDAP directory with `ldapadd`.
5. Use `ldapsearch` to see if `slapd` is accessing the information correctly.
6. At this point, your LDAP directory should be functioning properly and you can configure any LDAP-enabled applications to use the LDAP directory.

13.6.1. Editing `/etc/openldap/slapd.conf`

In order to use the `slapd` LDAP server, you will need to modify its configuration file, `/etc/openldap/slapd.conf`. You must edit this file to specify the correct domain and server.

The `suffix` line names the domain for which the LDAP server will provide information and should be changed from:

```
suffix          "dc=your-domain,dc=com"
```

so that it reflects a fully qualified domain name. For example:

```
suffix          "dc=example,dc=com"
```

The `rootdn` entry is the *Distinguished Name* (*DN*) for a user who is unrestricted by access controls 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. In the configuration file, change the `rootdn` line from its default value to something like the example below:

```
rootdn          "cn=root,dc=example,dc=com"
```

If you intend to populate the LDAP directory over the network, change the `rootpw` line — replacing the default value with an encrypted password string. To create an encrypted password string, type the following command:

```
slappasswd
```

You will be prompted to type and then re-type a password, then the program prints the resulting encrypted password to the terminal.

Next, copy the newly created encrypted password into the `/etc/openldap/slapd.conf` on one of the `rootpw` lines and remove the hash mark (`#`).

When finished, the line should look similar to the following example:

```
rootpw {SSHA}vv2y+i6V6esazrIv70xSSnNAJE18bb2u
```

Warning

LDAP passwords, including the `rootpw` directive specified in `/etc/openldap/slapd.conf`, are sent over the network in *unencrypted*, unless you enable TLS encryption.

To enable TLS encryption review the comments in `/etc/openldap/slapd.conf` and see the man page for `slapd.conf`.

For added security, the `rootpw` directive should be commented out after populating the LDAP directory by preceding it with a hash mark (`#`).

When using the `/usr/sbin/slapadd` command line tool locally to populate the LDAP directory, use of the `rootpw` directive is not necessary.

Important

You must be the root user to use `/usr/sbin/slapadd`. However, the directory server runs as the `ldap` user. Therefore the directory server will not be able to modify any files created by `slapadd`. To correct this issue, after you have finished using `slapadd`, type the following command:

```
chown -R ldap /var/lib/ldap
```

13.7. Configuring Your System to Authenticate Using OpenLDAP

This section provides a brief overview of how to configure a Red Hat Linux system to authenticate using OpenLDAP. Unless you are an OpenLDAP expert, you will probably need more documentation than is provided here. Please refer to the references provided in Section 13.9 *Additional Resources* for more information.

Install the Necessary LDAP Package

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-servers` package.

The `openldap`, `openldap-clients`, and `nss_ldap` packages need to be installed on all LDAP client machines.

Edit the Configuration Files

- On the server, edit the `/etc/openldap/slapd.conf` file on the LDAP server to make sure it matches the specifics of your organization. Please refer to Section 13.6.1 *Editing /etc/openldap/slapd.conf* for instructions on editing `slapd.conf`.
- On the client machines, both `/etc/ldap.conf` and `/etc/openldap/ldap.conf` need to contain the proper server and search base information for your organization.

  The simplest way to do this is to run the **Authentication Configuration Tool** (`authconfig-gtk`) and select **Enable LDAP Support** under the **User Information** tab.

  You can also edit these files by hand.
- On the client machines, the `/etc/nsswitch.conf` must be edited to use LDAP.

  The simplest way to do this is to run the **Authentication Configuration Tool** (`authconfig-gtk`) and select **Enable LDAP Support** under the **User Information** tab.

  If editing `/etc/nsswitch.conf` by hand, add `ldap` to the appropriate lines.

  For example:
  ```
  passwd: files ldap
  shadow: files ldap
  group: files ldap
  ```

13.7.1. PAM and LDAP

To have standard PAM-enabled applications use LDAP for authentication, run the **Authentication Configuration Tool** (`authconfig-gtk`) and select **Enable LDAP Support** under the the **Authentication** tab. For more on configuring PAM consult, Chapter 14 *Pluggable Authentication Modules (PAM)* and the PAM man pages.

13.7.2. Migrating Old Authentication Information to LDAP Format

The `/usr/share/openldap/migration/` directory contains a set of shell and Perl scripts for migrating authentication information into LDAP format.

First, modify the `migrate_common.ph` file so that it reflects your domain. The default DNS domain should be changed from its default value to something like:

```
$DEFAULT_MAIL_DOMAIN = "your_company";
```

The default base should also be changed, to something like:

```
$DEFAULT_BASE = "dc=your_company,dc=com";
```

The job of migrating a user database into a format that is LDAP readable falls to a group of migration scripts installed in the same directory. Using Table 13-1, decide which script to run in order to migrate your user database.

Existing name service	Is LDAP running?	Script to Use
`/etc` flat files	yes	`migrate_all_online.sh`
`/etc` flat files	no	`migrate_all_offline.sh`

Existing name service	Is LDAP running?	Script to Use
NetInfo	yes	migrate_all_netinfo_online.sh
NetInfo	no	migrate_all_netinfo_offline.sh
NIS (YP)	yes	migrate_all_nis_online.sh
NIS (YP)	no	migrate_all_nis_offline.sh

Table 13-1. LDAP Migration Scripts

Run the appropriate script based on your existing name service.

Note

You must have Perl installed on your system to use some of these scripts.

The README and the migration-tools.txt files in the /usr/share/openldap/migration/ directory provide more details on how to migrate the information.

13.8. Upgrading to OpenLDAP Version 2.0

In OpenLDAP Version 2.0, the on-disk storage format used by the slapd LDAP server has changed. If you are upgrading LDAP from Red Hat Linux 7.0 or earlier, you will need to extract the existing LDAP directories to an LDIF file using the following command:

```
ldbmcat -n > <ldif_file>
```

In the above command, change <*ldif_file*> to the name of the output file. Next type the following command to import this file into OpenLDAP 2.0:

```
slapadd -l <ldif_file>
```

Important

You must be the root user to use /usr/sbin/slapadd. However, the directory server runs as the ldap user. Therefore the directory server will not be able to modify any files created by slapadd. To correct this issue, after you have finished using slapadd, type the following command:

```
chown -R ldap /var/lib/ldap
```

13.9. Additional Resources

More information concerning LDAP is available. Please review these sources, especially the OpenLDAP website and the LDAP HOWTO, before configuring LDAP on your system.

13.9.1. Installed Documentation

- LDAP man pages — The `ldap` man page provides a good introduction to LDAP. Man pages also exist for the various LDAP daemons and utilities.
- `/usr/share/docs/openldap-`<*versionnumber*> — Contains a general `README` document and miscellaneous information.

13.9.2. Useful Websites

- http://www.openldap.org/ — Home of the OpenLDAP Project. This website contains a wealth of information about configuring OpenLDAP.
- http://www.redhat.com/mirrors/LDP/HOWTO/LDAP-HOWTO.html — An older, but still relevant LDAP HOWTO.
- http://www.padl.com/ — Developers of `nss_ldap` and `pam_ldap`, among other useful LDAP tools.
- 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.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 a directory tree and customizing directory structures.

13.9.3. Related Books

- *Implementing LDAP* by Mark Wilcox; Wrox Press, Inc.
- *Understanding and Deploying LDAP Directory Services* by Tim Howes et al.; Macmillan Technical Publishing

III. Security Reference

Using secure protocols is a critical part of maintaining system integrity. This part describes critical tools used for the purpose of user authentication, network access control, secure network communication, and intrusion detection. For more information about securing a Red Hat Linux system, refer to the *Red Hat Linux Security Guide*.

Table of Contents

Chapter 14. Pluggable Authentication Modules (PAM)

Programs which allow users access to a system verify the user's identity through a process called *authentication*. Historically, each such program had its own way of performing the task of authentication. Under Red Hat Linux, many such programs are configured to use a centralized authentication process called *Pluggable Authentication Modules* (*PAM*).

PAM uses a pluggable, modular architecture, which affords the system administrator a great deal of flexibility in setting authentication policies for the system.

In most situations, the default PAM configuration file for a PAM-aware application is sufficient. However, sometimes it may become necessary to edit a PAM configuration file. Because misconfiguration of PAM can compromise system security, it is important to understand the structure of these files before making any modifications (refer to Section 14.3 *PAM Configuration File Format* for more information).

14.1. Advantages of PAM

PAM offers the following advantages:

- Provides a common authentication scheme that can be used with a wide variety of applications.
- Allows a large amount of flexibility and control over authentication for both the system administrator and application developer.
- Allows application developers to develop programs without creating their own authentication scheme.

14.2. PAM Configuration Files

The directory `/etc/pam.d/` contains the PAM configuration files for each PAM-aware applications. In earlier versions of PAM, the file `/etc/pam.conf` was used, but this file is now deprecated and `pam.conf` is only read if the `/etc/pam.d/` directory does not exist.

14.2.1. PAM Service Files

Each PAM-aware application or *service* has a file within the `/etc/pam.d/` directory. Each of these files is named after the service for which it controls access.

It is up to the PAM-aware program to define its service name and install its PAM configuration file in the `/etc/pam.d/` directory. For example, the `login` program defines its service name as `/etc/pam.d/login`.

14.3. PAM Configuration File Format

Each PAM configuration file contains a group of directives formatted as follows:

```
<module interface>  <control flag>   <module path>   <module arguments>
```

Each of these elements are explained in the following sections.

14.3.1. Module Interface

There are four types of PAM module interfaces which correlate to different aspects of the authorization process:

- `auth` — These modules authenticate the user by, for example, asking for and checking a password. Modules with this interface can also set credentials, such as group membership or Kerberos tickets.
- `account` — These modules verify that access is allowed. For example, it can check if a user account is expired or is allowed to log in at a particular time of day.
- `password` — These modules set and verify passwords.
- `session` — These modules configure and manage user sessions. Modules with this interface can also perform additional tasks that are needed to allow access, like mounting a user's home directory and making the user's mailbox available.

Note

An individual module can provide any or all module interfaces. For instance, `pam_unix.so` provides all four interfaces.

In a PAM configuration file, the module interface is the first field defined. For example a typical line in a configuration may look like this:

```
auth      required  /lib/security/pam_unix.so
```

This instructs PAM to use the `pam_unix.so` module's `auth` interface.

14.3.1.1. Stacking Modules

Module interface directives can be *stacked*, or placed upon one another, so that multiple modules are used together for a one purpose. For this reason, the order in which the modules are listed is very important to the authentication process.

Stacking makes it very easy for an administrator to require specific conditions to exist before allowing the user to authenticate. For example, `rlogin` normally uses five stacked `auth` modules, as 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
```

Before someone is allowed to use `rlogin`, PAM verifies that the `/etc/nologin` file does not exist, that they are not trying to log in remotely as a root user over an unencrypted network connection, and that any environmental variables can be loaded. Then, if a successful `rhosts` authentication is performed, the connection is allowed. If the `rhosts` authentication fails, then a standard password authentication is performed.

14.3.2. Control Flag

All PAM modules generate a success or failure result when called. Control flags tell PAM what do with the result. Since modules can be stacked in a particular order, control flags decide how important the success or failure of a particular module is to the overall goal of authenticating the user to the service.

There are four predefined control flags:

- `required` — The module result must be successful for authentication to continue. If a `required` module result fails, the user is not notified until results on all modules referencing that interface are completed.
- `requisite` — The module result must be successful for authentication to continue. However, if a `requisite` module result fails, the user is notified immediately with a message reflecting the first failed `required` *or* `requisite` module.
- `sufficient` — The module result is ignored if it fails. But, if a `sufficient` flagged module result is successful *and* no `required` flagged modules above it have failed, then no other results are required and the user is authenticated to the service.
- `optional` — The module result is ignored if it fails. If the module result is successful, it does not play a role in the overall success or failure for the module interface. A module flagged as `optional` becomes necessary for successful authentication when there are no other modules referencing that interface. In this case, an `optional` module determines the overall PAM authentication for that interface.

Important

> The order in which `required` modules are called is not critical. The `sufficient` and `requisite` control flags cause order to become important.

A newer control flag syntax which allows for more precise control is now available for PAM. Please see the PAM docs located in the `/usr/share/doc/pam-`<*version-number*>`/` directory for information on this new syntax (where <*version-number*> is the version number for PAM).

14.3.3. Module Path

Module paths tell PAM where to find the pluggable module to be used with the module interface 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, then the module indicated is assumed to be in the `/lib/security/` directory, the default location for PAM modules.

14.3.4. Module Arguments

PAM uses arguments to pass information to a pluggable module during authentication for some modules.

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 embedded in many applications. The module takes a `db` argument so that Berkeley DB knows which database to use for the requested service.

A typical `pam_userdb.so` line within a PAM configuration file looks like this:

```
auth      required  /lib/security/pam_userdb.so db=<path-to-file>
```

In the previous example, replace <*path-to-file*> with the full path to the Berkeley DB database file.

Invalid arguments are ignored and do not otherwise affect the success or failure of the PAM module. However, most modules will report an error to the `/var/log/messages` file.

14.4. Sample PAM Configuration Files

Below is a sample PAM application configuration file:

```
#%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 retry=3
password   required   /lib/security/pam_unix.so shadow nullok use_authtok
session    required   /lib/security/pam_unix.so
```

The first line is a comment as denoted by the hash mark (`#`) at the beginning of the line.

Lines two through four stack three modules for login authentication.

```
auth       required   /lib/security/pam_securetty.so
```

This module makes sure that *if* the user is trying to log in as root, the tty on which the user is logging in is listed in the `/etc/securetty` file, *if* that file exists.

