**Unit I**

**What Is LAMP?**

LAMP is a proven, efficient set of software that works well as a system. The open architecture of each of these elements allows for smooth and seamless integration with one another and results in a powerful combination. Early adopters of these technologies back in 1997 were seen as radical, but today the open source movement is on the rise, and both large and small enterprises are adopting the LAMP method of development. Steering away from the high costs of implementing licensed server and client software is becoming increasingly beneficial because the stability of each application is surpassing that of its far more expensive brethren. In recent events, some governments have decided to make the jump to open source software, touting its reliability, efficiency, and substantial cost savings over proprietary solutions.

Besides reaching this dependability level capable of being embraced by entire governments, the major advantage seen by LAMP adopters is *speed.*

Each component of LAMP exhibits benchmarks that far exceed those of their competitors, and as with any equation the sum is much more than its parts. Let’s take a look at just the Linux/Apache combination for a moment. This outstanding combination is capable of serving more pages to its users than any other commercial or open source solution. “What about MySQL?” you might ask.

MySQL is the fastest open source database available, with speed comparable to that of Oracle’s.

This fact alone pushed NASA (the National Aeronautics and Space Administration) to switch to MySQL in 2000. With MySQL, the level of functionality offered to its consumers grows in leaps and bounds—most notably by offering *stored procedures,* a system found only in mature databases.

Which brings us to the power of PHP. PHP is the fastest server-side scripting program on the planet. Faster than a one-legged chicken on a downhill slope, PHP whisks by Active Server

Pages (ASP), Java, .Net, and ColdFusion, allowing a greater maximum user count per server while providing the same amount of functionality—taking into account proper programming methods of course. Now that we’ve touched on a few of the major advantages of choosing LAMP, let’s take a brief look at each of its elements.

***L*—Linux Operating System**

*Linux* is the operating system that runs the applications. It is specifically noted for its speed,minimal hardware requirements, security, and remote administration. Another great key pointabout Linux is that it’s free! Linux is a fully featured operating system that doesn’t cost you anythingto use. You can download it directly from the Internet, install it, and use it without everspending a dime (except for your own hardware of course). If you’re curious about how this ispossible, see the “Why LAMP Is Free” section later in this chapter.

Another major advantage of Linux is its ability to run with or without a graphical user interface

(GUI), depending on your needs. (The non-GUI interface could almost be related to DOS mode for those Windows users who are not yet familiar with Linux.)

Linux is a project that began as a hobby by mastermind Linus Torvalds while he was a student at the University of Helsinki in Finland. Torvalds, originally working with Minix (a small Unix system), decided to create an operating system that would exceed the Minix standards. He began his development in 1991, and his first public release was version 0.02. Development of Linux continues even now with updates released as enough major changes are made to justify a new *version* release. Now that Torvalds has gained a much larger development team, releases are becoming more and more frequent. He is also the one who chose Tux (the penguin) to be the mascot of Linux.

Because Linux is published under the GNU (GNU stands for

*GNU’s Not Unix*) General PublicLicense (GPL), many companies and individuals have taken the source code and adapted it totheir needs. Some (most notably Red Hat) have turned huge profits by offering support for theirreleases and are now closing their source to commercialize their distributions.

***A*—Apache Web Server**

Developed by the Apache Software Foundation (ASF),

*Apache* is an open source web server solutionthat is packed with features, is extremely fast, and works well with the Linux operating system.

With the Apache web server, you can create *virtual hosts* that enable you to run multiple websites on a single server, and it has many more awesome features. The Apache web server is available for the Windows environment as well; however, your system will suffer from decreased performance because of Microsoft’s memory management (a.k.a.*leaks*) and architectural differences.

Therefore it is *highly* recommended that you use Linux for all your web serving needs.

A quick overview of Apache’s features would include items such as enhanced logging, bandwidth throttling, directory access protection, Common Gateway Interface (CGI) support,

Secure Sockets Layer (SSL) support, and a handful of other built-in modules that enable you to do all sorts of neat things with your website.

Apache is rapidly growing in popularity and is currently the number one web server solution according to Netcraft (www.netcraft.com) surveys. It has held the number one position, with a large margin, for eight years. Taking the number one spot in 1996, Apache has grown to dominate the market with more than a 69 percent saturation. Coming in a rough second place is Microsoft, with 20 percent, which has stayed about the same since 1997. Next is Sun ONE with 3 percent.

***M*—MySQL Database Server**

*MySQL* is a powerful, robust database manager that enables you to store and retrieve data witha scripting language such as PHP. You can store various types of data, such as Boolean operators,text, integers, images, binary digits, and BLOBs (binary large objects) quickly and efficientlywith minimal effort. Using a database is important for creating dynamic sites. The term“dynamic site” is derived from being able to utilize a single page of code to display differentinformation based on a user’s interaction. This would be virtually impossible without the useof a database and a scripting language such as PHP to manipulate the data.

MySQL is packed full of features such as data replication, table locking, query limiting, user accounts, multiple databases, persistent connections, and—as of MySQL 5—stored procedures, triggers, and views. These features will be explained in more detail later, but for now you should be aware of some of the benefits you will enjoy from implementing such a great database manager.

MySQL, developed by MySQL AB, originated from a need for the founders to use mSQL to connect to their own fast, low-level (Indexed Sequential Access Method, or ISAM) routines.

After testing these procedures and functions, they were found to be neither fast nor flexible enough and so MySQL was born: a new system from almost the same Application Protocol

Interface (API) as mSQL, so that any third-party code that might be written for mSQL could easily be ported to MySQL. The mascot for MySQL is the dolphin seen in their logo. Her name is Sakila, and she was given her name from a Name the Dolphin contest held by MySQL.

***P*—PHP Scripting Language and Engine**

*PHP* is a recursive acronym that stands for *PHP: Hypertext Preprocessor*. This widely used general purposescripting language is especially suited for Web development and can be embedded into

HTML. What this means to you is that it’s a simple scripting language that can greatly enhance your website. You simply learn the code, apply the logic, and create a dynamic website that can interact with your users on many levels greater than the traditional “flat file” HTML methods of the Internet.

PHP’s initial inception in 1995 was a simple set of Perl scripts for tracking Rasmus Lerdorf’s online résumé. As time went on, Lerdorf began to write a much larger C implementation to handle the increased amount of functionality he needed, including database connectivity. Lerdorf then decided to send out an initial release, open source style, called

PHP/FI for anyone to use and to improve upon. Back in the day, this stood for Personal

Home Page/Forms Interpreter. By 1997, the second release was distributed (PHP/FI 2.0) and had started to gain a following of several thousand from around the globe. Although several individuals were contributing code, it was still Lerdorf who continued the majority of all development.

Mid-1997 saw the dawn of a new age of PHP: PHP 3. This version was a complete rewrite of PHP/FI 2.0 by Andi Gutmans and Zeev Suraski, who needed more juice than was previously available for a university project they had been working on. To capitalize on PHP’s growing user base, Lerdorf, Gutmans, and Suraski decided to release this new creation under the PHP name and so started what we know today as PHP. In the winter of 1998, PHP 4 development was begun by Gutmans and Suraski. They released the first official version in May of 2000. PHP 4 boasted much higher performance and pushed new technologies to its ever growing fan base with HTTP sessions, output buffering, and more secure ways of handling user input.

We believe that PHP 5 will knock the butterfly off its flower when it hits. A new object oriented model coupled with the Zend Engine 2, stack tracing, and exception handling is expected to push a wider acceptance across the planet. At the same time, an introduction of integration with external object-oriented models, such as COM and Java, will throw a wildcard into the mix. For the first time, the ability of other communities to integrate seamlessly with PHP will be available. This means that prewritten APIs will be able to be much more easily integrated with PHP, destroying the last remaining reasons for these other communities to

*Not* use PHP.

**Why LAMP?**

If you are wondering, “Why choose LAMP?” then the answer is something you must consider for yourself. The combination has been proven on many popular websites, and the technology is free to use. These factors are why many people and businesses have chosen this combination for launching their production-based websites. With LAMP, you have full control over your server. Most important, you have remote access, which enables you to easily administer your Linux server from anywhere in the world.

Linux enables you to run the services required without running a GUI, and therefore it uses less system resources—resources that could be used to speed up the process of delivering web pages to your audience.

**Who Else Is Using LAMP?**

The question of “Who is using LAMP?” is often asked. Because of the size of the Internet and the number of servers out there, we do not have the ability to give you a full report; however here’s a list of some of the more popular sites using LAMP:

**LinuxForum.com**

A free Linux support website.

**ApacheFreaks.com**

A free Apache web server support website.

**MySQLFreaks.com**

A free MySQL database manager support website.

**PHPFreaks.com**

A free PHP support website.

**Winamp.com**

The Nullsoft Winamp site is built around PHP.

**DevShed.com**

The open source development site Developer Shed is completely PHPdriven.

**MP3.com**

The single largest MP3 Web-based server is built using PHP.

**SpeedTV.com**

SPEED TV, the television station owned by Fox, has been completely developed in PHP with a few Flash/ColdFusion elements thrown in.

**Yahoo.com**

Even though Yahoo does not show the .php extension on their files, it has been confirmed by many sources that they are using it. Here’s our source:

public.yahoo.com/~radwin/talks/yahoo-phpcon2002.htm

.

Nearly 70 percent of the websites you visit are using Linux as their operating system, and

MySQL AB and Zend Technologies report that “over 10 million Web-based applications have been built using MySQL and PHP through low-cost open source software stacks such as LAMP”

(seewww.mysql.com/press/release\_2004\_05.html). This number continues to grow as Linux and other open source projects gain momentum within the industry. Every passing year adds more credibility, more features, and more support to the LAMP solution.

**Choosing Which Linux to Use**

Over the years many distributions, or *flavors,* of Linux have been developed and made available around the world. The following list includes some of the major players in the world of the penguin; although it is not a complete list of all the flavors, it’s enough to familiarize yourself with a little of what’s out there. Each of these distributions is a good choice, but a few stand out from the crowd in terms of reliability, availability, functionality, and ease of use. In alphabetical order we have these distributions:

**Debian**

Debian was officially released by Ian Murdock on August 16, 1993. Debian is the self-proclaimed *only Linux distribution to allow any developer to develop for*. Although Debian is no newcomer to the business, it has never reached the iconic status of many of the other distributions.

You can learn more about Debian at [www.debian.org](http://www.debian.org).

**Fedora**

Fedora Linux was originally started by Warren Togami as a means of developing high-quality Red Hat Package Management system (RPM) packages for Red Hat. A 2003 announcement of Red Hat turning commercial also declared the merging of Fedora Linux into what is now the Fedora Project. You can learn more about Fedora at fedora.redhat.com.

**Obtaining Fedora Linux**

Fedora Linux can be acquired in several ways, the two most popular being to download or to purchase. This section discusses both.

If you have access to a computer that has a CD-R or CD-RW drive, you can download the complete set of installation discs, for free, directly from the Internet in International Organization for

Standardization (ISO) format from a site such as LinuxForum (http://www.linuxforum.com).

After you’ve located the Fedora download section, it’s time to choose a version. It’s best to download the newest full-release version because a full release has been deemed *stable* and is out of its testing phases. Fedora Core 2 is the newest release, so this book will be covering that version in the most detail. If you are unable to find the distribution at LinuxForum, you can check the Fedora mirrors list at

<http://fedora.redhat.com/download/mirrors.html>.