```
auth       required   /lib/security/pam_unix.so shadow nullok
```

This module prompts the user for a password and then checks the password using the information stored in `/etc/passwd` and, if it exists, `/etc/shadow`. The `pam_unix.so` module automatically detects and uses shadow passwords to authenticate users. Please refer to the Section 6.5 *Shadow Passwords* for more information on shadow passwords.

The argument `nullok` instructs the `pam_unix.so` module to allow a blank password.

```
auth       required   /lib/security/pam_nologin.so
```

This is the final authentication step. It results to see if the file `/etc/nologin` exists. If `nologin` does exist and the user is not root, authentication fails.

Note

In this example, all three `auth` modules are checked, even if the first `auth` module fails. This prevents the user from knowing at what stage their authentication failed. Such knowledge in the hands of an attacker could allow them to more easily deduce how to crack the system.

```
account    required   /lib/security/pam_unix.so
```

This module performs any necessary account verification. For example, if shadow passwords have been enabled, the account component of the `pam_unix.so` module will check to see if the account has expired or if the user has not changed the password within the grace period allowed.

```
password   required   /lib/security/pam_cracklib.so retry=3
```

If a password has expired, the password component of the pam_cracklib.so module prompts for a new password. It then tests the newly created password to see whether it can easily be determined by a dictionary-based password cracking program. If it fails this test the first time, it gives the user two more chances to create a strong password, as specified in the retry=3 argument.

```
password  required  /lib/security/pam_unix.so shadow nullok use_authtok
```

This line specifies that if the program changes the user's password, it should use the password component of the pam_unix.so module to do so. This only happens if the auth portion of the pam_unix.so module has determined that the password needs to be changed.

The argument shadow tells the module to create shadow passwords when updating a user's password.

The argument nullok instructs the module to allow the user to change their password *from* a blank password, otherwise a null password is treated as an account lock.

The final argument on this line, use_authtok, provides a good example of the importance of order when stacking PAM modules. This argument tells the module not to prompt the user for a new password. Instead, it accepts any password that was recorded by a previous password module. In this way all, new passwords must pass the pam_cracklib.so test for secure passwords before being accepted.

```
session required /lib/security/pam_unix.so
```

The final line specifies that the session component of the pam_unix.so module will manage the session. This module logs the username and the service type to /var/log/messages at the the beginning and end of each session. It can be supplemented by stacking it with other session modules for more functionality.

The next sample configuration file illustrates auth module stacking for the rlogin program.

```
#%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 it does, no one can log in except for root.

```
auth      required    /lib/security/pam_securetty.so
```

The pam_securetty.so module prevents the root user from logging in on insecure terminals. This effectively disallows all root rlogin attempts due to the application's limited security safeguards.

Tip

To log in remotely as the root user, use OpenSSH instead. For more information on the SSH protocol, see Chapter 18 *SSH Protocol*.

```
auth      required    /lib/security/pam_env.so
```

This line loads the pam_env.so module, which sets the environmental variables specified in /etc/security/pam_env.conf.

```
auth      sufficient  /lib/security/pam_rhosts_auth.so
```

The pam_rhosts_auth.so module authenticates the user using .rhosts in the user's home directory. If this succeeds, PAM immediately considers the authentication to have succeeded. If pam_rhosts_auth.so fails to authenticate the user, the authentication attempt is ignored.

```
auth      required    /lib/security/pam_stack.so service=system-auth
```

If the pam_rhosts_auth.so module fails to successfully authenticate the user, the pam_stack.so module performs normal password authentication.

The argument service=system-auth indicates that the user must now pass through the PAM configuration for system authentication as found in /etc/pam.d/system-auth.

Tip

To prevent PAM from prompting for a password when the securetty result fails, change the pam_securetty.so module from required to requisite.

14.5. Creating PAM Modules

New PAM modules can be added at any time for PAM-aware applications to use. For example, if a developer invents a one-time-password creation method and writes a PAM module to support it, PAM-aware programs can immediately use the new module and password method without being recompiled or otherwise modified. This allows developers and system administrators to mix-and-match, as well as test, authentication methods for different programs without recompiling them.

Documentation on writing modules is included with the system in the /usr/share/doc/pam-<*version-number*>/ directory (where <*version-number*> is the version number for PAM).

14.6. PAM and Device Ownership

Red Hat Linux allows the first user to log in on the physical console of the machine the ability to manipulate some devices and perform some tasks normally reserved for the root user. This is controlled by a PAM module called pam_console.so.

14.6.1. Device Ownership

When a user logs into a machine under Red Hat Linux, the pam_console.so module is called by login or the graphical login programs, **gdm** and **kdm**. If this user is the first user to log in at the physical console — called the *console user* — the module grants the user ownership of a variety of devices normally owned by root. The console user owns these devices until the last local session for that user ends. Once the user has logged out, ownership of the devices reverts back to the root user.

The devices affected include, but are not limited to, sound cards, diskette drives, and CD-ROM drives.

This allows a local user to manipulate these devices without attaining root, thus simplifying common tasks for the console user.

By modifying the file /etc/security/console.perms, the administrator can edit the list of devices controlled by pam_console.so.

14.6.2. Application Access

The console user is also allowed access to certain programs with a file bearing the command name in the /etc/security/console.apps/ directory.

One notable group of applications the console user has access to are three programs which shut off or reboot the system. These are:

- /sbin/halt
- /sbin/reboot
- /sbin/poweroff

Because these are PAM-aware applications, they call the pam_console.so module as a requirement for use.

For more information, refer to the man pages for pam_console, console.perms, console.apps, and userhelper.

14.7. Additional Resources

The following is a list of resources for using and configuring PAM. In addition to these resources, read the PAM configuration files on the system to better understand how they are structured.

14.7.1. Installed Documentation

- man pam — Good introductory information on PAM, including the structure and purpose of the PAM configuration files.
- /usr/share/doc/pam-<*version-number*> — Contains a *System Administrators' Guide*, a *Module Writers' Manual*, and the *Application Developers' Manual*, as well as a copy of the PAM standard, DCE-RFC 86.0.

14.7.2. 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, a FAQ, and additional PAM documentation.

Chapter 15.

TCP Wrappers and `xinetd`

Controlling access to network services is one of the most important security tasks facing a server administrator. Fortunately, under Red Hat Linux there are a number of tools which do just that. For instance, an `iptables`-based firewall filters out unwelcome network packets within the kernel's network stack. For network services that utilize it, *TCP wrappers* add an additional layer of protection by defining which hosts are allowed or not allowed to connect to "*wrapped*" network services. One such wrapped network service is the `xinetd` *super server*. This service is called a super server because it controls connections to a subset of network services and further refines access control.

Figure 15-1 is a basic illustration of how these tools work together to protect network services.

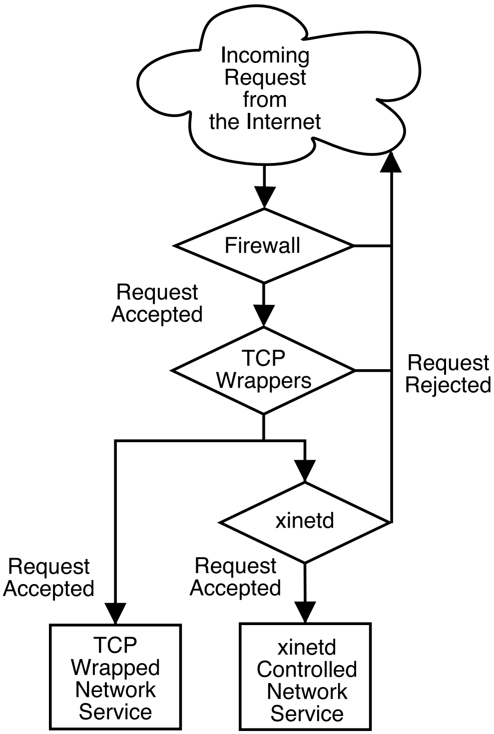

Figure 15-1. Access Control to Network Services

This chapter focuses on the role of TCP wrappers and `xinetd` in controlling access to network services and reviews how these tools can be used to enhance both logging and utilization management. For a discussion of firewalling with `iptables`, see Chapter 16 *`iptables`*.

15.1. TCP Wrappers

The TCP wrappers package (tcp_wrappers) is installed by default under Red Hat Linux and provides host-based access control to network services. The most important component within the package is the /usr/lib/libwrap.a library. In general terms, a TCP wrapped service is one that has been compiled against the libwrap.a library.

When a connection attempt is made to a TCP wrapped service, the service first references the *hosts access* files (/etc/hosts.allow and /etc/hosts.deny) to determine whether or not the client host is allowed to connect. It then uses the syslog daemon (syslogd) to write the name of the requesting host and the requested service to /var/log/secure or /var/log/messages.

If a client host is allowed to connect, TCP wrappers release control of the connection to the requested service and do not interfere further with communication between the client host and the server.

In addition to access control and logging, TCP wrappers can activate commands to interact with the client before denying or releasing control of the connection to the requested network service.

Because TCP wrappers are a valuable addition to any server administrator's arsenal of security tools, most network services within Red Hat Linux are linked against the libwrap.a library. Some such applications include /usr/sbin/sshd, /usr/sbin/sendmail, and /usr/sbin/xinetd.

Note

To determine if a network service binary is linked against libwrap.a, type the following command as the root user:

```
strings -f <binary-name> | grep hosts_access
```

Replacing *<binary-name>* with the name of the network service binary.

15.1.1. Advantages of TCP Wrappers

TCP wrappers provide the following advantages over other network service control techniques:

- *Transparency to both the client host and the wrapped network service.* — Both the connecting client and the wrapped network service are unaware that TCP wrappers are in use. Legitimate users are logged and connected to the requested service while connections from banned clients fail.
- *Centralized management of multiple protocols.* — TCP wrappers operate separately from the network services they protect, allowing many server applications to share a common set of configuration files for simpler management.

15.2. TCP Wrappers Configuration Files

To determine if a client machine is allowed to connect to a service, TCP wrappers reference the following two files, which are commonly referred to as hosts access files:

- /etc/hosts.allow
- /etc/hosts.deny

When a client request is received by a TCP wrapped service, it takes the following basic steps:

1. *The service references* `/etc/hosts.allow`. — The TCP wrapped service sequentially parses the `/etc/hosts.allow` file and applies the first rule specified for that service. If it finds a matching rule, it allows the connection. If not, it moves on to step 2.
2. *The service references* `/etc/hosts.deny`. — The TCP wrapped service sequentially parses the `/etc/hosts.deny` file. If it finds a matching rule is denies the connection. If not, access to the service is granted.

The following are important points to consider when using TCP wrappers to protect network services:

- Because access rules in `hosts.allow` are applied first, they take precedence over rules specified in `hosts.deny`. Therefore, if access to a service is allowed in `hosts.allow`, a rule denying access to that same service in `hosts.deny` is ignored.
- Since the rules in each file are read from the top down and the first matching rule for a given service is the only one applied, the order of the rules is extremely important.
- If no rules for the service are found in either file, or if neither file exists, access to the service is granted.
- TCP wrapped services do not cache the rules from the hosts access files, so any changes to `hosts.allow` or `hosts.deny` take effect immediately without restarting network services.

15.2.1. Formatting Access Rules

The format for both `/etc/hosts.allow` and `/etc/hosts.deny` are identical. Any blank lines or lines that start with a hash mark (`#`) are ignored, and each rule must be on its own line.

Each rule uses the following basic format to control access to network services:

```
<daemon list>: <client list> [: <option>: <option>: ...]
```

- *<daemon list>* — A comma separated list of process names (*not* service names) or the `ALL` *wildcard* (see Section 15.2.1.1 *Wildcards*). The daemon list also accepts *operators* listed in Section 15.2.1.3 *Operators* to allow greater flexibility.
- *<client list>* — A comma separated list of hostnames, host IP addresses, special *patterns* (see Section 15.2.1.2 *Patterns*), or special wildcards (see Section 15.2.1.1 *Wildcards*) which identify the hosts effected by the rule. The client list also accepts operators listed in Section 15.2.1.3 *Operators* to allow greater flexibility.
- *<option>* — An optional action or colon separated list of actions performed when the rule is triggered. Option fields support *expansions* (see Section 15.2.3.4 *Expansions*), launch shell commands, allow or deny access, and alter logging behavior (see Section 15.2.3 *Option Fields*).

The following is a basic sample hosts access rule:

```
vsftpd : .example.com
```

This rule instructs TCP wrappers to watch for connections to the FTP daemon (`vsftpd`) from any host in the `example.com` domain. If this rule appears in `hosts.allow`, the connection will be accepted. If this rule appears in `hosts.deny`, the connection will be rejected.

The next sample hosts access rule is more complex and uses two option fields:

```
sshd : .example.com  \
: spawn /bin/echo `/bin/date` access denied>>/var/log/sshd.log \
: deny
```

Note that in this example that each option field is preceded by the backslash (`\`). Use of the backslash prevents failure of the rule due to length.

Warning

If the last line of a hosts access file is not a newline character (created by pressing the [Enter] key), the last rule in the file will fail and an error will be logged to either `/var/log/messages` or `/var/log/secure`. This is also the case for a rule lines that span multiple lines without using the backslash. The following example illustrates the relevant portion of a log message for a rule failure due to either of these circumstances:

```
warning: /etc/hosts.allow, line 20: missing newline or line too long
```

This sample rule states that if a connection to the SSH daemon (`sshd`) is attempted from a host in the `example.com` domain, execute the `echo` command (which will log the attempt to a special file), and deny the connection. Because the optional `deny` directive is used, this line will deny access even if it appears in the `hosts.allow` file. For a more detailed look at available options, see Section 15.2.3 *Option Fields*.