**Installing Fedora Linux**

The time has finally arrived. After familiarizing yourself with the many flavors of Linux and learning why we have chosen Fedora, you built or purchased a new or used PC, and now you are ready to install Linux.

**Pre-Installation**

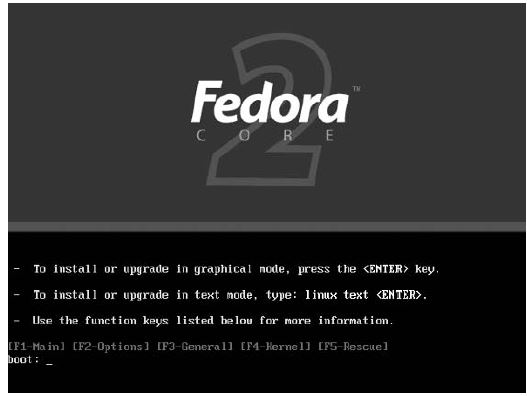
Before you are ready to begin that actual installation, you must first perform the following steps:

**1.** Insert disc 1 into the CD-ROM drive. Of course you’ll need to power on in order to do so.

**2.** You might need to change your boot order in the BIOS settings for your motherboard toboot from the CD first. To do this, pay attention to the first screen your computer showsduring the boot sequence. You should notice a line that saysPress <del> to Enter Setupor maybePress <F2> to Enter BIOSor something along those lines. Follow the instructionsand you will enter the BIOS configuration settings for your motherboard. The bootorder is most often located under the General Settings category. Enter the General Settingscategory and change the first boot device to be your CD-ROM drive. After you’vemade the appropriate changes, choose Exit and Save Changes and wait for your computerto boot from the CD.

**3.** You should be presented with a screen that looks identical to Figure 2.1. This is theFedora Core 2 Pre-installation screen. From here, hot keys will enable you to viewdocumentation and additional specifications you can use for more advanced installationsor recoveries should you be using nonstandard hardware or suffer from a corrupted kernel.

The first two options that are displayed on the screen are the ones to be concerned with now.



**4.** You will be using the second option, which provides the functionality to install or upgradeFedora Linux by using text-only mode. This option is usually reserved for individuals who are using an older cathode ray tube, or CRT, monitor (a *green screen*), people who are experiencing difficulty displaying the VGA graphics used during the installation procedure, or those who feel the need to express their inner geek. In order to support the greatest number of users, enter the text mode by typing **linux text** and pressing Enter to continue. You’ll then need to wait a brief moment while the installation detects some basic hardware and prepares the installation procedure for you.

**Installation Type Selection**

You are now ready to choose your installation type, follow the steps below being sure everything is correct before proceeding on the next step:

1. The first installation screen you should see is a small text-based graphic window asking whether you would like to check the installation CDs. If you have downloaded and burned these CDs, you should run the integrity check for CD1 and CD2. It’s better to wait a minimal amount of time now than to become frustrated with your installation if things head south due to unseen problems. If you purchased the CDs, then most likely they have alreadybeen checked by the distributor. In this case you can select the Skip option and continue. Ifyou run into problems, you can always reboot the machine and begin the installation procedureagain so you can verify the CDs.

**2.** The next screen prompts you to select a language to use during the installation process. Simpleenough, it should default to English, so selecting OK should continue you on your way.

You might want to note that to switch between elements of these screens (that is, lists, check boxes, buttons, and so on) you will be using the Tab key. The spacebar will enable you to select and deselect certain options, and F12 will enable you to go directly to the next screen.

**3.** A few text lines appear at the bottom of the screen as the Fedora Installer tries to detectyour hardware. Depending on whether your hardware is successfully identified, you willthen see one or more of the following screens: Keyboard, Mouse, and Video. The Keyboardand Mouse screens are straightforward and should already have hardware settingsthat most closely match your hardware. If these are in error, select the proper choice andcontinue. If you are brought to the video card selection screen, you will need to know thetype of video card you have and the resolution your monitor is capable of supporting. Make the appropriate selections and continue to the next step.

**4.** You are then brought to the Installation Type screen, shown in Figure 2.2. This enablesyou to set the overall basic packages that will be installed. The options listed here come withcertain features and modules pre-installed to fit your specific application of Linux. Here’sa breakdown of your options:

**Personal Desktop**

This option should be selected if you wish to run your Linux box as a personal computer for desktop publishing, web browsing, e-mail, graphics, games, and so on.

**Workstation**

This option provides a basic system setup with a developer’s needs in mind. It focuses more heavily on libraries that might be beneficial to programmers.

**Server**

Selecting the Server option installs software necessary for your Linux server to handle the routing of data to other computers on your network or the Internet.

**Custom**

For the more experienced Linux user, this option affords the opportunity to select each library and package specific to your case. This is not recommended to the first-time user because it is easy to overlook something you might need.

**5.** Because lessons in this book are based on setting up a LAMP environment, select Serverfrom the list by tabbing to the radio box and using the up or down arrows to highlight Server.

Press Tab once more to jump down to OK and press Enter to proceed to the next screen. south due to unseen problems. If you purchased the CDs, then most likely they have already been checked by the distributor. In this case you can select the Skip option and continue. If you run into problems, you can always reboot the machine and begin the installation procedure again so you can verify the CDs.

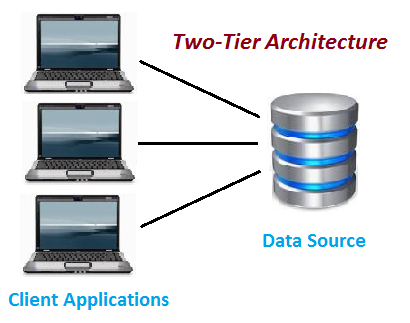


 In today’s software testing class we are discussing about the *Two-Tier and Three-Tier architecture*.

All projects are broadly divided into two types of applications **2 tier and 3 tier architecture**. Basically high level we can say that *2-tier architecture* is Client server application and *3-tier architecture* is Web based application. Below I am concentrating on the difference between Two-Tier and Three-Tier Architecture, what all advantages, disadvantages and practical examples.

**Two-Tier Architecture:**

The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server. Because of tight coupling a 2 tiered application will run faster.

[](https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/two-tier-architecture/)

Two-Tier Architecture

The above figure shows the architecture of two-tier. Here the direct communication between client and server, there is no intermediate between client and server.

Let’s take a look of real life example of Railway Reservation two-tier architecture:

Let’s consider that first Person is making Railway Reservation for Mumbai to Delhi by Mumbai Express at Counter No. 1 and at same time second Person is also try to make Railway reservation of Mumbai to Delhi from Counter No. 2

If staff from Counter No. 1 is searching for availability into system & at the same staff from Counter No. 2 is also looking for availability of ticket for same day then in this case there is might be good change of confusion and chaos occurs. There might be chance of lock the Railway reservation that reserves the first.

But reservations can be making anywhere from the India, then how it is handled?

So here if there is difference of micro seconds for making reservation by staff from Counter No. 1 & 2 then second request is added into queue. So in this case the Staff is entering data to Client Application and reservation request is sent to the database. The database sends back the information/data to the client.

In this application the Staff user is an end user who is using Railway reservation application software. He gives inputs to the application software and it sends requests to Server. So here both Database and Server are incorporated with each other, so this technology is called as “***Client-Server Technology***“.

The Two-tier architecture is divided into two parts:

**1) Client Application (Client Tier)  
2) Database (Data Tier)**

On client application side the code is written for saving the data in the SQL server database. Client sends the request to server and it process the request & send back with data. The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.

**Advantages:**

1. Easy to maintain and modification is bit easy
2. Communication is faster

**Disadvantages**:

1. In two tier architecture application performance will be degrade upon increasing the users.
2. Cost-ineffective

**Three-Tier Architecture:**

**Three-tier architecture** typically comprise a presentation tier, a business or data access tier, and a data tier. Three layers in the three tier architecture are as follows:

**1) Client layer**  
**2) Business layer**  
**3) Data layer**

**1) Client layer:**

It is also called as *Presentation layer* which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user. For example designing registration form which contains text box, label, button etc.

**2) Business layer:**

In this layer all business logic written like validation of data, calculations, data insertion etc. This acts as a interface between Client layer and Data Access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data layer.

**3) Data layer:**

In this layer actual database is comes in the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.

[](https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/three-tier-architecture/)

Three-tier Architecture

**Advantages**

1. High performance, lightweight persistent objects
2. Scalability – Each tier can scale horizontally
3. Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
4. High degree of flexibility in deployment platform and configuration
5. Better Re-use
6. Improve Data Integrity
7. Improved Security – Client is not direct access to database.
8. Easy to maintain and modification is bit easy, won’t affect other modules
9. In three tier architecture application performance is good.

**Disadvantages**

1. Increase Complexity/Effort

**Hard Disk Partitioning**

The Linux operating system requires that you partition your hard disks before you can install the operating system. A *partition* is a way for the operating system to reserve a block of your hard disk space for a specific purpose. Linux will make sure these partitions do not overlap, resulting in corrupted data.

Follow these steps to partition your hard disk:

**1.** You should be at the Disk Partitioning Setup screen, which presents you with these options:

**Autopartition** is the default, highlighted option.

**Disk Druid** enables you to set up your partitions in detail by inputting the device, size, type, and mount point for each partition.

**2.** If you do not already have any special partitions on your drive, choose Autopartition and skip to step 3. If you do have special conditions that you need to work around, choose Disk

Druid, proceed to step 2, and then skip to the rest of this section.

**3.** If you choose to set up your own partitions because of multiple hard drives or another reason, there are a few critical things to keep in mind. First, you must have a swap partition.

A *swap partition* is used for virtual memory support. If your computer runs out of RAM for storing temporary data, this partition will be used in its place to handle any overflow. When creating your swap partition, be sure to make its size equal to twice the amount of RAM you have installed, but no greater than 2048MB (or 2GB). Next you need to create a *boot* partition.

The boot partition is where your operating system kernel resides along with any files used during the bootstrap process. A 50MB partition should be more than sufficient; if you make it too large, it could cause problems with your BIOS.

**4.** The last partition that is required is the *root* partition. This partition houses all the files not residing on /boot so it will make up the vast majority of your drive. A minimum of about

2.5GB is needed to install the packages required for the Server selection you made earlier.

**5.** If you are manually configuring your partitions, you should be aware of the partition types available:

**ext2** This filesystem is the standard Unix/Linux partition type and supports long filenames

up to 255 characters.

**ext3** This filesystem is almost identical to ext2 but it allows journaling. Journaling requires less time to be spent during a system recovery because there is no need to run the fsck command, which attempts to repair the system.

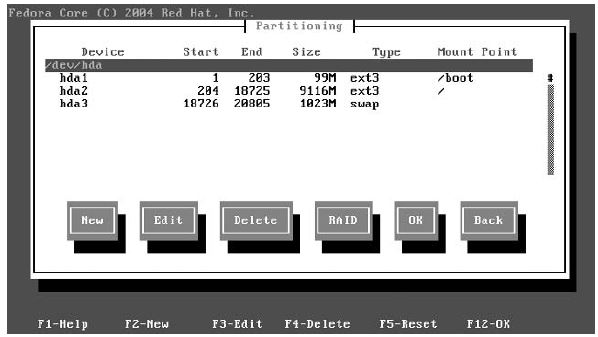
**software RAID** If you create two or more software RAID partitions, you can enable a RAID device. A RAID can provide faster read and/or write times as well as data redundancy should a drive fail. If you are building a system that might contain important or sensitive information, you might wish to research this further.

**swap** As discussed before, this type of partition provides Linux with virtual memory support.

**vfat** A vfat partition is used to create a partition compatible with a Windows 95/NT filesystem. It uses the standard FAT table format and also supports long filenames.

**6.** After selecting the Autopartition option and continuing to the next screen, a warning appears indicating that all the information contained on the drive will be lost. Go ahead and agree to this screen if that is okay. An Automatic Partitioning window appears in the center of the screen and prompts you for two pieces of information. The first asks how you would like to handle the existing information on your drive. Because you are installing a clean system, select Remove All Partitions on This System. Tab to the next prompt, which asks which drives you would like to use for this installation. If you have a special case and would like to keep any existing data on your computer, select the appropriate choice from the list and also tab down to the next section. From here, select all drives available on your system, and press F12 or select OK. Another warning screen is displayed, confirming that you do indeed wish to remove all the partitions on each of the drives selected. Make sure the Yes option is highlighted and press Enter to continue.

**7.** The Partitioning screen, shown in Figure 2.3, displays how the autopartitioner has configured your disk space. It also affords you the opportunity to make any last-minute changes to your filesystem schema before continuing with the installation. If you feel something isn’t right, refer back to step 2 for more information on specific partition options and setup, or you can press F1 for help. If everything appears okay, continue in the usual manner to bring up the next screen. At this point, if your system has only the minimum amount of RAM needed for the installation, the installer will ask whether it is okay to enable the swap partition you just enabled. This is fine, so go ahead and agree and you’ll be taken to the next screen.



**Selecting a Boot Loader**

The next task is to select whether you would like to use the GRand Unified Bootloader (GRUB).

GRUB will enable you to boot from multiple operating systems and also provide additional security from anyone who might be able to access your system locally. Follow these steps:

**1.** You will be using GRUB, so make sure Use GRUB Boot Loader is selected and continue.

**2.** The Boot Loader Configuration screen prompts you for any special options or arguments that might need to be passed to the kernel during the boot sequence. Most likely you should not need anything here, so simply pass through this page by continuing. A second page for the boot loader configuration is then displayed asking whether you would like to specify a password to access GRUB before the operating system boots. If you need tightened security, then use this option.

**3.** The next Boot Loader Configuration screen enables you to label other operating systems to boot from and to select the default option as well. You will see in Figure 2.4 that we have only one installation, and it is set to the default automatically.

**4.** The last Boot Loader Configuration screen asks where you would like to install the boot loader. Select the Master Boot Record (MBR) because this makes the most sense. Select

OK to move on.

**Exploring the Linux Shell**

During the “Login Process” section, we talked about a successful login spawning a new user shell. This shell is your command-line interface that enables you to run commands locally on your system and perform any number of administration tasks. In our case, the shell used is the Bash shell*.*

To access the shell, however, you must pass through other programs. Most prominently, mingetty is displayed asking you for a login. The mingetty program was designed to be a minimal getty for the virtual terminals you use when accessing the system from the local keyboard and monitor; mingetty does not have all the functionality that getty does, but it requires only a fraction of the resources. If you will eventually be using the server as remote only and will not be using any local access to the system, you might wish to remove all but one of the mingetty calls in the inittab file to save approximately 8KB of memory per instance.

You can still log in remotely because a remote login is usually available through the Secure Shell (SSH) connection protocols (controlled by the inetd daemon), but it is a good idea to leave one 8KB instance open in case you need to access the machine locally.

After a successful login, you are presented with a Bash shell. The *Bash shell* is a command language interpreter originally created for the GNU operating system. Bash stands for Bourne-Again SHell, which is a pun based on Stephen Bourne’s name. Bourne originally created the current Unix shell /bin/sh, which first appeared in the Seventh Edition Bell Labs Research version of Unix. The Bash shell also incorporates a large amount of the /bin/csh, or C Shell, functionality to allow for a more diverse set of instructions and commands available to the user.

Although a shell is a command-line interpreter, it also acts as a programming language on its own. From this concept stems much of the power of the Linux operating system. Any shell command can be used synchronously or asynchronously. This, in effect, means that processes can be executed in order, each waiting for the previous to finish, or they can be executed in parallel (multitasking). Furthermore, there are *redirection* constructs in place that allow microcontrol over the input and output of any of the commands. The shell also allows control over the commands’ environments in the form of flags and other arguments.

Let’s take a closer look at some fundamental principles that you should know before delving into more advanced concepts:

**Tab** You can use the Tab key to finish a file or directory name for you. If you begin typing

**super** for *supercalifragilisticexpialidocious,* just use Tab to auto-complete the rest of the file. If more than one word is a match for what you have typed so far, than nothing will appear to happen or auto-complete will fill in up to the fork in the matching. Pressing Tab again will give you a listing of all possible matches. After you enter a few more letters that result in only a single match for the rest of the string, pressing the Tab key again will complete the string.

**Escape character** Bash uses the backslash (\) character as an escape character. An escape character preserves the literal value of the next character that follows the backslash. If you wish to have a backslash contained within an argument, then you will need to use two backslashes (\\) in order to escape the backslash itself.

**Single quotes** Single quotes are used as enclosing characters to preserve the literal value for an entire string. Use single quotes when changing to a directory that contains a space or other special character that might interfere with the command line. For instance, if you needed to move into a directory called /Special Files, you would use this command: cd /'Special Files'

Without enclosing the directory name in quotes, you would confuse the cd command because it does not take more than one argument. You cannot use a single quote within your two enclosing single quotes.

**Double quotes** Enclosing a string in double quotes will also preserve the literal value, except for the characters $, `, and \. Backslashes can be used within double quotes to escape characters as discussed previously.

**ANSI-C quoting** Words that are used in the form of $'*string*' are also treated specially.

Any backslashed characters contained within the string will be replaced according to the ANSI C standard. They are as follows:

\a alert(bell)

\b backspace

\e an escape character (not ANSI C)

\f form feed

\n newline

\r carriage return

\t horizontal tab

\v vertical tab

\\ backslash

\' single quote

\nnn the eight-bit character whose value is the octal value nnn (one to three digits)

\xHH the eight-bit character whose value is the hexadecimal value HH (one or two hex digits)

\cx a control-x character

Here is an example that substitutes a newline character and a tab:

echo First Line $'\n'Second Line with a $'\t'Tab

The result would look like:

First line

Second Line with a Tab

**Locale-specific translation** Any double-quoted string that is immediately preceded by a dollar sign will cause the string to be translated according to its current locale.

**Pipelines** A *pipeline* is used to send the output of one command directly to another command.

You will use the pike symbol (|) to denote a pipe. A good example is piping the result from a directory listing into the more command, which will enable you to view one page at a time in case your directory is rather large:

ls -al | more

**Redirection** *Redirection* refers to a few specific characters: < , >, <<, and >>. These characters are used to redirect output information from one command or function into another or into a file. You use the command

ll > listing.txt to push all the data retrieved from a long listing of the current directory into a file called listing.txt. You can use the opposite symbol, <, to push all of the content of a file into a command. The combination >> is used to append data to a file, and << is used to append data from a file into a command or command-line program.

**Understanding the Bash Environment**

After the Bash shell has been launched, a few configuration files are analyzed by the system to establish your shell’s appearance. Most important, the /etc/profile file is run before your home directory, also referred to as $HOME, is scanned for one of three files. In order they are .bash\_profile, .bash\_login, and .profile. After one of these files is located, it is read to determine a set of environment variables for your shell. In our Fedora distribution, the .bash\_profile is used, which also calls the .bashrc file. A list of the current environment variables that are set can be retrieved with the following command:

env

You will then be returned the following lines:

HOSTNAME=localhost.localdomain

TERM=xterm

SHELL=/bin/bash

HISTSIZE=1000

SSH\_CLIENT=192.168.0.20 2200 22

SSH\_TTY=/dev/pts/0

USER=efilson

LS\_COLORS=no=00:fi=00:di=00;34:ln=00;36:pi=40;33:so=00;35:

bd=40;33;01:cd=40;33;01:or=01;05;37;41:mi=01;05;37;41:

ex=00;32:\*.cmd=00;32:\*.exe=00;32:\*.com=00;32:\*.btm=00;32:

\*.bat=00;32:\*.sh=00;32:\*.csh=00;32:\*.tar=00;31:\*.tgz=00;31:

\*.arj=00;31:\*.taz=00;31:\*.lzh=00;31:\*.zip=00;31:\*.z=00;31:

\*.Z=00;31:\*.gz=00;31:\*.bz2=00;31:\*.bz=00;31:\*.tz=00;31:

\*.rpm=00;31:\*.cpio=00;31:\*.jpg=00;35:\*.gif=00;35:\*.bmp=00;

35:\*.xbm=00;35:\*.xpm=00;35:\*.png=00;35:\*.tif=00;35:

MAIL=/var/spool/mail/efilson

PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/efilson/bin

INPUTRC=/etc/inputrc

PWD=/home/efilson

LANG=en\_US.UTF-8

SSH\_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass

SHLVL=1

HOME=/home/efilson

LOGNAME=efilson

SSH\_CONNECTION=192.168.0.20 2200 192.168.0.18 22

LESSOPEN=|/usr/bin/lesspipe.sh %s

G\_BROKEN\_FILENAMES=1

\_=/bin/env

Almost any program can change, unset, and create these variables. Manipulation of these variables is the key to changing the look and functionality of your shell. If you wish to create any custom variables, you can edit the .bashrc or .bash\_profile files.

If you retrieve a full directory listing of your home directory (ls -al), excluding *root*’s home directory, you will also see a .bash\_history file. This file is used for storing the history of commands you have typed. The number of commands stored depends on an environment variable (which by default is set to 1000). This is what enables the shortcut Up arrow and Down arrow keys to scroll through your history. Go ahead and try reviewing your previous commands by using the Up arrow. You can then use the Left and Right arrows to move through your command line and make changes before pressing Enter to run the command. This is especially useful when typing commands containing long pathnames.

There are books written on just the shell itself, so we won’t spend too much time delivering all the details to you. You should know, however, that the Bash shell offers several improvements over both of its ancestors, including:

● Command-line editing

● Command history of unlimited size

● Job control

● Shell functions and aliases

● Indexed arrays of unlimited size

● Integer arithmetic in any base from 2 to 64

**Navigating through Linux**

If you are not used to using a command-line interface (for example, with the Bash shell or with

DOS), navigating the Linux filesystem might seem complicated at first, but do not fear. We’ve put together a list of basic commands that will make your job much easier. The commands contained in this section are absolutely paramount, and you will need to be deeply familiar—nay, intimate—with each of them. If this is your first time using Linux, you might want to throw a bookmark in the page so that you can refer back to this section while you get adjusted to this new system architecture. The following few pages will become an invaluable resource when beginning with Linux.