15.2.1.1. Wildcards

Wildcards allow TCP wrappers to more easily match groups of daemons or hosts. They are used most frequently in the client list field of access rules.

The following wildcards may be used:

- `ALL` — Matches everything. It can be used for both the daemon list and the client list.
- `LOCAL` — Matches any host that does not contain a period (`.`), such as localhost.
- `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 with care as a disruption in name resolution may prevent legitimate users from gaining access to a service.

15.2.1.2. Patterns

Patterns can be used in the client list field of access rules to more precisely specify groups of client hosts.

The following is a list of the most common accepted patterns for a client list entry:

- *Hostname beginning with a period (`.`)* — Placing a period at the beginning of a hostname, matches all hosts sharing the listed components of the name. The following example would apply to any host within the `example.com` domain:

```
ALL : .example.com
```

- *IP address ending with a period (.)* — Placing a period at the end of an IP address matches all hosts sharing the initial numeric groups of an IP address. The following example would apply to any host within the `192.168.x.x` network:
  ```
  ALL : 192.168.
  ```
- *IP address/netmask pair* — Netmask expressions can also be used as a pattern to control access to a particular group of IP addresses. The following example would apply to any host with an address of `192.168.0.0` through `192.168.1.255`:
  ```
  ALL : 192.168.0.0/255.255.254.0
  ```
- *The asterisk (*)* — Asterisks can be used to match entire groups of hostnames or IP addresses, as long as they are not mixed in a client list containing other types of patterns. The following example would apply to any host within the `example.com` domain:
  ```
  ALL : *.example.com
  ```
- *The slash (/)* — If a client list begins with a slash, it is treated as a file name. This is useful if rules specifying large numbers of hosts are necessary. The following example refers TCP wrappers to the `/etc/telnet.hosts` file for all Telnet connections:
  ```
  in.telnetd : /etc/telnet.hosts
  ```

Other, lesser used patterns are also accepted by TCP wrappers. See the hosts access man 5 page for more information.

Warning

Be very careful when creating rules requiring name resolution, such as hostnames and domain names. Attackers can use a variety of tricks to circumvent accurate name resolution. In addition, any disruption in DNS service would prevent even authorized users from using network services.

It is best to use IP addresses whenever possible.

15.2.1.3. Operators

At present, access control rules accept one operator, `EXCEPT`. It can be used in both the daemon list and the client list of a rule.

The `EXCEPT` operator allows specific exceptions to broader matches within the same rule.

In the following example from a `hosts.allow` file, all `example.com` hosts are allowed to connect to all services except `cracker.example.com`:

```
ALL: .example.com EXCEPT cracker.example.com
```

In the another example from a `hosts.allow` file, clients from the 192.168.0.*x* network can use all services except for FTP:

```
ALL EXCEPT vsftpd: 192.168.0.
```

Note

Organizationally, it is often easier to use `EXCEPT` operators sparingly, placing the exceptions to a rule in the other access control file. This allows other administrators to quickly scan the appropriate files to see what hosts should are allowed or denied access to services, without having to sort through the various EXCEPT operators.

15.2.2. Portmap and TCP Wrappers

When creating access control rules for portmap, do not use hostnames as its implementation of TCP wrappers does not support host look ups. For this reason, only use IP addresses or the keyword ALL when specifying hosts is in hosts.allow or hosts.deny.

In addition, changes to portmap access control rules may not take affect immediately.

Widely used services, such as NIS and NFS, depend on portmap to operate, so be aware of these limitations.

15.2.3. Option Fields

In addition to basic rules allowing and denying access, the Red Hat Linux implementation of TCP wrappers supports extensions to the access control language through option fields. By using option fields within hosts access rules, administrators can accomplish a variety of tasks such as altering log behavior, consolidating access control, and launching shell commands.

15.2.3.1. Logging

Option fields let administrators easily change the log facility and priority level for a rule by using the severity directive.

In the following example, connections to the SSH daemon from any host in the example.com domain are logged to the the default authpriv facility (because no facility value is specified) with a priority of emerg:

```
sshd : .example.com : severity emerg
```

It is also possible to specify a facility using the severity option. The following example logs any SSH connection attempts by hosts from the example.com domain to the local0 facility with a priority of alert:

```
sshd : .example.com : severity local0.alert
```

Note

In practice, this example will not work until the syslog daemon (syslogd) is configured to log to the local0 facility. See the syslog.conf man page for information about configuring custom log facilities.

15.2.3.2. Access Control

Option fields also allow administrators to explicitly allow or deny hosts in a single rule by adding the allow or deny directive as the final option.

For instance, the following two rules allow SSH connections from client-1.example.com, but deny connections from client-2.example.com:

```
sshd : client-1.example.com : allow
sshd : client-2.example.com : deny
```

By allowing access control on a per-rule basis, the option field allows administrators to consolidate all access rules into a single file: either hosts.allow or hosts.deny. Some consider this an easier way of organizing access rules.

15.2.3.3. Shell Commands

Option fields allow access rules to launch shell commands through the following two directives:

- spawn — Launches a shell command as a child process. This option directive can perform tasks like using /usr/sbin/safe_finger to get more information about the requesting client or create special log files using the echo command.

  In the following example, clients attempting to access Telnet services from the example.com domain are quietly logged to a special file:

  ```
  in.telnetd : .example.com \
    : spawn /bin/echo `/bin/date` from %h>>/var/log/telnet.log \
    : allow
  ```

- twist — Replaces the requested service with the specified command. This directive is often used to set up traps for intruders (also called "honey pots"). It can also be used to send messages to connecting clients. The twist command must occur at the end of the rule line.

  In the following example, clients attempting to access FTP services from the example.com domain are sent a message via the echo command:

  ```
  vsftpd : .example.com \
  : twist /bin/echo "421 Bad hacker, go away!"
  ```

For more information about shell command options, see the hosts_options man page.

15.2.3.4. Expansions

Expansions, when used in conjunction with the spawn and twist directives provide information about the client, server, and processes involved.

Below is a list of supported expansions:

- %a — The client's IP address.
- %A — The server's IP address.
- %c — Supplies a variety 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.

The following sample rule uses an expansion in conjunction with the spawn command to identify the client host in a customized log file.

It instructs TCP wrappers that if a connection to the SSH daemon (sshd) is attempted from a host in the example.com domain, execute the echo command to log the attempt, including the client hostname (using the %h expansion), to a special file:

```
sshd : .example.com  \
: spawn /bin/echo `/bin/date` access denied to %h>>/var/log/sshd.log \
: deny
```

Similarly, expansions can be used to personalize messages back to the client. In the following example, clients attempting to access FTP services from the example.com domain are informed that they have been banned from the server:

```
vsftpd : .example.com \
: twist /bin/echo "421 %h has been banned from this server!"
```

For a full explanation of available expansions, as well as additional access control options, review see section 5 of the man pages for hosts_access (man 5 hosts_access) and the man page for hosts_options.

For additional resources concerning TCP wrappers, see Section 15.5 *Additional Resources*.

15.3. xinetd

The xinetd daemon is a TCP wrapped *super service* which controls access to a subset of popular network services including FTP, IMAP, and Telnet. It also provides service-specific configuration options for access control, enhanced logging, binding, redirection, and resource utilization control.

When a client host attempts to connect to a network service controlled by xinetd, the super service receives the request and checks for any TCP wrappers access control rules. If access is allowed, xinetd verifies the connection is allowed under its own access rules for that service and that the service is not consuming more than its alloted amount of resources or in breach of any defined rules. It then starts an instance of the requested service and passes control of the connection to it. Once the connection is established, xinetd does not interfere further with communication between the client host and the server.

15.4. xinetd Configuration Files

The configuration files for xinetd are as follows:

- /etc/xinetd.conf — The global xinetd configuration file.
- /etc/xinetd.d/ directory — The directory containing all service-specific files.

15.4.1. The /etc/xinetd.conf File

The /etc/xinetd.conf contains general configuration settings which effect every service under xinetd's control. It is read once when the xinetd service is started, so in order for configuration changes to take effect, the administrator must restart the xinetd service. Below is a sample /etc/xinetd.conf file:

```
defaults
{
        instances               = 60
        log_type                = SYSLOG authpriv
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
}
includedir /etc/xinetd.d
```

These lines control various aspects of xinetd:

- instances — Sets the maximum number of requests xinetd can handle at once.
- log_type — Configures xinetd to use the authpriv log facility, which writes log entries to the /var/log/secure file. Adding a directive such as FILE /var/log/xinetdlog here would create a custom log file called xinetdlog in the /var/log/ directory.
- log_on_success — Configures xinetd 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 — Configures xinetd to log if there is a connection failure or if the connection is not allowed.
- cps — Configures xinetd to allow no more than 25 connections per second to any given service. If this limit is reached, the service is retired for 30 seconds.
- includedir /etc/xinetd.d/ — Includes options declared in the service-specific configuration files located in the /etc/xinetd.d/ directory. Refer to Section 15.4.2 *The /etc/xinetd.d/ Directory* for more information about this directory.

Note

Often, both the log_on_success and log_on_failure settings in /etc/xinetd.conf are further modified in the service-specific log files. For this reason, more information may appear in a given service's log than this file may indicate. See Section 15.4.3.1 *Logging Options* for more about logging options.

15.4.2. The /etc/xinetd.d/ Directory

The files in the /etc/xinetd.d/ directory contains the configuration files for each service managed by xinetd and the names of the files correlate to the service. As with xinetd.conf, this file is read only when the xinetd service is started. In order for any changes to take effect, the administrator must restart the xinetd service.

The format of files in the /etc/xinetd.d/ directory use the same conventions as /etc/xinetd.conf. The primary reason the configuration for each service is stored in separate file is to make customization easier and less likely to effect other services.

To get an idea of how these files are structured, consider the /etc/xinetd.d/telnet file:

```
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = yes
}
```

These lines control various aspects of the `telnet` service:

- `service` — Defines the service name, usually to match a service listed in the `/etc/services` file.
- `flags` — Sets any of a number of attributes for the connection. `REUSE` instructs `xinetd` to reuse the socket for a Telnet connection.
- `socket_type` — Sets the network socket type to `stream`.
- `wait` — Defines whether the service is single-threaded (`yes`) or multi-threaded (`no`).
- `user` — Defines what user ID the process process will run under.
- `server` — Defines the binary executable to be launched.
- `log_on_failure` — Defines logging parameters for `log_on_failure` in addition to those already defined in `xinetd.conf`.
- `disable` — Defines whether or not the service is active.

15.4.3. Altering `xinetd` Configuration Files

There are a large assortment of directives available for `xinetd` protected services. This section highlights some of the more commonly used options.

15.4.3.1. Logging Options

The following logging options are available for both `/etc/xinetd.conf` and the service-specific configuration files in the `/etc/xinetd.d/` directory.

Below is a list of some of the more commonly used logging options:

- `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`).

For a complete list of logging options, consult the `xinetd.conf` man page.

15.4.3.2. Access Control Options

Users of `xinetd` services can choose to use the TCP wrappers hosts access rules, provide access control via the `xinetd` configuration files, or a mixture of both. Information concerning the use of TCP wrappers hosts access control files can be found in Section 15.2 *TCP Wrappers Configuration Files*. This section discusses using `xinetd` to control access to services.

Note

Unlike TCP wrappers, changes to access control only take effect if the `xinetd` administrator restarts the `xinetd` service.

The `xinetd` hosts access control differs 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 its own access control rules.

The following hosts access options are supported by `xinetd`:

- `only_from` — Allows only the specified hosts to use the service.
- `no_access` — Blocks listed hosts from using the service.
- `access_times` — Specifies the time range when a particular service may be used. The time range must be stated in 24-hour format notation, `HH:MM-HH:MM`.

The `only_from` and `no_access` options can use a list of IP addresses or host names, or can specify an entire network. Like TCP wrappers, combining `xinetd` access control with the enhanced logging configuration can enhance security by blocking requests from banned hosts while verbosely record each connection attempt.

For example, the following `/etc/xinetd.d/telnet` file can be used to block Telnet access from a particular network group and restrict the overall time range that even allowed 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 client system from the 10.0.1.0/24 network, such as 10.0.1.2, tries access the Telnet service, it will receive a message stating the following message:

```
Connection closed by foreign host.
```

In addition, their login attempt is logged in `/var/log/secure` as follows:

```
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 boo xinetd[16252]: EXIT: telnet status=0 pid=16256
```

When using TCP wrappers in conjunction with `xinetd` access controls, it is important to understand the relationship between the two access control mechanisms.

The following is the order of operations followed by `xinetd` when client requests a connection:

1. The `xinetd` daemon accesses the TCP wrappers hosts access rules through a `libwrap.a` library call. If a deny rule matches the client host, the connection is dropped. If an allow rule matches the client host, the connection is passed on to `xinetd`.
2. The `xinetd` daemon checks its own access control rules both for the `xinetd` service and the requested service. If a deny rule matches the client host the connection is dropped. Otherwise, `xinetd` starts an instance of the requested service and passes control of the connection to it.

Important

Care should be taken when using TCP wrappers access controls in conjunction with `xinetd` access controls. Misconfiguration can cause undesired effects.

15.4.3.3. Binding and Redirection Options

The service configuration files for `xinetd` support binding the service to an IP address and redirecting incoming requests for that service to another IP address, hostname, or port.

Binding is controlled with the `bind` option in the service-specific configuration files and links the service to one IP address on the system. Once configured, the `bind` option only allows requests for the proper IP address to access the service. This way different services can be bound to different network interfaces based on need.

This is particularly useful for systems with multiple network adapters or with multiple IP addresses configured. On such a system, insecure services, like Telnet, can be configured to listen only on the interface connected to a private network and not to the interface connected with the Internet.

The `redirect` option accepts an IP address or hostname followed by a port number. It configures the service to redirect any requests for this service to the specified host and port number. 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.

But the advantages of the `bind` and `redirect` options are most clearly evident 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, an internal system can be used to provide services for a totally different network. Alternatively, these options can be used to limit the exposure of a particular service on a multi-homed 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 Telnet service:

```
service telnet
{
        socket_type  = stream
        wait   = no
        server   = /usr/sbin/in.telnetd
```

```
        log_on_success  += DURATION USERID
        log_on_failure  += USERID
        bind                    = 123.123.123.123
        redirect                = 10.0.1.13 21 23
}
```

The `bind` and `redirect` options in this file ensures that the Telnet 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 Telnet service sent to 123.123.123.123 are 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 then send the communication between the two systems, and the connecting system thinks it is connected to 123.123.123.123 when 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.

15.4.3.4. Resource Management Options

The `xinetd` daemon can add a basic level of protection from a Denial of Service (DoS) attacks. Below is a list of directives which can aid in limiting the effectiveness of such attacks:

- `per_source` — Defines the maximum number of instances for a service per source IP address. It accepts only integers as an argument and can be used in both `xinetd.conf` and in the service-specific configuration files in the `xinetd.d/` directory.
- `cps` — Defines the maximum of connections per second. This directive takes two integer arguments separated by white space. The first is the maximum number of connections allowed to the service per second. The second is the number of seconds `xinetd` must wait before re-enabling the service. It accepts only integers as an argument and can be used in both `xinetd.conf` and in the service-specific configuration files in the `xinetd.d/` directory.
- `max_load` — Defines the CPU usage threshold for a service. It accepts a floating point number argument.

There more resource management options available for `xinetd`. See the chapter titled *Server Security* in the *Red Hat Linux Security Guide* for more information. Also consult the `xinetd.conf` man page.

15.5. Additional Resources

Additional information concerning TCP wrappers and `xinetd` is available from system documentation and on the Web.

15.5.1. 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 service-specific configuration files in the /etc/xinetd.d/ directory.
- man 5 hosts_access — The man page for the TCP wrappers hosts access control files.
- man hosts_options — The man page for the TCP wrappers options fields.
- man xinetd.conf — The man page listing xinetd configuration options.
- man xinetd — The man page for the xinetd super service daemon.

15.5.2. 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.

15.5.3. Related Books

- *Red Hat Linux Security Guide* ; Red Hat, Inc. — Provides an overview of workstation, server, and network security with specific suggestions regarding TCP wrappers and xinetd.
- *Hacking Linux Exposed* by Brian Hatch, James Lee, and George Kurtz; Osbourne/McGraw-Hill — An excellent security resource with featuring information about TCP wrappers and xinetd.

Chapter 16.

iptables

Installed with Red Hat Linux are advanced tools for network *packet filtering* — the process of controlling network packets as they enter, move through, and exit the network stack within the kernel. Pre-2.4 kernels relied on `ipchains` for packet filtering and used lists of rules applied to packets at each step of the filtering process. The introduction of the 2.4 kernel brought with it `iptables` (also called *netfilter*), which is similar to `ipchains` but greatly expands the scope and control available for filtering network packets.

This chapter focuses on packet filtering basics, defines the differences between `ipchains` and `iptables`, explains various options available with `iptables` commands, and shows how filtering rules can be preserved between system reboots.

For instructions on constructing `iptables` rules or setting up a firewall based on these rules, refer to Section 16.5 *Additional Resources*.

Warning

The default firewall mechanism under the 2.4 kernel is `iptables`, but `iptables` cannot be used if `ipchains` are already running. If `ipchains` are present at boot time, the kernel will issue an error and fail to start `iptables`.

The functionality of `ipchains` is not affected by these errors.

16.1. Packet Filtering

Traffic moves through a network in *packets*. A network packet is a collection of data in a specific size and format. In order to transmit a file over a network, the sending computer must first break the file into packets using the rules of the network protocol. Each of these packets holds a small part of the file data. Upon receiving the transmission, the target computer reassembles the packets into the file.

Every packet contains information which helps it navigate the network and move toward 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. For example, the *Transmission Control Protocol (TCP)* uses a SYN packet, which contains no data, to initiate communication between two systems.

The Linux kernel has the built-in ability to filter packets, allowing some of them into the system while stopping others. The 2.4 kernel's netfilter has three built-in *tables* or *rules lists*. They are as follows:

- `filter` — The default table for handling network packets.
- `nat` — Used to alter packets that create a new connection.
- `mangle` — Used for specific types of packet alteration.

Each of these tables in turn have a group of built-in *chains* which correspond to the actions performed on the packet by the netfilter.

The built-in chains for the `filter` table are as follows:

- *INPUT* — Applies to network packets that are targeted for the host.
- *OUTPUT* — Applies to locally-generated network packets.

- *FORWARD* — Applies to network packets routed through the host.

The built-in chains for the `nat` table are as follows:

- *PREROUTING* — Alters network packets when they arrive.
- *OUTPUT* — Alters locally-generated network packets before they are sent out.
- *POSTROUTING* — Alters network packets before they are sent out.

The built-in chains for the `mangle` table are as follows:

- *INPUT* — Alters network packets targeted for the host.
- *OUTPUT* — Alters locally-generated network packets before they are sent out.
- *FORWARD* — Alters network packets routed through the host.
- *PREROUTING* — Alters incoming network packets before they are routed.
- *POSTROUTING* — Alters network packets before they are sent out.

Every network packet received by or sent out of a Linux system is subject to at least one table.

A packet may be checked against multiple rules within each table 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 in one of the tables, a *target* or action is 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, that packet is refused access to the system and nothing is sent back to the host that sent the packet. If a rule specifies a `QUEUE` target, the packet to be passed to user-space. If a rule specifies the optional `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`, `REJECT`, or `QUEUE`. If none of the rules in the chain apply to the packet, then the packet is dealt with in accordance with the default policy.

The `iptables` command configures these tables, as well as sets up new tables if necessary.

16.2. Differences between `iptables` and `ipchains`

At first glance, `ipchains` and `iptables` appear to be quite similar. 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` offers a much more extensible way of filtering packets, giving the administrator a greater amount of control without building a great deal of complexity into the 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 processed using rules from only one chain rather than multiple chains.* For instance, 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, place the rule designed to catch a particular packet in the rule that will actually see the packet.
- *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. This target must be changed to DROP under `iptables`.

- *Order matters when placing options in a rule.* Previously, with `ipchains`, the order of the rule options did not matter. The `iptables` command uses stricter syntax. For example, in `iptables` commands the protocol (ICMP, TCP, or UDP) must be specified before the source or destination ports.
- *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 FORWARD or OUTPUT chains.* This is necessary because OUTPUT chains are no longer used by incoming interfaces, and INPUT chains are not seen by packets moving through outgoing interfaces.

This is not a comprehensive list of the changes, given that `iptables` represents a fundamentally rewritten network filter. For more specific information, refer to the *Linux 2.4 Packet Filtering HOWTO* found in Section 16.5 *Additional Resources*.

16.3. Options Used in `iptables` Commands

Rules that allow packets to be filtered by the kernel are put in place by running the `iptables` command. When using the `iptables` command, specify the following options:

- *Packet Type* — Dictates what type of packets the command filters.
- *Packet Source/Destination* — Dictates which packets the command filters based on the source or destination of the packet.
- *Target* — Dictates what action is taken on packets matching the above criteria.

The options used with given `iptables` rule must be grouped logically, based on the purpose and conditions of the overall rule, in order for the rule to be valid.

16.3.1. Tables

A powerful aspect of `iptables` is that multiple tables can be used to decide the fate of a particular packet. Thanks to the extensible nature of `iptables`, specialized tables can be created and stored in the `/lib/modules/<kernel-version>/kernel/net/ipv4/netfilter/` directory, where *<kernel-version>* corresponds to the version kernel number.

The default table, named `filter`, contains the standard built-in INPUT, OUTPUT, and FORWARD chains. This is 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 alters packets in specialized ways.

Each table contains default chains that perform necessary tasks based on the purpose of the table, although new chains can be added to any table.

16.3.2. Structure

Many `iptables` commands have 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 dictates a specific action to perform, such as appending or deleting the rule specified by the *<chain-name>* option. Following the *<chain-name>* are pairs of parameters and options that define 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. When creating `iptables` commands it is helpful to recognize that some parameters and options may create the need for 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.

16.3.3. Commands

Commands tell `iptables` to perform a specific action. Only one command is allowed per `iptables` command string. With the exception of the help command, all commands are written in upper-case characters.

The `iptables` commands are as follows:

- `-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.
- `-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 command structures, as well as a quick summary of command parameters and options.
- `-I` — Inserts a rule in a chain at a point specified by a user-defined integer value. If no number is specified, `iptables` will place the command at the top of the chain.

  Caution

  Be aware of which option (`-A` or `-I`) is used when adding a rule. The order of the rules within a chain are important for determining which rules apply to which packets.

- `-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 16.3.7 *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. The rule's number must be specified after the chain's name. The first rule in a chain corresponds to rule number one.
- `-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.

16.3.4. 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 construct a 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, the following IP address/netmask formats are supported:
  - *N.N.N.N/M.M.M.M* — Where *N.N.N.N* is the IP address range and *M.M.M.M* is the netmask.
  - *N.N.N.N/M* — Where *N.N.N.N* is the IP address range and *M* is the netmask.

- -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. 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 also supports the following special options:
  - ! — 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 which match a particular string. For example, the parameter -i eth+ would apply this rule to any Ethernet interfaces 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.

  You may also direct a packet matching this rule to a user-defined chain outside of the current chain so that other rules can be applied to the packet.

  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 one, as the packet matched the specified rule.
- -o — Sets the outgoing network interface for a 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 supported protocol. In addition, any protocols listed in /etc/protocols may also be used. 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.

16.3.5. 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, by using `-p tcp <protocol-name>` (where `<protocol-name>` is the target protocol), to make the options for that protocol available.

16.3.5.1. TCP Protocol

These match options are available for the TCP protocol (`-p tcp`):

- `--dport` — Sets the destination port for the packet. 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. The `--destination-port` match option is synonymous with `--dport`.

  To specify a specific range of port numbers, separate the two numbers with a colon (`:`), such as `-p tcp --dport 3000:3200`. The largest acceptable valid range is `0:65535`.

  Use an exclamation point character (`!`) 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`. The `--source-port` match option is synonymous with `--sport`.

- `--syn` — Applies to all TCP packets designed to initiate communication, commonly called *SYN packets*. Any packets that carry a data payload are not 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. The first parameter is the mask, which sets the flags to be examined in the packet. The second parameter refers to the flag that must be set in order to match.

  The possible flags are:

  - `ACK`
  - `FIN`
  - `PSH`
  - `RST`
  - `SYN`
  - `URG`
  - `ALL`
  - `NONE`

  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.

  Using the exclamation point character (`!`) after `--tcp-flags` reverses the effect of the match option.

- `--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 (`!`).

16.3.5.2. 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 is synonymous with --dport. Refer to the --dport match option in Section 16.3.5.1 *TCP Protocol* for 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 is synonymous with --sport. Refer to the --sport match option in Section 16.3.5.1 *TCP Protocol* for ways to use this option.

16.3.5.3. ICMP Protocol

These match options are available for the Internet Control Message Protocol (ICMP) (-p icmp):

- --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.

16.3.5.4. Modules with Additional Match Options

Additional match options are also available through modules loaded by the iptables command. To use a match option module, load the module by name using the -m option, such as -m *<module-name>* (replacing *<module-name>* with the name of the module).

A large number of modules are available by default. It is even possible to create your own modules to provide additional match option functionality.

Many modules exist, but only the most popular modules are discussed here.

- limit module — Allows limit to be placed on how many packets are matched to a particular rule. This is especially beneficial when logging rule matches so that a flood of matching packets will not fill up the system logs with repetitive messages or use up system resources.

  The limit module enables the following options:

  - --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.

- state module — Enables state matching.

  The state module enables the following options:

  - --state — match a packet with the following 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.

- mac module — Enables hardware MAC address matching.

  The mac module enables the following option:

  - --mac-source — Matches a MAC address of the network interface card that sent the packet. 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, refer to the iptables man page.

16.3.6. 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. 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.

The following are the standard targets:

- *<user-defined-chain>* — Replace *<user-defined-chain>* with the name of a user-defined chain within the table. This target passes the packet to the target chain.
- ACCEPT — Allows the packet to successfully move on to its destination or another chain.
- DROP — Drops the packet without responding to the requester. The system that sent the packet is not notified of the failure.
- QUEUE — The packet is queued for handling by a user-space application.
- 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 action to take.

In addition to these standard targets, various other targets may be used with extensions called *target modules*. For more information about match option modules, see Section 16.3.5.4 *Modules with Additional Match Options*.

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 of a logging event. A list of priority levels can be found in the syslog.conf man page.

- `--log-ip-options` — Any options set in the header of a IP packet is logged.
- `--log-prefix` — Places a string of up to 29 characters before the log line when it is written. This is 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 are logged.
- `--log-tcp-sequence` — Writes the TCP sequence number for the packet in the log.

- `REJECT` — Sends an error packet back to the remote system and drops the packet.

  The `REJECT` target accepts `--reject-with` <*type*> (where <*type*> is the rejection type) which allows more detailed information to be sent back 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 for IP masquerading using the `nat` table or with packet alteration using the `mangle` table, can be found in the `iptables` man page.