You should not be intimidated by the length of the list. After you begin using Linux, these commands will become second nature. If you need any additional information on each command, simply use the man command and pass in the function name as the only argument. This will take you to the manual page for the command. You can use the arrow keys to scroll up and down, and pressing Q will exit the manual. We’ve provided the following functions in alphabetical order to make it a bit easier should you need to reference this section again:

**cat** The cat command is used to concatenate files and print the output to the screen.

Because Linux is all about text files, you can use this command to view the contents of a file without entering an editor. Simply use cat *filename* to view a file. You can also pipe the results to the less or more command to allow for interactive scrolling.

**cd** The cd command stands for *change directory.*

The proper usage is cd *path*. This is probably the most common command you will be using for navigating your way through the Linux filesystem. You can also use the command by itself with no parameters to change to the home directory of the user who owns the shell you are currently in.

**chmod** chmod is a command used to change file permissions. File permissions in Linux are displayed when retrieving a long listing of a file or files. The first block on each line will have 10 characters. The first character denotes whether the listing is a directory, file, device, link, socket, or pipe. The character d means the listing is a directory, - is for a file, b is for a buffered device, c is for an unbuffered device, l is for a link, s is for socket, and p is for pipe. The next nine characters are split into three equal sections: user : group : world. Each section of three characters represents r for read, w for write, and x for execute. For example: drwxr-xr--

The preceding line means that the file is a directory; the user who owns the file has read, write, and execute permissions; the group who owns the file has read and execute permissions; and the *world* (meaning every other user on the system who is not the file owner and is not in the group who owns the file) has only read permissions. For the basics on file ownership, see the chown command. For now, just be familiar with the command. We will cover this in more detail in the next chapter, in the section “Managing Users and Groups.”

**chown** chown is used to modify the ownership of a file or directory. When retrieving a directory listing, you will find that the third and fourth blocks are username and group name, respectively. When chowning a file, simply use:

chown *flags username*:*groupname*

A common flag for this command is the -R flag, which indicates to perform this function recursively. *Recursive* means that it will repeat itself on each file contained in the directory specified and in each subdirectory of that directory, and so on. We will also be covering this command in depth in the next chapter. For now you should also be aware that you can leave off the colon and group name to change only the user who owns the file. If you wish to change just the group, you should use chgrp.

**cp** The cp command is the copy command. You can use this command to create duplicates of directories or files. A common flag combination for the cp command is -rfp, which stands for *recursive force preserve*. The preserve argument will preserve the ownership and timestamp of the original file. You can also use the -l command to create a link to the file instead of actually copying it. There are soft links and hard links in Linux. A *soft link* is much like a shortcut in Windows; it merely acts as a pointer to the existing file. A *hard link* is a new name for the target file, which will update itself whenever its twin is changed, and visa versa.

**exit** The exit command is used to exit a shell. If you have used the su command to log in as a different user, then typing **exit** will exit only that shell and not the entire virtual terminal.

**find** The find command is your friend; this command will help you locate files in your filesystem.

You might choose to use the locate command described later in this list, but it is important to note that the locate command is based on a database-type system that is updated every night at midnight. Any files created after last night at midnight will not be found. So instead, use the find command with the following syntax: find *path* -*name* '*filename string*'

You can also specify a myriad of other flags, such as time and size, to narrow your search. Use the man find command to learn more.

**grep** grep is used to find file(s) that contain a string matching the argument you pass in. By default, grep will print the line in the file(s) that match your argument. Here is the proper format for the grep command: grep *options pattern file(s)*

The following example will return the lines of any files inside /usr/local/apache/vdocs that match the term *PHP Freaks*: grep ‘PHP Freaks’ /usr/local/apache/vdocs/\*

The \* tells grep to search in any files located in the given directory. The asterisk is also known as a *wildcard*.

**ll** ll is just an *alias,* or shortcut, for the ls command with -al flags. The ll is short for *long listing* and is mentioned here to provide you with a shortcut for retrieving long directory listings.

Check the ls command for the appropriate syntax for this command.

**ls** The ls command is the directory listing command. There are a plethora of flags for this command to enable you to sort and limit your results. Running the man ls command will give you a complete listing. Some of the more popular flags are listed here:

-a Do not hide directory listings that begin with.

-l Use the long listing format

-r Reverse the order while sorting

-t Sort by modification time

-F Classify: this will append indicators to the listing such as \*/=@|

The following example will return all of the files, including those that begin with a period, starting with the oldest file first in long listing format: ls -rtal

This can be useful when trying to find newly added files. You can also specify a directory to list as an argument after your flags.

**man** The man command is used to access your manual. One of the most useful commands in Linux, the manual can provide a wealth of information on virtually every program or command on your system. Occasionally, when using the man function, you will come across a command with multiple entries. What makes these entries unique is their locations within different sections of the manual. If this is the case, you can use the -S flag followed by a colon separated list of sections to specify which sections to search.

**mkdir** mkdir is the command used to make (create) a new directory. Just pass in the name of your directory as the only argument, and your directory will be created. If any special characters are needed, make sure you enclose the directory name string in the appropriate quotes.

However, it is best to avoid using special characters in directories and filenames to avoid overcomplicating an otherwise simple command.

**mv** The mv command is used to move files in your system. The optional -f flag will force the move. Both directories and files can be moved in this manner. You can also use the mv command to move a file or directory to /dev/null*,* which will effectively delete the file. However, it is advised that you use the rm command covered later in this list.

**pico**. This command will be used throughout this book for its ease of use over vi for new users. A simple pico *filename* will open the file in the pico editor.

**pwd** pwd stands for *print working directory*. This command will return a line indicating the directory in which you are located. This can be especially useful if you are remotely logged in and wish to copy your current location into memory to paste or print later.

**rm** This command is your remove command. It is most often used with the -rf flags for *recursive* and *force.* But beware: this command will immediately remove your files. There is no *are you sure.* They are simply destroyed. Never to be seen again—by anyone. There is no undelete command to correct a mistake as in Windows. Even experienced users might slip and make a mistake with this one, so double-check your command before pressing Enter.

**su** The su command stands for *switch user.* The switch user command is used to change from user to user on the system. Using the exit command will close the current user section and return you to the previous user shell you were in. When remotely logging in to a system that has multiple users, it is generally good practice to log in as your own personal username and then su to *root* by using the su - command. This will populate the appropriate log files and histories with any commands that you enter. Later, if a mistake is found, you can find which account is responsible and take the appropriate action.

**tar** The tar command is another command you will use quite often. It is an archiving utility that now includes a built-in compression algorithm called gzip and another named

Bzip2. You will often hear of files that contain the .tar extension as *tarballs.* This book will teach you how to download tarballs of your needed programs and install them. To untar a file, use this command: tar -xvf *filename*

The optional -z flag is used if the file has a .z or .gzip extension. The -z flag when unarchiving a tar file will automatically decompress it as well. To create a tar archive, use this command:

tar -cvfz *filename files*

The preceding line will create a gzip tarball with a filename of *filename* and will include the files and/or directories specified for the *files* argument. For a Bzip2 file you would use the -j switch.

**vi** vi stands for *visual editor.* This classic editing program for Unix houses a powerful set of functionality for finding and replacing and repeating operations to make changes and produce files very rapidly. Its abundance of functionality produces excellent results; however, it acts as a double-edged sword because it is complicated for a beginning user to grasp.

**whoami** The whoami command will tell you exactly *who* you are. Not in the metaphysical sense, but which user you are in the system. Some installation commands will vary depending on whether you are *root*. Checking who you are can oftentimes bypass some frustration when programs don’t seem to compile correctly.

These commands should give you an excellent foundation for navigating through the Linux filesystem. You will need to use these commands throughout your affair with Linux, so committing these to memory now is an excellent idea. If the majority of these commands are new to you, remember to place a bookmark in the beginning of this section for future reference.

After you have done that, you will be ready to head into the next section.

**Understanding the Linux Filesystem Layout**

The Linux filesystem layout has been organized carefully so those with a basic understanding of the filesystem can find files quickly and easily. Although it might not seem so at first, after you grow accustomed to the layout, you will breeze through directories targeting the exact locations you need. Strict naming conventions and storage locations for files enable Linux to continue its open source development without anyone getting in anyone else’s way. Let’s take a look inside each of the root-level directories and the files they contain. Some of the more important subdirectories and their files are also included:

drwxr-xr-x 2 root root 4096 Jan 9 11:37 bin

drwxr-xr-x 4 root root 1024 Jan 8 16:41 boot

drwxr-xr-x 21 root root 118784 Jan 9 10:19 dev

drwxr-xr-x 57 root root 4096 Jan 13 04:02 etc

drwxr-xr-x 3 root root 4096 Jan 9 10:15 home

drwxr-xr-x 2 root root 4096 Oct 7 07:16 initrd

drwxr-xr-x 9 root root 4096 Jan 9 11:33 lib

drwx------ 2 root root 16384 Jan 8 11:38 lost+found

drwxr-xr-x 2 root root 4096 Sep 8 14:11 misc

drwxr-xr-x 3 root root 4096 Jan 9 10:19 mnt

drwxr-xr-x 2 root root 4096 Oct 7 07:16 opt

dr-xr-xr-x 49 root root 0 Jan 9 05:18 proc

drwxr-x--- 13 root root 4096 Jan 13 15:09 root

drwxr-xr-x 2 root root 8192 Jan 9 11:37 sbin

drwxrwxrwt 6 root root 4096 Jan 13 04:02 tmp

drwxr-xr-x 15 root root 4096 Jan 8 16:40 usr

drwxr-xr-x 19 root root 4096 Jan 8 16:49 var

**/bin**

The /bin directory contains all the essential binary programs and commands needed for Linux.

Go ahead and change directories into the /bin directory and retrieve a directory listing. Here you will see the majority of commands listed in the previous section, excluding commands that are built into the Bash shell. As you install software and packages on your computer, this directory will expand as new priority commands are added.

**/boot**

The /boot directory contains status files of the boot loader and the Linux kernel. Go ahead and cd into this directory and take a look—not really too much to see here, mostly NPTL files (Native POSIX Thread Library files) required for your boot sequence. This is also where the

GRUB files are located, inside the directory /boot/grub.

**/dev**

The /dev directory holds the special device files. Taking a look at this directory will yield some interesting results. If you are retrieving a long listing, you will see tens of pages of results. Each one of these represents a specific device on your system, whether it is a hard (hardware) or soft (software) device. Look at where the file size is normally located; it has been split into two parts.

The first part represents the *major device number;* this is a kind of parameter passed to the kernel when accessing the device and represents the device driver. The second part represents the *minor device number* and is passed to the kernel to allow further device selection.

As an example, look at the hard disk device files. These are preceded by the characters hd for IDE drives. Your first hard disk is hda. If you perform ll hda\*, you will be able to see each minor device listed as well. Notice the first device number column reads 3 because it is the third device. Each listing then has a different *minor device number,* which pertains to the partition on that device, or in the case of *hda*, the 0 specifies accessing the entire drive.

**WARNING** Accessing the “files” in the device directory means accessing some sort of device and *not* the filesystem. Piping data to one of these devices could have a severely unexpected effect.

Some other important devices you might want to take note of include mouse, CD-ROM, CD writer, scanner, and modem. Any media devices can usually be mounted to a specific directory by using the mount command. For instance, running the command mount *device location* will mount the device specified to the location you enter. It is most commonly mounted as the device name under the /mnt directory (for example, /dev/cdrom mounted to /mnt/cdrom).