16.3.7. Listing Options

The default list command, `iptables -L`, provides a very basic overview of the default filter table's current chains. Additional options provide more information:

- `-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.
- `-t` — Specifies a table name.

16.4. Storing `iptables` Information

Rules created with the `iptables` command are stored in memory. If the system is restarted after setting up `iptables` rules, they will be lost. In order for netfilter rules to persist through system reboot, they need to be saved. To do this, log in as root and type:

```
/sbin/service iptables save
```

This executes the `iptables` init script, which runs the `/sbin/iptables-save` program and writes the current `iptables` configuration to the `/etc/sysconfig/iptables`. This file should only be readable by root.

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 provides a quick way to distribute sets of iptables rules to multiple machines.

Important

If distributing the /etc/sysconfig/iptables file to other machines, type /sbin/service iptables restart for the new rules take effect.

16.5. Additional Resources

Refer to the following sources for additional information on packet filtering with iptables.

16.5.1. Installed Documentation

- man iptables — Contains a comprehensive description of various commands, parameters, and other options.

16.5.2. Useful Websites

- http://netfilter.samba.org — Contains assorted information about iptables, including a FAQ addressing specific problems and various helpful guides by Rusty Russell, the Linux IP firewall maintainer. The HOWTO documents on the site cover subjects such as basic networking concepts, 2.4 kernel packet filtering, and NAT configurations.
- http://www.linuxnewbie.org/nhf/Security/IPtables_Basics.html — A basic and general look at the way packets move through the Linux kernel, plus an introduction to constructing simple iptables commands.
- http://www.redhat.com/support/resources/networking/firewall.html — This webpage links to a variety of update-to-date packet filter resources.

Chapter 17. Kerberos

Kerberos is a network authentication protocol created by MIT which uses symmetric key cryptography to authenticate users to network services — eliminating the need to send passwords over the network. When users authenticate to network services using Kerberos, unauthorized users attempting to gather passwords by monitoring network traffic are effectively thwarted.

17.1. Advantages of Kerberos

Most conventional network systems use password-based authentication schemes. Such schemes require a user to authenticate to a given network server by supplying their user name and password. Unfortunately, the transmission of authentication information for many services is unencrypted. For such a scheme to be secure, the network has to be inaccessible to outsiders, and all computers and users on the network must be trusted and trustworthy.

Even if this is the case, once a network is connected to the Internet, it can no longer be assumed that the network is secure. An attacker who gains access can use a simple packet analyzer, also known as a packet sniffer, to intercept usernames and passwords sent in this manner, compromising user accounts and the integrity of the entire security infrastructure.

The primary design goal of Kerberos is to eliminate the transmission of unencrypted passwords across the network. If used properly, Kerberos effectively eliminates the threat packet sniffers would otherwise pose on a network.

17.1.1. Disadvantages of Kerberos

Although Kerberos removes a common and severe security threat, it may be difficult to implement for a variety of reasons:

- Migrating user passwords from a standard UNIX password database, such as `/etc/passwd` or `/etc/shadow`, to a Kerberos password database can be tedious, as there is no automated mechanism to perform this task. For more information, refer to question number 2.23 in the Kerberos FAQ online at the following URL:

  http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html.

- Kerberos has only partial compatibility with the Pluggable Authentication Modules (PAM) system used by most servers running Red Hat Linux. For more information on this issue, see Section 17.4 *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 can be quite problematic due to the size of the application or its design. For other incompatible applications, changes must be made to the way in which the server and client side communicate. Again, this may require extensive programming. Closed-source applications that do not have Kerberos support by default are often the most problematic.
- Kerberos assumes that you are a trusted user using an untrusted host on an untrusted network. Its primary goal is to prevent plain text passwords from being sent across that network. However, if anyone other than the proper user has access to the one host that issues tickets used for authentication — called the *key distribution center* (*KDC*) — the entire Kerberos authentication system is at risk of being compromised.
- Kerberos is an all or nothing solution. If you decide to use Kerberos on your network, you must remember that any passwords transferred to a service which does not use Kerberos for authentica-

tion are at risk of being captured by packet sniffers. Thus, your network gains no benefit from the use of Kerberos. To secure a network with Kerberos, one must either use *kerberized* versions of *all* client/server applications which send unencrypted passwords or not use any such client/server applications at all.

17.2. Kerberos Terminology

Kerberos has its own terminology to define various aspects of the service. Before learning how kerberos works, it is important to learn the following terms.

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 supports a framework for using other cache types, such as shared memory, but files are more thoroughly supported.

crypt hash

A one way hash used to authenticate users. While more secure than plain text, it is fairly easy to decrypt for an experienced cracker.

GSS-API

The Generic Security Service Application Program Interface [RFC-2743] is a set of functions which provide security services which clients can use to authenticate to servers and which servers can use to authenticate to clients without specific knowledge of the underlying mechanism. If a network service (such as IMAP) uses GSS-API, it can authenticate using Kerberos.

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 KDC administration server, `/usr/kerberos/sbin/kadmind`, is the only service that uses any other file (it uses `/var/kerberos/krb5kdc/kadm5.keytab`).

`kinit`

The `kinit` command allows a principal who has already logged in to obtain and cache the initial Ticket Granting Ticket (TGT). For more on using the `kinit` command, see its man page.

principal

The principal is the unique name of a user or service that can authenticates 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 considered 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 for users is derived from a password or is randomly set for services.

realm

A network that uses Kerberos, composed of one or more servers called KDCs and a potentially large number of clients.

service

A program 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)

A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC

Ticket Granting Ticket (TGT)

A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

unencrypted password

A plain text, human-readable password.

17.3. How Kerberos Works

Kerberos differs from other authentication methods. Instead of authenticating each user to each network service, Kerberos uses symmetric encryption and a trusted third party — known as the Key Distribution Center (KDC) — to authenticate users to a suite of network services. Once a user authenticates to the KDC, it sends a ticket specific to that session back the user's machine and any kerberized service will look for the ticket on the user's machine rather than asking the user to authenticate using a password.

When a user on a kerberized network logs in to their workstation, their principal is sent to the KDC in a request for a Ticket Granting Ticket (TGT) from the Ticket Granting Service (TGS). 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 tell the TGS to create a TGT, which is encrypted using the user's key and returned to that user.

The login or `kinit` program on the client machine then decrypts the TGT using the user's key (which it computes from the user's password). The user's key is used only on the client machine and is *not* sent over the network.

The TGT is set to expire after a certain period of time (usually ten hours) and stored in the client machine's credentials cache. An expiration time is set so that a compromised TGT is of use to an attacker for only a short period of time. Once the TGT is issued, the user will not have to re-enter their password to the KDC until the TGT expires or they logout and login again.

Whenever the user needs access to a network service, the client software uses the TGT to request a new ticket for that specific service from the TGS. The service ticket is then used to authenticate the user to that service transparently.

Warning

The Kerberos system can be compromised any time any user on the network authenticates against a non-kerberized service by sending a password in plain text. Therefore use of non-kerberized services is discouraged. Such services include Telnet and FTP. Use of other encrypted protocols, such as SSH or SSL secured services, however, is acceptable, though not ideal.

This is only a broad overview of how Kerberos authentication on a network works, those seeking a more in-depth look at Kerberos authentication, should refer to Section 17.7 *Additional Resources*.

Note

Kerberos depends on certain network services to work correctly. First, Kerberos requires approximate clock synchronization between the machines on the network. Therefore, a clock synchronization program should be set up for the network, such as `ntpd`. For more on configuring `ntpd`, see `/usr/share/doc/ntp-`<*version-number*>`/index.htm` for details on setting up Network Time Protocol servers.

Also, since certain aspects of Kerberos rely on the Domain Name Service (DNS), be sure that the DNS entries and hosts on the network are all properly 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.

17.4. Kerberos and PAM

Currently, kerberized services do not make use of Pluggable Authentication Modules (PAM) — kerberized servers bypass PAM completely. However, applications that use PAM can make use of Kerberos for authentication if the `pam_krb5` module (provided in the `pam_krb5` package) is installed. The `pam_krb5` package contains sample configuration files that allow services like `login` and `gdm` to authenticate users and obtain initial credentials using their passwords. If access to network servers is always performed using kerberized services or services that use GSS-API, such as IMAP, the network can be considered reasonably safe.

Administrators should be careful to not allow users to authenticate to most network services using Kerberos passwords. Many protocols used by these services do not encrypt the password before sending it over the network, destroying the benefits of the Kerberos system. For example, users should not be allowed to authenticate using their Kerberos passwords over Telnet.

The next section describes how to set up a basic Kerberos server.

17.5. Configuring a Kerberos 5 Server

When you are setting up Kerberos, install the server first. If you need to set up slave servers, the details of setting up relationships between master and slave servers are covered in the *Kerberos 5 Installation Guide* located in the `/usr/share/doc/krb5-server-`<*version-number*> directory.

To configure a basic Kerberos server, follow these steps:

1. Be sure that you have clock synchronization and DNS working on your server before configuring Kerberos 5. Pay particular attention to time synchronization between the Kerberos server and its various clients. If the server and client clocks are different by more than five minutes (this default amount is configurable in Kerberos 5), Kerberos clients will not be able to authenticate to the server. This clock synchronization is necessary to prevent an attacker from using an old Kerberos ticket to masquerade as a valid user.

   You should set up a Network Time Protocol (NTP) compatible client/server network even if you are not using Kerberos. Red Hat Linux includes the `ntp` package for easy installation. See /usr/share/doc/ntp-<*version-number*>/index.htm for details on setting up Network Time Protocol servers and http://www.eecis.udel.edu/~ntp for additional information on NTP.

2. Install the `krb5-libs`, `krb5-server`, and `krb5-workstation` packages on the dedicated machine which will run the KDC. This machine needs to be very secure — if possible, it should not run any services other than the KDC.

   If you would like to use a graphical user interface utility to administrate Kerberos, you should also install the `gnome-kerberos` package. It contains `krb5`, a GUI tool for managing tickets.

3. Edit the `/etc/krb5.conf` and `/var/kerberos/krb5kdc/kdc.conf` configuration files to reflect your realm name and domain-to-realm mappings. A simple realm can be constructed by replacing instances of *EXAMPLE.COM* and *example.com* with your domain name — being certain to keep uppercase and lowercase names in the correct format — and by changing the KDC from *kerberos.example.com* to the name of your Kerberos server. By convention, all realm names are uppercase and all DNS hostnames and domain names are lowercase. For full details on the formats of these files, see their respective man pages.

4. Create the database using the `kdb5_util` utility from a shell prompt:

   ```
   /usr/kerberos/sbin/kdb5_util create -s
   ```

   The `create` command creates the database that will be used to store keys for your Kerberos realm. The `-s` switch forces creation of a *stash* file in which the master server key is stored. If no stash file is present from which to read the key, the Kerberos server (`krb5kdc`) will prompt the user for the master server password (which can be used to regenerate the key) every time it starts.

5. Edit the `/var/kerberos/krb5kdc/kadm5.acl` file. This file is used by `kadmind` to determine which principals have administrative access to the Kerberos database and their level of access. Most organizations will be able to get by with a single line:

   ```
   */admin@EXAMPLE.COM  *
   ```

   Most users will be represented in the database by a single principal (with a *NULL*, or empty, instance, such as *joe@EXAMPLE.COM*). With this configuration, users with a second principal with an instance of *admin* (for example, *joe/admin@EXAMPLE.COM*) will be able to wield full power over the realm's Kerberos database.

   Once `kadmind` is started on the server, any user will be able to access its services by running `kadmin` on any of the clients or servers in the realm. However, only users listed in the `kadm5.acl` file will be able to modify the database in any way, except for changing their own passwords.

   Note

   The `kadmin` utility communicates with the `kadmind` server over the network, and they use Kerberos to handle authentication. Of course, you need to create the first principal before you can connect to the server over the network to administer it. Create the first principal with the `kadmin.local` command, which is specifically designed to be used on the same host as the KDC and does not use Kerberos for authentication.

   Type the following `kadmin.local` command at the KDC terminal to create the first principal:

```
/usr/kerberos/sbin/kadmin.local -q "addprinc username/admin"
```

6. Start Kerberos using the following commands:

   ```
   /sbin/service krb5kdc start
   /sbin/service kadmin start
   /sbin/service krb524 start
   ```

7. Add principals for your users using the `addprinc` command with `kadmin`. `kadmin` and `kadmin.local` are command line interfaces to the KDC. As such, many commands are available after launching the `kadmin` program. See the `kadmin` man page for more information.
8. Verify that your server will issue tickets. First, run `kinit` to obtain a ticket and store it in a credential cache file. Next, use `klist` to view the list of credentials in your cache and use `kdestroy` to destroy the cache and the credentials it contains.

   Note

   By default, `kinit` attempts to authenticate using the login user name of the account you used when you first logged into your system (not the Kerberos server). If that system user name does not correspond to a principal in the Kerberos database, you will receive an error message. If that happens, supply `kinit` with the name of your principal as an argument on the command line (`kinit` *`principal`*).

Once you have completed the steps listed above, the Kerberos server should be up and running. Next, we will set up a Kerberos client.

17.6. Configuring a Kerberos 5 Client

Setting up a Kerberos 5 client is less involved than setting up a server. At a minimum, install the client packages and provide each client with a valid `krb5.conf` configuration file. Kerberized versions of `rsh` and `rlogin` will also require some configuration changes.

1. Be sure that you have time synchronization in place between the Kerberos client and the KDC. See Section 17.5 *Configuring a Kerberos 5 Server* for more information. In addition, verify that DNS is working properly on the Kerberos client before configuring the Kerberos client programs.
2. Install the `krb5-libs` and `krb5-workstation` packages on all of the client machines. You must supply a version of `/etc/krb5.conf` for each client; usually this can be the same `krb5.conf` file used by the KDC.
3. Before a workstation in the realm can allow users to connect using kerberized `rsh` and `rlogin`, that workstation will need to have the `xinetd` package installed and have its own host principal in the Kerberos database. The `kshd` and `klogind` server programs will also need access to the keys for their service's principal.

   Using `kadmin`, add a host principal for the workstation on the KDC. The instance in this case will be the hostname of the workstation. You can use the `-randkey` option to `kadmin`'s `addprinc` command to create the principal and assign it a random key:

   ```
   addprinc -randkey host/blah.example.com
   ```

   Now that you have created the principal, you can extract the keys for the workstation by running `kadmin` *on the workstation itself*, and using the `ktadd` command within `kadmin`:

   ```
   ktadd -k /etc/krb5.keytab host/blah.example.com
   ```

4. If you wish to use other kerberized network service, they will need to be started. Below is a list of some of the more common kerberized services and instructions on enabling them:

- rsh and rlogin — In order to use the kerberized versions of rsh and rlogin, you must enable klogin, eklogin, and kshell.
- Telnet — To use kerberized Telnet, you must enable krb5-telnet.
- FTP — To provide FTP access, create and extract a key for the principal with a root of ftp. Be certain to set the instance to the fully qualified hostname of the FTP server, then enable gssftp.
- IMAP — The IMAP server included in the imap package will use GSS-API authentication using Kerberos 5 if it finds the proper key in /etc/krb5.keytab. The root for the principal should be imap.
- CVS — CVS's kerberized gserver uses a principal with a root of cvs and is otherwise identical to the CVS pserver.

For details on enabling services, refer to the chapter titled *Controlling Access to Services* in the *Red Hat Linux Customization Guide*.

17.7. Additional Resources

For more information on Kerberos, refer to the following resources.

17.7.1. 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. You must have the krb5-server package installed.
- /usr/share/doc/krb5-workstation-<*version-number*> — The *Kerberos V5 UNIX User's Guide* in PostScript and HTML formats. You must have the krb5-workstation package installed.

17.7.2. Useful Websites

- http://web.mit.edu/kerberos/www — *Kerberos: The Network Authentication Protocol* webpage from MIT.
- 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 — The 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 — *How to Kerberize your site* is a good reference for kerberizing a network.
- http://www.networkcomputing.com/netdesign/kerb1.html — *Kerberos Network Design Manual* is a thorough overview of the Kerberos system.

Chapter 18. SSH Protocol

SSH™ allows users to log into host systems remotely. Unlike protocols such as FTP or Telnet, SSH encrypts the login session, making it impossible for intruders to collect unencrypted passwords.

SSH is designed to replace older, less secure terminal applications used to log into remote hosts, such as **telnet** or **rsh**. A related program called **scp** replaces older programs designed to copy files between hosts, such as **rcp**. Because these older applications do not encrypt passwords transmitted between the client and the server, avoid them whenever possible. Using secure methods to remotely log into remote systems decreases the risks for both the client system and the remote host.

18.1. Features of SSH

SSH (or *S*ecure *SH*ell) is a protocol which facilitates secure communications between two systems using a client/server architecture.

The SSH protocol provides the following safeguards:

- After an initial connection, the client can verify that it is connecting to the same server it connected to previously.
- The client transmits its authentication information to the server using strong, 128 bit encryption.
- All data sent and received during a session is transferred using 128 bit encryption, making intercepted transmissions extremely difficult to decrypt and read.
- The client can forward X11 [1] applications from the server. This technique, called *X11 forwarding*, provides a secure means to use graphical applications over a network.

Because the SSH protocol encrypts everything it sends and receives, it can be used to secure otherwise insecure protocols. Using a technique called *port forwarding*, an SSH server can become a conduit to secure otherwise insecure protocols, like POP, increasing overall system and data security.

Red Hat Linux includes the general OpenSSH package (`openssh`), the OpenSSH server (`openssh-server`) and client (`openssh-clients`) packages. Please see the chapter titled *OpenSSH* in the *Red Hat Linux Customization Guide* for instructions on installing and deploying OpenSSH. Also note that the OpenSSH packages require the OpenSSL package (`openssl`). OpenSSL installs several important cryptographic libraries which enable OpenSSH to provide encrypted communications.

A large number of client and server programs can use the SSH protocol. SSH client applications are available for almost every major operating system in use today.

18.1.1. Why Use SSH?

Nefarious computer users have a variety of tools at their disposal which enable them to disrupt, intercept, and re-route network traffic in an effort to gain access to a system. In general terms, these threats can be categorized as:

- *Interception of communication between two systems* — In this scenario, the attacker can be somewhere on the network between the communicating entities, copying any information passed between them. The attacker may intercept and keep the information or alter the information and send it on to the intended recipient.

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.

This attack can be mounted through the use of a packet sniffer — a common network utility.

- *Impersonation of a particular host* — Using this strategy, an attacker's system is configured to pose as the intended recipient of a transmission. If this strategy works, the user's system will remain unaware it is communicating with the wrong host.

  This attack can be mounted through techniques known as DNS poisoning [2] or IP spoofing [3].

Both techniques intercept potentially sensitive information, and if the interception is for hostile reasons, the results can be disastrous.

If SSH is used for remote shell login and file copying, these security threats can be greatly diminished. This is because the SSH client and server use digital signatures to verify their identity. Additionally, all communication between the client and server systems 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.

18.2. SSH Protocol Versions

The SSH protocol allows any client and server programs built to the protocol's specifications to communicate securely and to be used interchangeably.

Two varieties of SSH currently exist. SSH version 1 makes use of several patented encryption algorithms (however, some of these patents have expired) and is vulnerable to a security hole that potentially allows an attacker to insert data into the communication stream. The OpenSSH suite under Red Hat Linux uses SSH version 2 by default, although it also supports version 1.

Important

It is recommended that only SSH version 2-compatible servers and clients are used whenever possible.

18.3. Event Sequence of an SSH Connection

The following series of events help protect the integrity of SSH communication between two hosts.

- A cryptographic handshake is made so that the client can verify that it is communicating with the correct server.
- The transport layer of the connection between client and remote host is encrypted using a symmetric cipher.
- The client authenticates itself to the server.
- the remote client can now interact safely with the remote host over the encrypted connection.

---

2. DNS poisoning occurs when an intruder cracks a DNS server, pointing client systems to a maliciously duplicated host.
3. IP spoofing occurs when an intruder sends network packets which falsely appear to be from a trusted host on the network.

18.3.1. 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. The transport layer accomplishes this by handling the encryption and decryption of data and providing integrity protection of data packets as they are sent and received. In addition, the transport layer provides compression, speeding the transfer of information.

Once an SSH client contacts a server, key information is exchanged so that the two systems can correctly construct the transport layer. The following steps occur during this exchange:

- Keys are exchanged
- The public key encryption algorithm is determined
- The symmetric encryption algorithm is determined
- The message authentication algorithm is determined
- The hash algorithm to be used is determined

During the key exchange, the server identifies itself to the client with a unique *host key*. If the client has never communicated with this particular server before, the server's key will be unknown to the client and it will not connect. OpenSSH gets around this problem by accepting the server's host key after the user is notified and verifies the acceptance of the new host key. In subsequent connections, the server's host key is checked against the saved version on the client, providing confidence that the client is indeed communicating with the intended server. If, in the future, the host key no longer matches, the user must remove the client's saved version before a connection can occur.

Caution

> It is possible for an attacker to masquerade as the SSH server during the initial contact since the local system does not know the difference between the intended server and a false one set up by an attacker. To help prevent this, verify the integrity of a new SSH server by contacting the server administrator before connecting for the first time or in the event of a host key mismatch.

SSH is designed to work with almost any kind of public key algorithm or encoding format. After an initial key exchange creates 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.

After a certain amount of data has been transmitted using a given key and algorithm (the exact amount depends on the SSH implementation), another key exchange occurs, which generates another set of hash values and a new shared secret value. Even if an attacker is able to determine the hash and shared secret value, this information would be useful for only a limited period of time.

18.3.2. 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 then tries to authenticate itself to the server using one of these supported methods.

SSH servers and clients can be configured to allow different types of authentication, which 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 and password-based authentication, are safe to use.

18.3.3. Channels

After a successful authentication over the SSH transport layer, multiple *channels* are opened via a technique called multiplexing[4]. Each of these channels handles communication for different terminal sessions and for forwarded X11 sessions.

Both clients and servers can create a new channel. Each channel is then assigned a different number on each end of the connection. When the client attempts to open a new channel, the clients sends the channel number along with the request. This information is stored by the server and is used to direct communication to that channel. This is done so that different types of sessions will not affect one another and so that when a given session ends, its channel can be closed without disrupting the primary SSH connection.

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 client receives a message that the channel is open.

The client and server negotiate the characteristics of each channel automatically, depending on the type of service the client requests and the way the user is connected to the network. This allows great flexibility in handling different types of remote connections without having to change the basic infrastructure of the protocol.

18.4. OpenSSH Configuration Files

OpenSSH has two different sets of configuration files: one for client programs (`ssh`, `scp`, and `sftp`) and one for the server daemon (`sshd`).

System-wide SSH configuration information is stored in the `/etc/ssh/` directory:

- `moduli` — Contains Diffie-Hellman groups used for the Diffie-Hellman key exchange which is critical for constructing a secure transport layer. When keys are exchanged at the beginning of an SSH session, a shared, secret value is created which cannot be determined by either party alone. This value is then used to provide host authentication.
- `ssh_config` — The system-wide default SSH client configuration file. It is overridden if one is also present in the user's home directory (`~/.ssh/config`).
- `sshd_config` — The configuration file for the `sshd` daemon.
- `ssh_host_dsa_key` — The DSA private key used by the `sshd` daemon.
- `ssh_host_dsa_key.pub` — The DSA public key used by the `sshd` daemon.
- `ssh_host_key` — The RSA private key used by the `sshd` daemon for version 1 of the SSH protocol.
- `ssh_host_key.pub` — The RSA public key used by the `sshd` daemon for version 1 of the SSH protocol.
- `ssh_host_rsa_key` — The RSA private key used by the `sshd` daemon for version 2 of the SSH protocol.
- `ssh_host_rsa_key.pub` — The RSA public key used by the `sshd` for version 2 of the SSH protocol.

User-specific SSH configuration information is stored in the user's home directory within the `~/.ssh/` directory:

4. 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.

- `authorized_keys` — This file holds a list of authorized public keys for servers. When the client connects to a server, the server authenticates the client by checking its signed public key stored within this file.
- `id_dsa` — Contains the DSA private key of the user.
- `id_dsa.pub` — The DSA public key of the user.
- `id_rsa` — The RSA private key used by `ssh` for version 2 of the SSH protocol.
- `id_rsa.pub` — The RSA public key used by `ssh` for version 2 of the SSH protocol
- `identity` — The RSA private key used by `ssh` for version 1 of the SSH protocol.
- `identity.pub` — The RSA public key used by `ssh` for version 1 of the SSH protocol.
- `known_hosts` — This file contains DSA host keys of SSH servers accessed by the user. This file is very important for ensuring that the SSH client is connecting the correct SSH server.

Important

If a SSH server's host key has changed, the client will notify the user that the connection cannot proceed until the server's host key is deleted from the `known_hosts` file using a text editor. Before doing this, however, contact the system administrator of the SSH server to verify the server is not compromised.

See the man pages for `ssh` and `sshd` for information concerning the various directives available in the SSH configuration files.

18.5. 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.

18.5.1. X11 Forwarding

Opening an X11 session over an established SSH connection is as easy as running an X program on a local machine. When an X program is run from the secure shell prompt, the SSH client and server create a new secure channel, and the X program data is sent over that channel to the client machine transparently.

X11 forwarding can be very useful. For example, X11 forwarding can be used to create a secure, interactive session with `up2date`. To do this, connect to the server using **ssh** and type:

```
up2date &
```

After supplying the root password for the server, the **Red Hat Update Agent** will appear and allow the the remote user to safely update the remote system.

18.5.2. Port Forwarding

SSH can secure otherwise insecure TCP/IP protocols via port forwarding. When using this technique, the SSH server becomes an encrypted conduit to the SSH client.

Port forwarding works by mapping a local port on the client to a remote port on the server. SSH can map any port from the server to any port on the client, and the port numbers do not need to match for this technique to work.

To create a TCP/IP port forwarding channel which listens for connections on the localhost, use the following command:

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

Note

Setting up port forwarding to listen on ports below 1024 requires root access.

To check email on a server called mail.example.com using POP through an encrypted connection, use the following command:

```
ssh -L 1100:mail.example.com:110 mail.example.com
```

Once the port forwarding channel is in place between the client machine and the mail server, direct a POP mail client to use port 1100 on the localhost to check for new mail. Any requests sent to port 1100 on the client system will be directed securely to the mail.example.com server.

If mail.example.com is not running an SSH server, but another machine on the same network is, SSH can still be used to secure part of the connection. However, a slightly different command is necessary:

```
ssh -L 1100:mail.example.com:110 other.example.com
```

In this example, POP requests from port 1100 on the client machine are forwarded through the SSH connection on port 22 to the SSH server, `other.example.com`. Then, `other.example.com` connects to port 110 on `mail.example.com` to check for new mail. Note that by using this technique, only the connection between the client system and `other.example.com` SSH server is secure.