**/etc**

The /etc directory holds configuration files specific to your computer. Almost all major configuration files are housed here, and applications that require more than a few files will have their own subdirectories. Retrieving a listing of this directory will astound you because of the number of files located here. The sheer volume of files serves as proof to the power of Linux and its ability to be customized.

Let’s look at some of the more important files located in this directory. We have sorted them by category for you for quick reference, as shown in Tables 3.1–3.3.

TABLE 3.1 Network-Related Files

**File Description**

exports Network File System (NFS) server exports

host.conf Configuration for resolving domain names

hosts List of hosts with IP addresses

hosts.allow TCP wrapper allowed hosts

hosts.deny TCP wrapper deny hosts

protocols A list of protocols with descriptions

resolv.conf Configuration for resolving domain names

services Services with their ports/protocols

You will be using a lot of these files frequently as you customize your server to your needs.

TABLE 3.2 Shell Profiles

**File Description**

bashrc The default for all users

csh.cshrc The C shell profile

csh.login C shell login functionality profile Bash shell profile shells List of all shells for your system

TABLE 3.3 Miscellaneous System Files

**File Description**

Passwd User definitions

shadow Shadow password file with MD5 passwords

group Group definitions

gshadow Shadow group file

aliases System and mail aliases

aliases.db System and mail aliases

crontab System crontab configuration, which holds a list of commands to run on a schedule

fstab List of filesystems and mount points

inittab init process configuration

motd Message off the day

mtab List of currently mounted filesystems

printcap Printer configuration

sysctl.conf Kernel configuration for Fedora

syslog.conf Configuration for logging

termcap Terminal type descriptions

Besides the files that are located within this directory, there are also quite a few important

subdirectories as well. Let’s take a look at some:

**/etc/cron.d** This directory houses scripts run by the crontab file. The crontab is a file parsed by the cron daemon (crond) and determines at what times a script should be run.

**/etc/cron.daily** The daily cron directory will run any scripts located within it once a day. You do not need to have a crontab entry for the scripts located here.

**/etc/cron.hourly** Like the /etc/cron.daily directory, this directory does not require scripts located within it to have a separate crontab entry. The only difference here is that, much like the name suggests, the scripts are run hourly.

**/etc/cron.monthly** Scripts located within this directory also do not need a separate cron tab entry. The only difference here is that, much like the name suggests, the scripts are run monthly.

**/etc/cron.weekly** Like all of its brothers, the files here are run automatically. The weekly extension means that the files will be run weekly.

**/etc/default** The /etc/default directory contains the useradd file by default (no pun intended). This file contains the following:

# useradd defaults file

GROUP=100

HOME=/home

INACTIVE=-1

EXPIRE=

SHELL=/bin/bash

SKEL=/etc/skel

Each time a new user is created, these values will be assigned to the user. When customizing your system, you might wish to change the values.

**/etc/httpd** This is where your web server configuration files are located for Apache.

There will be much more about this later in Chapter 8, “Apache Web Server: Installation and

Configuration.”

**/etc/mail** This directory contains configuration files for send mail, a program used to send mail from your system. There are various configuration files pertaining to access levels and users located here.

**/etc/pcmcia** This directory contains all of the configuration and startup files necessary for Personal Computer Memory Card International Association (PCMCIA) operations. You will most likely not need these unless you are using a laptop.

**/etc/ppp** Files pertaining to your Point-to-Point Protocol (PPP) are located here. PPP is used

for Internet traffic. Your firewall-masq and stand-alone configuration files are also located here.

**/etc/profile.d** This directory contains other profile information for the system shells.

The color creation for your ls command is kept here as well as the alias for ll. Language files and less configuration can be found in the lang\* and less\* files.

**/etc/rc.d** We covered the files in this directory extensively earlier in this chapter. To summarize, these files are used during the initial boot sequence in Linux to set up your filesystems and basic commands. If you would like more information on these files, see the earlier section “The rc Script and its Kill and Startup Files.”

**/etc/rpm** Earlier in this chapter, you used rpm to install pine*,* which includes pico. The configuration files for rpm are contained in this directory. There are relatively few files contained in this directory. The macro files contain pre-built commands that rpm uses internally, and the platform file contains a single line that states your system type.

**/etc/security** The /etc/security directory houses files that set permissions for specified users or groups pertaining to login access and limits.

**/etc/ssh** The /etc/ssh directory contains configurations and keys for the Secure Shell login procedure. A system of *keys* is used for security when you access the server remotely in order to transfer encrypted information. Modifying the ssh\_config file will change settings for all users who access the system remotely through ssh.

**/etc/sysconfig** Changing to this directory and retrieving a directory listing will present you with many of the files used in configuring your system. You can read /usr/share/doc/ initscripts-*x.xx*/sysconfig.txt for the complete list with all options. The *x.xx* is the Linux kernel version. Here are some of the major configuration files contained in this directory and a few of the options available within each one:

**/etc/sysconfig/authconfig** used by authconfig to store information about the system'suser information and authentication setup; changes made tothis file have no effect until the next time authconfig isrun

USEHESIOD=no

Whether or not the hesiod naming service is in use.

If not set, authconfig examines the passwd setting in /etc/nsswitch.conf.

USELDAP=no

Whether or not LDAP is used as a naming service. If not set, authconfig examines the passwd setting in /etc/nsswitch.conf.

USENIS=no

Whether or not NIS is in use. If not set, authconfig examines the passwd setting in /etc/nsswitch.conf.

USEKERBEROS=no

Whether or not Kerberos is in use. If not set, authconfig examines the settings in

/etc/pam.d/system-auth.

USELDAPAUTH=no

Whether or not LDAP is being used for authentication.

If not set, authconfig examines the settings in

/etc/pam.d/system-auth. Note that this option is separate from USELDAP, and that neither implies the other.

USEMD5=no

Whether or not MD5-based hashing should be used when setting passwords. If not set, authconfig examines the settings in /etc/pam.d/system-auth. This option affects authentication using both local files and

LDAP.

USESHADOW=no

Whether or not shadow passwords are in use. If not set, authconfig checks for the existence of /etc/shadow.

USESMBAUTH=no

Whether or not SMB authentication is in use. If not set, authconfig examines the settings in /etc/pam.d/system-auth.

**/etc/sysconfig/clock** deprecated values from earlier releases:

CLOCKMODE=GMT indicates that the clock is set to UTC

CLOCKMODE=ARC on alpha only indicates the ARC console's 42-year time offset is in effect currently correct values:

UTC=true,yes Indicates that the hardware clock is set to UTC.

UTC=no,false Indicates that the hardware clock is set to Local Time.

Not having UTC set defaults to the last used (if recorded in the adjtime file), or to localtime, if not adjtime file exists.

ARC=true on alpha only indicates the ARC console's 42-year time offset is in effect; otherwise the normal Unix epoch is assumed.

SRM=true on alpha only indicates the SRM 1900 epoch is in effect; otherwise the normal Unix epoch is assumed.

ZONE="filename" indicates the zonefile under /usr/share/zoneinfo that /etc/localtime is a copy of, for example:

ZONE="US/Eastern"

**/etc/sysconfig/desktop**

DESKTOP=GNOME|KDE|AnotherLevel

This determines the default desktop for new users.

DISPLAYMANAGER=GNOME|KDE|XDM

This determines display manager started by /etc/X11/prefdm, independent of the desktop.

**/etc/sysconfig/init**

BOOTUP=<some bootup mode>

BOOTUP=graphical means use X Windows graphical boot up

BOOTUP=color means colorized text mode boot display.

BOOTUP=verbose means old style display Anything else means simplified display, but without color or ANSI formatting

LOGLEVEL=<a number>

Sets the initial console logging level for the kernel. The default is 7. 8 means everything (including debugging); 1 means nothing except kernel panics. syslogd will override this once it starts.

RES\_COL=<a number>

Column of the screen to start status labels at.

Defaults to 60

MOVE\_TO\_COL=<a command>

A command to move the cursor to $RES\_COL.

Defaults to nasty ANSI sequences output by echo -e.

SETCOLOR\_SUCCESS=<a command>

A command to set the color to a color indicating success. Defaults to nasty ANSI sequences output by echo -e setting the color to green.

SETCOLOR\_FAILURE=<a command>

A command to set the color to a color indicating failure. Defaults to nasty ANSI sequences output by echo -e setting the color to red.

SETCOLOR\_WARNING=<a command>

A command to set the color to a color indicating warning. Defaults to nasty ANSI sequences output by echo -e setting the color to yellow.

SETCOLOR\_NORMAL=<a command>

A command to set the color to 'normal'. Defaults to nasty ANSI sequences output by echo -e.

PROMPT=yes|no

Set to 'no' to disable the key check for interactive mode.

**/etc/sysconfig/keyboard**

KEYTABLE=<keytable file> for example: KEYTABLE="/usr/lib/kbd/keytables/us.map"

If you dump a keymap (using 'dumpkeys') to /etc/sysconfig/console/default.kmap it will be loaded on bootup before filesystems are mounted/checked.

This could be useful if you need to emergency type the root password. This has to be a dumped keymap, as opposed to copying the shipped keymap files, as the shipped files include other maps from the /usr/lib/kbd/keytables directory.

KEYBOARDTYPE=sun|pc on SPARC only, sun means a sun keyboard is attached on /dev/kbd, pc means a PS/2 keyboard is on ps/2 port.

**/etc/sysconfig/mouse**

MOUSETYPE=microsoft|mouseman|mousesystems|ps/2|msbm|logibm|atibm|logitech|mmseries|mmhittab

XEMU3=yes|no (emulate three buttons with two buttons whenever necessary, most notably in X)

DEVICE=<a device node> (the device of the mouse)

**/etc/sysconfig/network**

NETWORKING=yes|no

HOSTNAME=<fully qualified domain name by default>

GATEWAY=<gateway IP>

GATEWAYDEV=<gateway device>

(e.g. eth0)

NISDOMAIN=<nis domain name>

VLAN=yes|no

IPX=yes|no

IPXAUTOPRIMARY=on|off (note, that MUST be on|off, not yes|no)

IPXAUTOFRAME=on|off (again, not yes|no)

IPXINTERNALNETNUM=<netnum>

IPXINTERNALNODENUM=<nodenum>

All the IPX stuff is optional, and should default to off.

NETWORKING\_IPV6=yes|no

Enable or disable global IPv6 initialization

Default: no

IPV6FORWARDING=yes|no

Enable or disable global forwarding of incoming IPv6

packes on all interfaces. Note: Actual packet forwarding cannot be controlled per-device, use netfilter6 for such issues Default: no

IPV6INIT=yes|no

Enable or disable IPv6 configuration for all interfaces Use with caution! Default: value not set in this file

IPV6\_AUTOCONF=yes|no

Sets the default for device-based autoconfiguration.

Default: yes if IPV6FORWARDING=no, no if

IPV6FORWARDING=yes

IPV6\_ROUTER=yes|no

Sets the default for device-based Host/Router

behaviour. Default: yes if IPV6FORWARDING=yes, no if

IPV6FORWARDING=no

IPV6\_AUTOTUNNEL=yes|no

Controls automatic IPv6 tunneling.