Port forwarding can also be used to get information securely through network firewalls. If the firewall is configured to allow SSH traffic via its standard port (22) but blocks access to other ports, a connection between two hosts using the blocked ports is still possible by redirecting their communication over an established SSH connection.

Note

Using port forwarding to forward connections in this manner allows any user on the client system to connect to that service. If the client system becomes compromised, the attacker will also have access to forwarded services.

System administrators concerned about port 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.

18.6. Requiring SSH for Remote Connections

For SSH to be truly effective, using all insecure connection protocols, such as Telnet and FTP should be prohibited. Otherwise, a user's password may be protected using SSH for one session, only to be captured later while logging in using Telnet.

Some services to disable include:

- `telnet`
- `rsh`
- `rlogin`
- `vsftpd`

To disable insecure connection methods to the system, use the command line program `chkconfig`, the ncurses-based program **ntsysv**, or the graphical application **Services Configuration Tool** (`redhat-config-services`). All of these tools require root access.

For more information on runlevels and configuring services with `chkconfig`, **ntsysv**, and **Services Configuration Tool**, refer to the chapter titled *Controlling Access to Services* in the *Red Hat Linux Customization Guide*.

Chapter 19.

Tripwire

Tripwire data integrity assurance software monitors the reliability of critical system files and directories by identifying changes made to them. It does this through an automated verification regimen run at regular intervals. If Tripwire detects that a monitored file has been changed, it notifies the system administrator via email. Because Tripwire can positively identify files that have been added, modified, or deleted, it can speed recovery from a break-in by keeping the number of files which must be restored to a minimum. These abilities make Tripwire an excellent tool for system administrators seeking both intrusion detection and damage assessment for their servers.

Tripwire works by comparing files and directories against a database of file locations, dates they were modified, and other data. This database contains *baselines* — which are snapshots of specified files and directories at a specific point in time. The contents of the baseline database should be generated before the system is at risk of intrusion, meaning before it is connected to the network. After creating the baseline database, Tripwire compares the current system to the baseline and reports any modifications, additions, or deletions.

While Tripwire is a valuable tool for auditing the security state of Red Hat Linux systems, Tripwire is not supported by Red Hat, Inc. If you need more information about Tripwire, a good place to start is the project's website located at http://www.tripwire.org.

19.1. How to Use Tripwire

The following flowchart illustrates how Tripwire works:

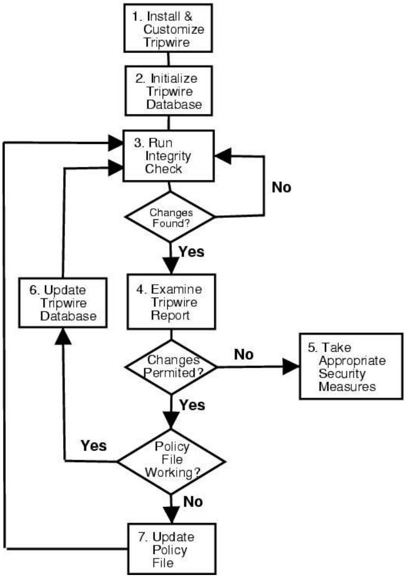

Figure 19-1. Using Tripwire

The following describes in more detail the numbered blocks shown in Figure 19-1.

1. Install Tripwire and customize the policy file.

Install the Tripwire RPM (see Section 19.2 *Installing the Tripwire RPM*). Then, customize the sample configuration and policy files (`/etc/tripwire/twcfg.txt` and `/etc/tripwire/twpol.txt` respectively), and run the configuration script, `/etc/tripwire/twinstall.sh`. For more information, see Section 19.3 *Customizing Tripwire*.

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 19.4 *Initialize the Tripwire 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 19.5 *Running an Integrity Check*.

4. Examine the Tripwire report file.

View the Tripwire report file using `/usr/sbin/twprint` to note integrity violations. For more information, see Section 19.6.1 *Viewing Tripwire Reports*.

5. If unauthorized integrity violations occur, take appropriate security measures.

If monitored files have been altered inappropriately, you can either replace the original files from backup copies, reinstall the program, or completely reinstall the operating system.

6. If the file alterations are valid, verify and update the Tripwire database file.

If the changes made to monitored files are intentional, edit Tripwire's database file to ignore those changes in subsequent reports. For more information, see Section 19.7 *Updating the Tripwire Database*.

7. If the policy file fails verification, update the Tripwire policy file.

To change the list of files Tripwire monitors or how it treats integrity violations, update the supplied policy file (`/etc/tripwire/twpol.txt`), regenerate a signed copy (`/etc/tripwire/tw.pol`), and update the Tripwire database. For more information, see Section 19.8 *Updating the Tripwire Policy File*.

Refer to the appropriate sections within this chapter for detailed instructions on each step.

19.2. Installing the Tripwire RPM

The easiest way to install Tripwire is to select the Tripwire RPM during the Red Hat Linux installation process. However, if you have already installed Red Hat Linux, you can use the `rpm` command or the **Package Management Tool** (`redhat-config-packages`) to install the Tripwire RPM from the Red Hat Linux 9 CD-ROMs.

If you are not sure whether Tripwire is installed, type the following command at a shell prompt:

```
rpm -q tripwire
```

If Tripwire is installed, this command will return the following:

```
tripwire-<version-number>
```

In the above output, <*version-number*> is the version number of the package.

If Tripwire is not installed, the shell prompt will return.

The following steps outline how to find and install Tripwire from CD-ROM using the RPM command line application:

1. Insert *CD 2* of the Red Hat Linux 9 installation CD-ROMs.
2. If the CD-ROM does not automatically mount, type the following command:
   `mount /mnt/cdrom`
3. Verify that the Tripwire RPM is on the CD-ROM by typing:
   `ls /mnt/cdrom/RedHat/RPMS/ | grep tripwire`

   If the RPM is on the CD-ROM, this command will display the package name.

   If the RPM is *not* on the CD-ROM, the shell prompt will return. In this case, you will need to check the other Red Hat Linux 9 installation CD-ROMs by first unmounting the CD-ROM and then repeating steps one through three.

   Unmount the CD-ROM by right-clicking on the CD-ROM icon and selecting **Eject** or by typing the following command at the shell prompt:
   `umount /mnt/cdrom`
4. Once you have located the Tripwire RPM, install it by typing the following command as the root user:
   `rpm -Uvh /mnt/cdrom/RedHat/RPMS/tripwire*.rpm`

You will find release notes and README files for Tripwire in the `/usr/share/doc/tripwire-`<*version-number*>`/` directory (where <*version-number*> is the version number of the software). These documents contain important information about the default policy file and other topics.

19.3. Customizing Tripwire

After you have installed the Tripwire RPM, you must complete the following steps to initialize the software:

19.3.1. Edit `/etc/tripwire/twcfg.txt`

Although you are not required to edit this sample Tripwire configuration file, you may find it necessary for your situation. For instance, you may want to alter the location of Tripwire files, customize email settings, or customize the level of detail for reports.

Below is a list of *required* user configurable variables in the `/etc/tripwire/twcfg.txt` file:

- `POLFILE` — Specifies the location of the policy file; `/etc/tripwire/tw.pol` is the default value.
- `DBFILE` — Specifies the location of the database file; `/var/lib/tripwire/$(HOSTNAME).twd` is the default value.
- `REPORTFILE` — Specifies the location of the report files. By default this value is set to `/var/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr`.
- `SITEKEYFILE` — Specifies the location of the site key file; `/etc/tripwire/site.key` is the default value.
- `LOCALKEYFILE` — Specifies the location of the local key file; `/etc/tripwire/$(HOSTNAME)-local.key` is the default value.

Important

If you edit the configuration file and leave any of the above variables undefined, the configuration file will be invalid. If this occurs, when you execute the `tripwire` command it will report an error and exit.

The rest of the configurable variables in the sample `/etc/tripwire/twcfg.txt` file are optional. These include the following:

- `EDITOR` — Specifies the text editor called by Tripwire. The default value is `/bin/vi`.
- `LATEPROMPTING` — If set to `true`, this variable configures Tripwire to wait as long as possible before prompting the user for a password, thereby minimizing the amount of time the password is in memory. The default value is `false`.
- `LOOSEDIRECTORYCHECKING` — If set to `true`, this variable configures Tripwire to report if a file within a watched directory changes, but not to report the change for the directory itself. This limits redundancy in Tripwire reports. The default value is `false`.
- `SYSLOGREPORTING` — If set to `true`, this variable configures Tripwire to report information to the syslog daemon via the "user" facility. The log level is set to `notice`. See the `syslogd` man page for more information. The default value is `false`.
- `MAILNOVIOLATIONS` — If set to `true`, this variable configures Tripwire to email a report at a regular interval regardless of whether any violations have occurred. The default value is `true`.
- `EMAILREPORTLEVEL` — Specifies the level detail for emailed reports. Valid values for this variable are `0` through `4`. The default value is `3`.
- `REPORTLEVEL` — Specifies the level detail for reports generated by the `twprint` command. This value can be overridden on the command line, but is set to `3` by default.
- `MAILMETHOD` — Specifies which mail protocol Tripwire should use. Valid values are `SMTP` and `SENDMAIL`. The default value is `SENDMAIL`.
- `MAILPROGRAM` — Specifies which mail program Tripwire should use. The default value is `/usr/sbin/sendmail -oi -t`.

After editing the sample configuration file, you will need to configure the sample policy file.

Warning

For security purposes, you should either delete or store in a secure location any copies of the plain text `/etc/tripwire/twcfg.txt` file after running the installation script or regenerating a signed configuration file. Alternatively, you can change the permissions so that it is not world readable.

19.3.2. Edit `/etc/tripwire/twpol.txt`

Although it is not required, you should edit this heavily commented sample Tripwire policy file to take into account the specific applications, files, and directories on your system. Relying on the unaltered sample configuration from the RPM may not adequately protect your system.

Modifying the policy file also increases the usefulness of Tripwire reports by minimizing false alerts for files and programs you are not using and by adding functionality, such as email notification.

Note

Notification via email is not configured by default. See Section 19.8.1 *Tripwire and Email* for more on configuring this feature.

If you modify the sample policy file after running the configuration script, see Section 19.8 *Updating the Tripwire Policy File* for instructions on regenerating a signed policy file.

Warning

For security purposes, you should either delete or store in a secure location any copies of the plain text `/etc/tripwire/twpol.txt` file after running the installation script or regenerating a signed configuration file. Alternatively, you can change the permissions so that it is not world readable.

19.3.3. Run the `twinstall.sh` Script

As the root user, type `/etc/tripwire/twinstall.sh` at the shell prompt to run the configuration script. The `twinstall.sh` script will ask you for site and local passwords. These passwords are used to generate cryptographic keys for protecting Tripwire files. The script then creates and signs these files.

When selecting the site and local passwords, you should consider the following guidelines:

- Use at least eight alphanumeric and symbolic characters for each unique password, but no more than 1023 total characters.
- Do not use quotes in a password.
- Make the Tripwire passwords completely different from the root or any other password for the system.
- Use unique passwords for both the site key and the local key.

The site key password protects the Tripwire configuration and policy files. The local key password protects the Tripwire database and report files.

Warning

There is no way to decrypt a signed file if you forget your password. If you forget the passwords, the files are unusable and you will have to run the configuration script again.

By encrypting its configuration, policy, database, and report files, Tripwire protects them from being viewed by anyone who does not know the site and local passwords. This means that, even if an intruder obtains root access to your system, they will not be able to alter the Tripwire files to hide their tracks.

Once encrypted and signed, the configuration and policy files generated by running the `twinstall.sh` script should not be renamed or moved.

19.4. Initialize the Tripwire Database

When initializing its database, Tripwire builds a collection of file system 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.

Once you finish these steps successfully, Tripwire has the baseline snapshot of your file system necessary to check for changes in critical files. After initializing the Tripwire database, you should run an initial integrity check. This check should be done prior to connecting the computer to the network and putting it into production. For further instructions, see Section 19.5 *Running an Integrity Check*.

Once Tripwire is configured to your satisfaction, you are free to place the system into production.

19.5. Running an Integrity Check

By default, the Tripwire RPM adds a shell script called `tripwire-check` to the `/etc/cron.daily/` directory. This script automatically runs an integrity check once per day.

You can, however, run a Tripwire integrity check at any time by typing the following command:

```
/usr/sbin/tripwire --check
```

During an integrity check, Tripwire compares the current state of file system objects with the properties recorded in its database. Violations are printed to the screen and an encrypted copy of the report is created in `/var/lib/tripwire/report/`. You can view the report using the `twprint` command as outlined in Section 19.6.1 *Viewing Tripwire Reports*.

If you would like to receive an email when certain types of integrity violations occur, you can configure this in the policy file. See Section 19.8.1 *Tripwire and Email* for instructions on how to set up and test this feature.

19.6. Examining Tripwire Reports

The `/usr/sbin/twprint` command is used to view encrypted Tripwire reports and databases.

19.6.1. Viewing Tripwire Reports

The `twprint -m r` command will display the contents of a Tripwire report in clear text. You must, however, tell `twprint` which report file to display.