Default: no

IPV6\_DEFAULTGW=<IPv6 address[%interface]> (optional) Add a default route through specified gateway An interface can be specified: required for link-local addresses

IPV6\_DEFAULTDEV=<interface> (optional) Add a default route through specified interface without specifying next hop. Type of interface will be tested whether this is allowed

IPV6\_RADVD\_PIDFILE=<pid-file> (optional)

Location of PID file for controlling radvd

**/etc/sysconfig/rawdevices**

This is used for setting up raw device to block device mappings. It has the format:

<rawdev> <major> <minor>

<rawdev> <blockdev>

For example:

/dev/raw/raw1 /dev/sda1

/dev/raw/raw2 8 5

**/etc/sysconfig/pcmcia**

PCMCIA=yes|no

PCIC=i82365|tcic

PCIC\_OPTS=<socket driver timing parameters>

CORE\_OPTS=<pcmcia\_core options>

CARDMGR\_OPTS=<cardmgr options>

**/etc/sysconfig/saslauthd** used by the saslauthd init script (part of the cyrus-saslpackage) to control which arguments are passed to saslauthdat startup time; changes made to this file have no effectuntil saslauthd is restarted

MECH=shadow controls which data source saslauthd will consult when checking user passwords; run 'saslauthd -a' to get a full list of available authentication mechanisms

SOCKETDIR=/var/run/saslauthd controls in which directory saslauthd will be directed to create its listening socket; any change to this value will require a corresponding change in client configuration files **/etc/sysconfig/sendmail**

DAEMON=yes|no yes implies -bd (i.e., listen on port 25 for new mail)

QUEUE=1h given to sendmail as -q$QUEUE -q option is not given to sendmail if /etc/sysconfig/sendmail exists and

QUEUE is empty or undefined.

**/etc/sysconfig/i18n**

LANG= set locale for all categories, can be any two letter

ISO language code

LC\_CTYPE= localedata configuration for classification and conversion of characters

LC\_COLLATE= localedata configuration for collation (sort order) of strings

LC\_MESSAGES= localedata configuration for translation of yes and no messages

LC\_NUMERIC= localedata configuration for non-monetary numeric data

LC\_MONETARY= localedata configuration for monetary data

LC\_TIME= localedata configuration for date and time

LC\_ALL= localedata configuration overriding all of the above

LANGUAGE= can be a : separated list of ISO language codes

LINGUAS= can be a ' ' separated list of ISO language codes

**/etc/sysconfig/harddisks**

/etc/sysconfig/harddiskhd[a-h] (for specific devices) These options are used to tune (E)IDE hard drives and other devices - read the hdparm man page for more information USE\_DMA=1 Set this to 1 to enable DMA. This might cause some data corruption on certain chipset / hard drive combinations. USE WITH CAUTION AND BACKUP. This is used with the "-d" option

MULTIPLE\_IO=16

Multiple sector I/O. a feature of most modern IDE hard drives, permitting the transfer of multiple sectors per I/O interrupt, rather than the usual one sector per interrupt. When this feature is enabled, it typically reduces operating system overhead for disk I/O by 30-50%. On many systems, it also provides increased data throughput of anywhere from 5% to 50%. Some drives, however (most notably the WD Caviar series), seem to run slower with multiple mode enabled. Under rare circumstances, such failures can result in massive filesystem corruption. USE WITH

CAUTION AND BACKUP. This is the sector count for multiple sector I/O - the "-m" option EIDE\_32BIT=3

(E)IDE 32-bit I/O support (to interface card). USE WITH CAUTION.

LOOKAHEAD=1

Enable drive read-lookahead (safe)

EXTRA\_PARAMS=<anything>

Add any extra parameters you want to pass to hdparm here.

We have purposely left off /etc/sysconfig/network-scripts because it will be covered more in depth in **/home**

The /home directory contains all of the users’ home directories. It also contains each user’s preferences for shell options and X Window interface options. Performing the ls command with the -a option in a user’s home directory will show that each of the files begins with a dot (period) in order to hide the files from a normal listing. You will remember these files from the earlier discussion in the “Bash Environment” section earlier in this chapter. When a new user is added to your system, that user will have a directory listed under /home created for them automatically.

**/lib**

The /lib directory stands for *library.* The library contains the shared files needed to run the binaries in the /root filesystem. This directory also has a subdirectory called /lib/modules*,* which contains the dynamic loadable modules for the Linux kernel. The directory structure is set up as /lib/modules/version/. The version number will be the value of the Linux kernel you are using. In the case of Fedora Core 2, it will be 2.4.22-1.2115.nptl.

**/lost+found**

Your /lost+found directory is used by the filesystem should any files become corrupted. Earlier, we discussed how the system might ask you during the boot process to run a system check via the fsck command if your system was not shut down properly. If fsck then discovers any files that were damaged, they will be placed in this directory. Each mounted drive will have its own directory for just such an occasion.

If you find any files here, you should attempt to place them back where they originated and whatever you do, do not remove this directory. If the filesystem needs to recover a lost file and needs to access this directory, you could experience some unwanted results. You should also note that the files here will have most likely lost their original names. You might want to open them in an editor or cat the contents to try to determine what the file actually is.

**/mnt**

This directory is used as a generic mounting location for devices on your system. Chances are your CD-ROM drive has already been mounted here under /mnt/cdrom. Although mounting devices here is not necessary, it makes them easier to locate rather than having to remember or guess where a device was mounted.

**/opt**

The /opt directory, short for *optional,* contains software that is in addition to your server software. Examples of such software are browsers, games, or X Window editors. Not a lot of programs install their files here, only a few. The majority of your X11 applications will most likely be installed to /usr/X11R6. This directory, if you installed our recommended packages, should be empty for now. If you choose to run the X Window interface, some of your software may or may not be installed here.

**/proc**

This directory is your kernel and system configuration parameter directory. This directory resides in your system memory rather than on an actual drive partition. Many of the files in this directory cannot be written to, even by the *root* user. The numbered directories, or directories with an integer name, are the numerical values for each running process. The following is a list of other files and a short description of what each one does:

**NUMBERED DIRECTORIES CONTAIN THE FOLLOWING SUBDIRECTORIES**

cwd This is a link to the current working directory of the process.

environ This file contains the environment for the process.

Exe Under Linux 2.2 and 2.4 exe is a symbolic link containing the actual path name of the executed command.

fd This is a subdirectory containing one entry for each file which the process has open, named by its file

descriptor, and which is a symbolic link to the actual file (as the exe entry does).

Maps A file containing the currently mapped memory regions and their access permissions.

mem Via the mem file one can access the pages of a processes memory through open(2), read(2), and fseek(3).

root Unix and Linux support the idea of a per-process root of the filesystem, set by the chroot(2) system call.

stat Status information about the process.

statm Provides information about memory status in pages.

HERE ARE THE OTHER DIRECTORIES UNDER /proc

apm Advanced power management version and battery information when CONFIG\_APM is defined at kernel

compilation time.

bus Contains subdirectories for installed busses.

cmdline Arguments passed to the Linux kernel at boot time.

cpuinfo This is a collection of CPU and system architecture dependent items, for each supported architecture a

different list.

devices Text listing of major numbers and device groups.

dma This is a list of the registered ISA DMA (direct memory access) channels in use.

driver Empty subdirectory.

execdomains List of the execution domains (ABI personalities).

Fb Frame buffer information when CONFIG\_FB is defined during kernel compilation.

Filesystems A text listing of the filesystems which were compiled into the kernel.

Fs Empty subdirectory.

Ide ide exists on systems with the ide bus.

interrupts This is used to record the number of interrupts

per each IRQ on (at least) the i386 architecture.

iomem I/O memory map in Linux 2.4.

ioports This is a list of currently registered Input-Output port regions that are in use.

kcore This file represents the physical memory of the system and is stored in the ELF core file format.

kmsg This file can be used instead of the syslog(2) system call to read kernel messages.

ksyms This holds the kernel exported symbol definitions used by the modules(X) tools to dynamically link and

bind loadable modules.

loadavg The load average numbers give the number of jobs in the run queue.

locks This file shows current file locks (flock(2) and fcntl(2)) and leases (fcntl(2)).

malloc This file is only present if CONFIGDEBUGMALLOC was defined during compilation.

meminfo This is used by free(1) to report the amount of free and used memory (both physical and swap) on the

system as well as the shared memory and buffers used by the kernel.

mounts This is a list of all the file systems currently mounted on the system.

Modules A text list of the modules that have been loaded by the system.

mtrr Memory Type Range Registers.

net various net pseudo-files, all of which give the status of some part of the networking layer.

partitions Contains major and minor numbers of each partition as well as number of blocks and partition name.

pci This is a listing of all PCI devices found during kernel initialization and their configuration.

scsi A directory with the scsi midlevel pseudo-file and various SCSI low level driver directories, which contain

a file for each SCSI host in this system, all of which give the status of some part of the SCSI IO

subsystem.

self This directory refers to the process accessing the /proc filesystem, and is identical to the /proc directory

named by the process ID of the same process.

slabinfo Information about kernel caches.

stat kernel/system statistics.

swaps Swap areas in use.

sys This directory (present since 1.3.57) contains a number of files and subdirectories corresponding to kernel

variables.

sysvipc Subdirectory containing the pseudo-files msg, sem and shm.

tty Subdirectory containing the psuedo-files and subdirectories for tty drivers and line disciplines.

uptime This file contains two numbers: the uptime of the system (seconds), and the amount of time spent in idle

process (seconds).

version This string identifies the kernel version that is currently running.

The preceding code is a summarized version of the manual entry for proc. You can find more specifics on each one of these by reading the manual entry. Remember that when in the man command, you can press Q to exit along with Ctrl+C.

**/root**

This is the home directory for the *root* user. In addition to your own files, various log files will be created here for you to review should programs or processes head south.

**TIP** Keep your files and directories organized. Starting a regimen of creating categorized directories and always placing the appropriate files in them will save massive amounts of time when you are searching for things.

**/sbin**

The /sbin directory contains binary programs required for booting the system that are not contained in /bin. Retrieve a listing of the /sbin directory and you will see that all users of the system have access to execute almost any of these commands. The /sbin directory was separated from /bin in order to create a distinction between commands that the system uses and commands that might be applicable to users of the system. Originally this separation stemmed from creating a separate directory for files that were *statically linked,* meaning they did not require any other supporting files or libraries to work. This way they could still be used in a single-user environment or when filesystems refuse to mount.

**/tmp**

This directory contains any temporary files currently in use by programs running on the system.

Any user of the system can write to this directory, including users owned and operated by commands and programs. Fedora does not execute a cleanup of this directory on a regular basis, but you might wish to have a script run at boot or, if your system is heavily trafficked,more often by using a cron job.

**/usr**

The /usr directory is a *major* section of your filesystem. It contains files that are shared between users and should not be writeable by anyone but *root*. Some major software packages such as Apache will also install under their own subdirectory here. The following is a list of the subdirectories and a brief description of their content:

**/usr/bin**

Most user commands are stored here. All binaries that do not belong in the /bin or /sbin directories will be found here.

**/usr/dict**

This directory holds dictionary files for multiple language support.

**/usr/etc**

This directory contains configuration files for non-systemwide programs such as the programs and commands contained in /usr/bin and /usr/sbin.

**/usr/games**

Any default-installed games will be installed in this directory. Because this system will be used as a server, we do not cover them in this book.

**/usr/include**

This is the proper directory for standard include files. Different programs that you can install will deposit C and C++ files here for usage within their functionality. This allows for easy software development in Linux by using other open source include files.

**/usr/kerberos**

*Kerberos* is a network authentication protocol designed to provide strongkey-based encryption for information transferred via secure sockets and connections.

**/usr/lib**

These files, much like the /lib directory’s files, are for standard library files. Theycan be used by any programs installed, as well as by your own if you develop software for Linux.

**/usr/libexec**

The /usr/libexec directory holds system daemons and commands run strictly by other programs.

**/usr/local**

The /usr/local directory is used when an administrator account is installing software locally. Apache and MySQL will install themselves here unless otherwise directed during the installation procedures.

**/usr/sbin**

This directory houses programs and binaries for the system administrator that are not absolutely necessary for standard system operation. Repair files, system daemons, and administration files are kept here. Although most of them show permissions that allow any user to execute them, each one handles its own permissions whether or not the current user is allowed to run the command.

**/usr/share**

The /share directory contains files that are not dependent on a specific architecture. These files can be run by i386, Alpha, or any other architecture without encountering problems. The /usr/share directory is also used to store documentation and sample configuration files for applications.

**/usr/src**

This directory contains the kernel source code for your system. Advanced users can modify the files contained in this directory and recompile their kernel in order to create a highly customized system. This type of modification is *not* recommended for beginning users.

**/usr/tmp** This is simply a link to the /var/tmp directory.

**/usr/X11R6** This directory contains the X Window system and all of its related files. The host-specific information is contained in the /usr/X11R6/lib/X11 directory.

**/var**

The /var directory is used for files that can be shared or are in a transient state. Data that is cached, locked, spooled, and logged will be in the appropriate subdirectory under /var. If you need to find why a program is not functioning properly, you can look at its log files most likely contained here.

**Managing Users and Groups**

User accounts serve a number of purposes on any Linux system. Primarily, they give the system and other users a way to distinguish themselves and the files they own from other users and their files. When we refer to an *account,* we are talking about the user’s name and all of the files and directories owned by the user.

In addition to accounts owned by people, there are also accounts owned by programs and processes that reside on your system. This level of distinction for the files a program touches or modifies is needed for multiple reasons, the first of which is security. You would not want a program that is accessible to any user on your system to have privileges allowing it to modify files and directories that only the *root* user should be allowed to access.

Most of these processes that are running in the background are referred to as *daemons.*

For example, you might have a daemon running that retrieves updated files from a few servers via

HTTP or FTP. This daemon might need to store its files in the /var/spool directory so that anyone can read them. This daemon user would have full access to these files, whereas each of the other users, who are owned and operated by a person, would have only read access to these files in order to prevent tampering and to prevent an inexperienced user from accidentally deleting the file(s).

As the system administrator, it is your responsibility to create each of these accounts and assign the proper levels of access to each one. This should not be taken lightly because mistakes could easily lead to your system being compromised and ultimately to you losing complete control over your system. After a malicious user gains access to your *root* account, they can completely erase all data your system has access to.

*Groups,* as defined by Red Hat, are “logical expressions of organization, tying users togetherfor a common purpose.” Groups help you separate types of users from each other and alloweach user in a group to have the same level of access to common files and directories that theymight share. When you set up your web server, you might wish to give a friend or coworkeraccess to the web directories. Adding this person to the group that your web server user ownswould allow them to access the group permissions of files that the web server group owns. Wewill cover this more in depth later in this section.

**The */etc/passwd* File**

For now, let’s take a look at an account, its login information, and its files. You’ll start with

The /etc/passwd file. The passwd file contains one line for each user on your system. Move to your /etc directory now and pico the passwd file. You can read the information in this file by using the following format:

*login*

:

*password*

:

*UID*

:

*GID*

:

*Full Name*

:

*homedir*

:

*shell*

The login is also referred to as your

*username.*

Notice that the password is shown as an *x.*

This is because Linux stores your password in a shadow password file. The shadow password file is used for security reasons. Your shell, which runs at your user and access level, requires access

to the passwd file in order to retrieve your full name and home directory. However, allowing every user to see the passwords would open the doors for malicious users attempting to crack the password file and gain access to your system. For this reason, the actual encrypted passwords are stored elsewhere.

The User ID (UID) and Group ID (GID) fields are integer values and can be modified directly in this file.

**NOTE**

This GID is the user’s *primary* GID. Others can be assigned via the group file covered later.

If you scroll to the bottom of this file, you will see the first user you created during the installation procedure at the very bottom. The UID and GID are both set to 500. This is what is referred to as User Private Groups (UPGs). UPGs are used to make default permissions for files and directories created by a user. These default permissions will ensure that no other users of the system will be able to change or delete these files. This is because the file will be owned by a user and a group that are unique.

The following default users should be installed on your system if you are running Fedora

Core 2. Almost all of the default users will be at a value less than 100, with root always being 0.

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

halt:x:7:0:halt:/sbin:/sbin/halt

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

news:x:9:13:news:/etc/news:

uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

gopher:x:13:30:gopher:/var/gopher:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/:/sbin/nologin

rpm:x:37:37::/var/lib/rpm:/sbin/nologin

vcsa:x:69:69:virtual console:/dev:/sbin/nologin

nscd:x:28:28:NSCD Daemon:/:/sbin/nologin

sshd:x:74:74:Priv-sep SH:/var/empty/sshd:/sbin/nologin

rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin

rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin

**nfsnobody:x:65534:65534:Anon NFS:/var/lib/nfs:/sbin/nologin**

mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin

smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin

pcap:x:77:77::/var/arpwatch:/sbin/nologin

apache:x:48:48:Apache:/var/www:/sbin/nologin

dbus:x:81:81:System message bus:/:/sbin/nologin

xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin

named:x:25:25:Named:/var/named:/sbin/nologin

ntp:x:38:38::/etc/ntp:/sbin/nologin

desktop:x:80:80:desktop:/var/lib/menu/kde:/sbin/nologin

gdm:x:42:42::/var/gdm:/sbin/nologin

On line 23 is the user nfsnobody . This user has a UID and GID of 65534, which is the maximum allotted user number. No single server should ever need this many users for any reason.

You can manually add lines here to add a user, but it is recommended that you use the industry standard command-line method of useradd covered later in this chapter.

**The */etc/group* File**

This file contains virtually all of the relationships for assigning groups to users and vice versa.

Exit the/etc/passwd file if it is still open and pico the /etc/group file. A list of default groups will be displayed. To read these, use this format:

*groupname*

:

*password*

:

*GID*

:

*members*

The Groupname is much like a username for a group. This is what is displayed under a long listing of directories’ contents in the group ownership column. The next field, password , is optional.

You might wish to add a password for a group so that users of the system can add themselves to the group, for the duration of their shell’s existence, with the newgrp command. These passwords are stored in a shadow file similar to the shadow file for /etc/passwd. This can cut down on administration time if your system supports a large number of users. The GID field is the same ID that is used in the /etc/passwd file to specify a user’s default group.

With a default installation, your /etc/group file should read as follows:

root:x:0:root

bin:x:1:root,bin,daemon

daemon:x:2:root,bin,daemon

sys:x:3:root,bin,adm

adm:x:4:root,adm,daemon

tty:x:5:

disk:x:6:root

lp:x:7:daemon,lp

mem:x:8:

kmem:x:9:

wheel:x:10:root

mail:x:12:mail

news:x:13:news

uucp:x:14:uucp

man:x:15:

games:x:20:

gopher:x:30:

dip:x:40:

ftp:x:50:

lock:x:54:

nobody:x:99:

users:x:100:

rpm:x:37:

floppy:x:19:

vcsa:x:69:

utmp:x:22:

slocate:x:21:

nscd:x:28:

sshd:x:74:

rpc:x:32:

rpcuser:x:29:

nfsnobody:x:65534:

mailnull:x:47:

smmsp:x:51:

pcap:x:77:

apache:x:48:

dbus:x:81:

xfs:x:43:

named:x:25:

ntp:x:38:

desktop:x:80:

gdm:x:42:

Notice that each default user has a UPG assigned to it. You should also note that the username is not required on its own UPG. For example, named*,* which is the UPG for named*,* does not need its own username listed at the end of the line. A group with more than one user assigned should list its own UPG owner. However, even if you wish a user to have root access to your system, it is best not to assign that user to the root group. It is best to have users log in as themselves and

su to root . This just serves to keep security as tight as possible.

**Linux Passwords**

Linux passwords, as discussed earlier in this chapter, are stored in shadow files. The shadow files used by Fedora Core 2 are shadow and gshadow for users and groups, respectively, and are located with the non-shadow files in the /etc directory.

Editing these files will show an encrypted string of characters. In order to be stored, the password has been encrypted by using a one-way encryption algorithm (or hash) called Message

Digest 5 (MD5). This means that the encrypted string is never translated back into its original form to be matched. Instead, the password you type when logging in is MD5 encrypted and then matched against the stored encrypted string. This ensures that if the password file is stolen, it will still be *relatively* secure. The term *relatively* is used liberally here because the password file can still be cracked.

Methods used to crack a password file of this type are commonly referred to as *brute force methods.*

The cracking program must try to guess the password by encrypting its guess and matching it against the encrypted string contained in the stolen passwd file. This is why short passwords or passwords based on a single word are extremely bad. Usually the first method tried by someone trying to crack the file is to use every dictionary word.

The installed package that allows for shadowing to be enabled is accompanied by some extra functionality. This extra functionality contains the commands enabling you to add, edit, and delete users and groups as well as the commands for password aging and expiration. Password expiration enables you to specify a set amount of time until a user has to change their password.

This also helps to tighten security on your server.

**User Administration**

Creating a new account on your system is done with the useradd command. The useradd command can also be used to update a user’s information by passing in different flags. The following is the proper format for useradd:

useradd -*flag[s] login*

This command creates a single user with a username of login. You will then be taken to another command line with no message printed.

After initially creating the user, you will need to set a password. To set a user’s password, use this command:

passwd *username*

If you are logged in as root when you use the passwd command, you will not need to enter the old password. If you are logged in as that user or you are a user other than root who has permission to change passwords for other users, you will need to enter the original password before you are prompted for the new one. If you make a mistake in typing the username, simply press Ctrl+C and you will be dropped back to a prompt. If you enter a new password, you will then need to retype the password for verification. After you finish, you should see this message:

passwd: all authentication tokens updated successfully

This means that the password change was successful, and you are ready to continue with your next task.

When you are adding a user, you might wish to use some of the flags listed here for customizing your user layout and structure:

**-c *comment*** This adds a comment to the password file comment field, which is where the user’s full name is stored.

**-d *directory*** The directory argument enables you to specify the user’s home directory.

The default is to append the user’s login to the default home directory setup on your server.

**-e *expire\_date*** The date must be specified in the format *YYYY-MM-DD* in order to cause a user’s account to be disabled on this date. This does not remove the user or the user’s files from the system.

**-f *inactive\_days*** Specifying a value here causes an account to be disabled *x* days after their password has expired. The default value is -1, which disables this feature.

**-g *initial\_group*** This specifies the initial group a user belongs to, in the GID field in the

/etc/passwd file.