A `twprint` command for printing Tripwire reports looks similar to the following:

```
/usr/sbin/twprint -m r --twrfile /var/lib/tripwire/report/<name>.twr
```

The `-m r` option in the command directs `twprint` to decode a Tripwire report. The `--twrfile` option directs `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
```

19.6.2. View Tripwire Databases

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`, use the following 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 man page for `twprint` for more options.

19.7. Updating the Tripwire Database

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, reinstall the program, or, if the breach is severe enough, completely reinstall the operating system.

To update the Tripwire database so it accepts valid policy violations, Tripwire first cross-references a report file against the database and then integrates into it valid violations from the report file. When updating the database, be sure to use the most recent report.

Use the following command to update the Tripwire database, where *name* is the name of the most recent report file:

```
/usr/sbin/tripwire --update --twrfile /var/lib/tripwire/report/<name>.twr
```

Tripwire will display the report file using the default text editor specified on the `EDITOR` line of the Tripwire configuration file. This gives you an opportunity to deselect files you do not wish to update in the Tripwire database.

Important

It is important that you change only *authorized* integrity violations in the database.

All proposed updates to the Tripwire database start with an `[x]` before the file name, similar to the following example:

```
Added:
[x] "/usr/sbin/longrun"

Modified:
[x] "/usr/sbin"
[x] "/usr/sbin/cpqarrayd"
```

If you want to specifically exclude a valid violation from being added to the Tripwire database, remove the `x`.

To edit files in the default text editor, `vi`, type `i` and press [Enter] to enter insert mode and make any necessary changes. When finished, press the [Esc] key, type `:wq`, and press [Enter].

After the editor closes, enter your local password and the database will be rebuilt and signed.

After a new Tripwire database is written, the newly authorized integrity violations will no longer show up as warnings.

19.8. Updating the Tripwire Policy File

If you want to change the files Tripwire records in its database, change email configuration, or modify the severity at which certain violations are reported, you need to edit your Tripwire policy file.

First, make whatever changes are necessary to the sample policy file `/etc/tripwire/twpol.txt`. If you deleted this file (as you should whenever you are finished configuring Tripwire), you can regenerate it by issuing the following command:

```
twadmin --print-polfile > /etc/tripwire/twpol.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` with the `#` character as in the following example:

```
#     /etc/smb.conf                     -> $(SEC_CONFIG) ;
```

Next, you must generate a new, signed `/etc/tripwire/tw.pol` file and generate an updated database file based on this policy information. Assuming `/etc/tripwire/twpol.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 password. Then, the `twpol.txt` file will be encrypted 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 `bob.domain.com.twd`, type this command:

```
rm /var/lib/tripwire/bob.domain.com.twd
```

Then type the following command to create a new database using the updated policy file:

```
/usr/sbin/tripwire --init
```

To make sure the database was correctly changed, run the first integrity check manually and view the contents of the resulting report. See Section 19.5 *Running an Integrity Check* and Section 19.6.1 *Viewing Tripwire Reports* for more on doing these tasks.

19.8.1. Tripwire and Email

You can configure Tripwire to send an email to one or more accounts if a specific type of policy is violated. In order to do this, you need to figure out what policy rules should be monitored and who should get the email when those rules are broken. Note that on large systems with multiple administrators, you can have different sets of people notified depending on the types of violations.

Once you have determined who to notify and what rule violations to report to them, edit the `/etc/tripwire/twpol.txt` file, adding an **`emailto=`** line to the rule directive section for each appropriate rule. Do this by adding a comma after the **`severity=`** line and putting **`emailto=`** on the next line, followed by one or more email addresses. More than one email address can be specified if the addresses are separated by a semi-colon.

For example, if two administrators, Johnray and Bob, need to be notified when 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 = johnray@domain.com;bob@domain.com
)
```

After changing the policy file, follow the instructions in Section 19.8 *Updating the Tripwire Policy File* to generate an updated, encrypted, and signed copy of the Tripwire policy file.

19.8.1.1. Sending Test Email Messages

To test Tripwire's email notification configuration, 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.

19.9. Updating the Tripwire Configuration File

If you want to change Tripwire's configuration file, you should first edit the sample configuration file `/etc/tripwire/twcfg.txt`. If you deleted this file (as you should whenever you are finished configuring Tripwire), you can regenerate it by issuing the following command:

```
twadmin --print-cfgfile > /etc/tripwire/twcfg.txt
```

Tripwire will not recognize any configuration changes until the configuration text file is correctly signed and converted to `/etc/tripwire/tw.pol` with the `twadmin` command.

Use the following command to regenerate a configuration file from the `/etc/tripwire/twcfg.txt` text 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 Tripwire database.

19.10. Tripwire File Location Reference

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.

- Within the `/usr/sbin/` directory, you will find the following programs:
  - `tripwire`
  - `twadmin`
  - `twprint`
- Within the `/etc/tripwire/` directory, you will find the following files:
  - `twinstall.sh` — The initialization script for Tripwire.
  - `twcfg.txt` — The sample configuration file supplied by the Tripwire RPM.
  - `tw.cfg` — The signed configuration file created by the `twinstall.sh` script.
  - `twpol.txt` — The sample policy file supplied by the Tripwire RPM.
  - `tw.pol` — The signed policy file created by the `twinstall.sh` script.
  - Key Files — The local and site keys created by the `twinstall.sh` script which end with a `.key` file extension.
- After running the `twinstall.sh` installation script, you will find the following files in the `/var/lib/tripwire/` directory:
  - The Tripwire Database — The database of your system's files and has a `.twd` file extension.
  - Tripwire Reports — The `report/` directory is where Tripwire reports are stored.

The next section explains more about the roles these files play in the Tripwire system.

19.10.1. Tripwire Components

The following describes in more detail the roles the listed in the previous section play in the Tripwire system.

/etc/tripwire/tw.cfg

This is the encrypted Tripwire configuration file which stores system-specific information, such as the location of Tripwire data files. The `twinstall.sh` installer script and `twadmin` command generate this file using the information in the text version of the configuration file, `/etc/tripwire/twcfg.txt`.

After running the the installation script, the system administrator can change parameters by editing `/etc/tripwire/twcfg.txt` and regenerating a signed copy of the `tw.cfg` file using the `twadmin` command. See Section 19.9 *Updating the Tripwire Configuration File* for more information on how to do this.

/etc/tripwire/tw.pol

The active Tripwire policy file is an encrypted 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 sample text policy file, `/etc/tripwire/twpol.txt`, is used to generate the active Tripwire policy file.

After running the the installation script, the system administrator can update the Tripwire policy file by editing `/etc/tripwire/twpol.txt` and regenerating a signed copy of the `tw.pol` file using the `twadmin` command. See Section 19.8 *Updating the Tripwire Policy File* for more information on how to do this.

/var/lib/tripwire/host_name.twd

When first initialized, Tripwire uses the signed policy file rules to create this database file. The Tripwire database 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*.

/var/lib/tripwire/report/host_name-date_of_report-time_of_report.twr

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. Tripwire reports are named using the following convention: *host_name-date_of_report-time_of_report*`.twr`. These reports detail the differences between the Tripwire database and your actual system files.

19.11. Additional Resources

Tripwire can do much more than what is covered in this chapter. Refer to these additional sources for more information about Tripwire.

19.11.1. Installed Documentation

- `/usr/share/doc/tripwire-`<*version-number*> — An excellent starting point for learning 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.

19.11.2. Useful Websites

- http://www.tripwire.org — The home of the Tripwire Open Source Project. Here you can find the latest news on the application, including a helpful FAQ.
- http://sourceforge.net/project/showfiles.php?group_id=3130 — This links to the latest official documentation from the Tripwire project.

IV. Appendixes

Table of Contents

Appendix A.

General Parameters and Modules

This appendix is provided to illustrate *some* of the possible parameters available for common hardware device *drivers*[1], which under Red Hat Linux are called kernel *modules*. In most cases, the default parameters will work. However, there may be times when extra module parameters are necessary for a device to function properly or if it is necessary to override module's default parameters for the device.

During installation, Red Hat Linux uses a limited subset of device drivers to create a stable installation environment. Although the installation program supports installation on many different types of hardware, some drivers (including those for SCSI adapters, network adapters, and many CD-ROM drives) are not included in the installation kernel. Rather, they must be loaded as modules by the user at boot time. For information on where one can find extra kernel modules during the installation process, refer to the section concerning alternative boot methods in the chapter titled *Steps to Get You Started* in the *Red Hat Linux Installation Guide*.

Once installation is completed, support exists for a large number of devices through kernel modules.

A.1. Specifying Module Parameters

In some situations, it may be necessary to supply parameters to a module as it is loaded in order for it so function properly. This can be done in one of two ways:

- 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. 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 specific parameters.

Caution

When a parameter has commas, be sure *not* to put a space after a comma.

1. A driver is software which enables Linux to use a particular hardware device. Without a driver, the kernel can not communicate with attached devices.

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.redhat.com to make sure the device is supported.

Even though parameters are specified after loading the driver disk and specifying the device, one of the more commonly used parameters `hdX=cdrom` (where *X* corresponds to the appropriate drive letter) *can* be entered at a boot prompt during installation. This is allowed since it effects IDE/ATAPI CD-ROMs, which are 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 will require manual changes to settings in the module source code and a recompile.

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*
Sanyo CDR-H94A	`sjcd.o`	sjcd=*io_port* *OR* sjcd_base=*io_port*

Hardware	Module	Parameters
Sony CDU-535 & 531 (some Procomm drives)	`sonycd535.o`	sonycd535=*io_port*

Table A-1. Hardware Parameters

Here are some examples of these modules in use:

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

Table A-2. Hardware Parameters Configuration Examples

Note

Most newer Sound Blaster cards come with IDE interfaces. For these cards, do not use `sbpcd` parameters; only use `hdX` parameters (where *X* corresponds to the appropriate drive letter).

A.3. SCSI parameters

Hardware	Module	Parameters
Adaptec 28xx, R9xx, 39xx	`aic7xxx.o`	
3ware Storage Controller	`3w-xxxx.o`	

Hardware	Module	Parameters
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-A100U2W	`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`	
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`	
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`	

Hardware	Module	Parameters
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`	
UltraStor 14F/34F (not 24F)	`u14-34f.o`	
UltraStor 14F, 24F, and 34F	`ultrastor.o`	
WD7000 Series	`wd7000.o`	

Table A-3. SCSI Parameters

Here are some examples of these modules in use:

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

Table A-4. SCSI Parameters Configuration Examples

A.4. Ethernet Parameters

Important

Most modern Ethernet-based network interface cards (NICs), do not require module parameters to alter settings. Instead, they can be configured using `ethtool` or `mii-tool`. Only after these tools fail to work should module parameters be adjusted.

For information about using these tools, consult the man pages for `ethtool` and `mii-tool`.

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`	
3Com EtherLink PCI III/XL Vortex (3c590, 3c592, 3c595, 3c597) Boomerang (3c900, 3c905, 3c595)	`3c59x.o`	full_duplex= *0* is off *1* is on
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*
Broadcom BCM5700 10/100/1000 ethernet adapter	`bcm5700.o`	
Crystal SemiconductorCS89[02]0	`cs89x0.o`	

Hardware	Module	Parameters
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*'
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 Ether Express/100 driver	`e100.o`	e100_speed_duplex=*X* *If X* = 0 = autodetect speed and duplex 1 = 10Mbps, half duplex 2 = 10Mbps, full duplex 3 = 100Mbps, half duplex 4 = 100Mbps, full duplex
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`	

Hardware	**Module**	**Parameters**
Intel EtherExpress 16 (i82586)	`eexpress.o`	eexpress=*io_port*,*IRQ* ***OR*** eexpress io=*io_port* irq=*IRQ* options= 0x10 10base T half duplex 0x20 10base T full duplex 0x100 100base T half duplex 0x200 100baseT full duplex
SMC EtherPower II 9432 PCI (83c170/175 EPIC series)	`epic100.o`	
Racal-Interlan ES3210 EISA	`es3210.o`	
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, Shared-Memory 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`	
IBM Olympic-based PCI token ring	`olympic.o`	
AMD PCnet32 and AMD PCnetPCI	`pcnet32.o`	
SIS 900/701G PCI Fast Ethernet	`sis900.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*
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*

Hardware	Module	Parameters
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`	

Table A-5. Ethernet Module Parameters

Here are some examples of these modules in use:

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

Table A-6. Ethernet Parameter Configuration Examples

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), add `alias` (and possibly `options`) lines for each card to `/etc/modules.conf`. Refer to the chapter titled *Kernel Modules* in the *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), 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

Symbols

A

B

C

D

E

F

G

K

L

M

N

O

P

R

S

T

U

V

W

X

Colophon

The Red Hat Linux manuals are written in DocBook SGML v4.1 format. The HTML and PDF formats are produced using custom DSSSL stylesheets and custom jade wrapper scripts. The DocBook SGML files are written in **Emacs** with the help of PSGML mode.

Garrett LeSage created the admonition graphics (note, tip, important, caution, and warning). They may be freely redistributed with the Red Hat documentation.

The Red Hat Linux Product Documentation Team consists of the following people:

Sandra A. Moore — Primary Writer/Maintainer of the *Red Hat Linux x86 Installation Guide*; Contributing Writer to the *Red Hat Linux Getting Started Guide*

Tammy Fox — Primary Writer/Maintainer of the *Red Hat Linux Customization Guide*; Contributing Writer to the *Red Hat Linux Getting Started Guide*; Writer/Maintainer of custom DocBook stylesheets and scripts

Edward C. Bailey — Primary Writer/Maintainer of the *Red Hat Linux System Administration Primer*; Contributing Writer to the *Red Hat Linux x86 Installation Guide*

Johnray Fuller — Primary Writer/Maintainer of the *Red Hat Linux Reference Guide*; Co-writer/Co-maintainer of the *Red Hat Linux Security Guide*; Contributing Writer to the *Red Hat Linux System Administration Primer*

John Ha — Primary Writer/Maintainer to the *Red Hat Linux Getting Started Guide*; Co-writer/Co-maintainer of the *Red Hat Linux Security Guide*; Contributing Writer to the *Red Hat Linux System Administration Primer*