**-G *group,[…]*** You can list as many groups as you wish here that the user should belong to other than his initial group.

**-m** This specifies that a directory will be created, if it does not exist, to serve as the user’s home directory. The default files for the user will be copied into this directory unless the

-k option is specified. If -k is given, then you can specify what directory you would like the default files to be copied from. The -k option is valid only when -m is used.

**-M** This option tells Linux not to create a home directory at all.

**-n** This flag turns off the UPG option, in effect not creating a default group for the user with their own name. This is beneficial to setups that will be assigning all users to users group.

**-o** This turns on the ability to create a new user with a duplicate UID.

**-p *password*** This specifies the password on the command line. Note that the user add function expects this to be in an encrypted form—*not* clear text.

**-r** This flag enables you to create a system account with a UID lower than 100.

**-s *shell*** This lets you specify the default shell that a user is assigned.

**-u *uid*** Pass in your desired UID for the user, and it shall be so.

These options provide you greater control over your user system. Mentioned in the -m flag is the directory of files used to set up a user’s home directory. By default, this is the /etc/skel directory. Placing a file in this directory or modifying a file’s contents will affect each user created from that point forward. You might wish to place an .htaccess file here if the account will have Web access, or perhaps a system rules and regulations document to make sure the user reads it. Creating your desired setup for new users in this directory will ensure that your modifications do not have undesired effects for users who might have already changed certain files in their home directories to modify their own environments.

To delete or remove a user, you can user the userdel command. This command takes one argument, the username, and removes a user from the system. Specifying the -r option removes that user’s home directory as well. If you wish to remove all the files and directories owned by that user, it’s a good idea to use the find / -user username –ls command to locate any files or directories owned by the user. Then you can remove whichever files you need and change the permissions and ownership for the rest.

**Group Administration**

If you were to retrieve a long listing of directory that contained

-wrxrw-r-- 1 joe jingle 36521 Jan 9 11:37 bebop you would see that the file is owned by the user joe. This does not necessarily mean that joe is in the group that owns the file: jingle. In fact, we see by the permissions in the first block that joe has execute permissions to the file, whereas users of the jingle group do not. When a user owns a file and is in the group assigned to the file as well, the user privileges will supersede the group privileges. For instance, if the user has only read permissions and the group has write and execute, the user will still have only read permissions. Make sure to keep these rules in mind as you set up your user/group system.

To create a new group, use the groupadd command. You can use the -g *gid* flag to specify a GID for the new group or you can leave it blank to select the next one available after 500.

Much like the delete user command, you can use groupdel to remove a group from the list.

To administer the /etc/group file with more than adding and deleting, you can use the gpasswd command. This command has several flags you can specify to set up your system. Here we have compiled a short list to help you out:

**-a *user group*** This assigns a user to the specified group.

**-d *user group*** This deletes a user from the targeted group.

**-R *group*** This flag disables access to a group.

**-r *group*** This option removes a group password.

**-M *user*,*[…] group*** This enables you to assign multiple users to a group.

**-A *user*,*[…] group*** This enables you to assign a single or multiple administrators to a group at once.

Passing in no arguments will enable you to define the password to a group. Simply passing in the single argument of the group name will prompt you to enter a password.

Any user on the system can attempt to use the newgrp command to add themselves to a group.

If this group has a password assigned to it, the user will be prompted to enter the correct password.

If the group does not have a password, the user will not be allowed to join the group.

**Modifying Users or Groups**

There are a few other commonly used administrative functions for changing users and groups.

We’ve listed them here as a resource for you to use. As for the other commands in this book, refer to the manual for each program for more information. Here are the commands:

**chfn** This command is used to change a user’s finger information; finger is a command that enables remote users to get basic information about users on your system.

**chsh** This lets you change the default shell for a user. Simply pass in the username and then the location of your target shell.

**groupmod** This enables you to modify a GID or name for a group.

**id** This echoes the GID or UID values for a user.

**newusers** If you need to create a large number of users at once, you can use this command.

This command accepts a text file as an argument that contains usernames and passwords in plain text. It will then parse the entire file and create the user accounts. See the manual for more information on the actual format.

**su** This command is used for any user on the system to change to another user. All users except root must enter the correct password for the user they wish to change to.

**Linux Distribution**

A Linux distribution (often abbreviated as distro) is an operating system made from a software collection, which is based upon the Linux kernel and, often, a package management system.

A typical Linux distribution comprises a Linux kernel, GNU tools and libraries, additional software, documentation, a window system.

Linux distributions optionally include some proprietary software that may not be available in source code form, such as binary blobs required for some device drivers.

A Linux distribution may also be described as a particular assortment of application and utility software (various GNU tools and libraries, for example), packaged together with the Linux kernel in such a way that its capabilities meet the needs of many users.

The software is usually adapted to the distribution and then packaged into software packages by the distribution's maintainers.

Two types of Distributions

1. RPM(Red-hat Package Manager) based (Fedora)

Red Hat Linux and SUSE Linux were the original major distributions that used the .rpm file format, which is today used in several package management systems. Both of these were later divided into commercial and community-supported distributions

1. Debian (.deb) Based (ubantu)

Debian is a distribution that emphasizes free software. It supports many hardware platforms.

Debian and distributions based on it use the .deb package format and the dpkg package manager and its frontends (such as apt-get or synaptic).

**Disk Partitions**

fdisk stands (for “fixed disk or format disk“) is an most commonly used command-line based

disk manipulation utility for a Linux/Unix systems. With the help of fdisk command you can

view, create, resize, delete, change, copy and move partitions on a hard drive using its own

user friendly text based menu driven interface.

View all Disk Partitions in Linux

fdisk -l

View Specific Disk Partition in Linux

# fdisk -l /dev/sda

Check all Available fdisk Commands

# fdisk /dev/sda

then type m on the screen, you will see the all available options for fdisk

that you can be used on the /dev/sda device.

Print all Partition Table in Linux

# fdisk /dev/sda

From the command mode, enter ‘p‘ instead of ‘m‘ as we did earlier.

As I enter ‘p‘, it will print the specific /dev/sda partition table.

How to Delete a Partition in Linux

f you would like to delete a specific partition (i.e /dev/sda9) from the specific hard disk such as /dev/sda. You must be in fdisk command mode to do this.

]# fdisk /dev/sda

Next, enter ‘d‘ to delete any given partition name from the system.

As I enter ‘d‘, it will prompt me to enter partition number that I want to delete from /dev/sda hard disk.

How to Create a New Partition in Linux

Bootloader is a piece of code that runs before any operating system is running.

Bootloader are used to boot other operating systems, usually each operating system has a set of bootloaders specific for it.

Bootloaders usually contain several ways to boot the OS kernel and also contain commands for debugging and/or modifying the kernel environment.

**Unit 2**

Apache Web Server is the most versatile web server on the Internet today. It has an endlesss supply of features that were drafted by the most experienced web server experts around theworld—features that include name-based and IP-based virtual hosting, user authentication, URL rewriting, Server Side Include (SSI), advanced logging, environment variable handlers, content negotiation, Common Gateway Interface (CGI) handlers, Secure Sockets Layer (SSL), and much more.

Apache maintains a presence of over 65 percent of all web servers on the Internet, according to Netcraft (www.netcraft.com). This growth has shown a steady trend over the past nine years.

Apache as it is known today was released on December 1, 1995. It was originally born from the Public Domain HTTPS Daemon developed by Rob McCool at the National Center for

Supercomputing Applications (NCSA), University of Illinois, Urbana-Champaign. McCool left the NCSA in mid-1994, and the project was temporarily stalled. During this time, many web developers had compiled extensions and bug fixes of their own that needed a common release to the public. Out of this necessity, a small group of webmasters collaborated to form a common information space and mailing list and began work on a new release. This became Apache 0.6.2 in April of 1995. At the same time, the NCSA also began further development on their own web server and were added as honorary members to the Apache project so that they could pool their ideas and code to further the realm of web serving.

The early Apache web server was a hit, but the code base required a massive overhaul and redesign. In the next few months, Rob Hartill and the rest of the development group began developing new features while Robert Thau designed a new server architecture that provided more stability and extensibility. This gave birth to Apache 0.8 in August 1995, and with further beta testing and ports24­2 to many platforms, we were given Apache 1.0 in December of that year.

In less than a year after the group was formed, Apache server took the lead as the number one web server and has stayed there ever since.

Now that you are aware of some of the features and the presence of Apache Web Server, let’s get started with learning more in-depth about the current versions and what you can do with the product.

# APACHE WEB SERVER AND ITS OVERALL WORKING IN DETAIL

Apache Web Server is an open source software that creates, deploys and manages the overall functioning of the web server. It was developed by a group of software programmers but now owned by the Apache Software Foundation. It currently holds over 50% of the global market share.

The job of web server is to accept the clients’ requests and serve with relevant responses in the form of web pages, if available, else return an error message (404!). In case of static requests, the web server gets a URL, takes out the requested filename from the local disk, and sends it over the internet wherever it is intended. In case of dynamic requests, the URL gets translated into a program name, which then gets executed so that the output is sent over the internet back to the intended location. Apache can serve both dynamic as well as static contents. The term ‘web server’ could be referred to the machine itself that receives requests and responds with the results based on the circumstances.

“***A web server is nothing but a computer device with a special software and functioning, and an internet connection so that other devices could get connected. This is how a valuable information transmission is possible.”***

Apache Web Server is free to download and install easily, an open source software which suits all the requirements. The typical requirement could be two to three pages in a website, or a detail oriented site with thousands of pages. A website also might need to handle millions of (regular) visitors in a month, once it is launched. It must [not crash](http://apachebooster.com/blog1/why-do-servers-crash-and-what-are-its-preventive-measures/), or even if it does initially, it must get fixed soon enough.

**How does Apache web server work?**

In an idle state, Apache server listens to various IP addresses that are being determined in its Config file. Whenever the requests are received, it quickly analyzes the headers and applies the protocols as specified in this file. This further helps take appropriate action, therefore making it a swift affair.

One web server can handle countless websites depending on its memory capacity and other crucial factors. To the outside world, it appears as if all the websites are independent, despite of the fact that all are going to eventually mapped to the same machine. The concept of *virtual hosts* comes into the picture where the sites are termed as virtually different.

“***The method of hosting a wide range of domain names under the name of a single server or a pool of servers is what we call Virtual Hosting.”***

The most famous examples of web hosting companies are – GoDaddy, HostGator, DreamHost, InMotion, Bluehost, 1&1, and A2 Web Hosting. Any firm, whether it is a start-up or an already established company, has a website and the domain is purchased online at one of the mentioned[web hosting companies](http://www.ndimensionz.com/kb/how-to-select-a-new-web-host/). The web hosting company has several packages at different prices. The buyers decide the best suitable ones, based on their current status of the business.

**What is Apache HTTP server?**

It is a list of codes (or a software program, to be specific), under relevant operating system, that runs in the background. It supports multi-tasking and serves other applications connected to it. Initially, it was designed for Linux and Unix operating system only but later its design was adapted to work under Macintosh and Windows as well.

Installing Apache on Linux operating system requires good programming skills but on Windows it is easy and straight forward because of the presence of GUI (Graphical User Interface). The source code of the Apache web server is very much basic, and therefore needs to have added functionality through many modules, being written by software programmers. This is how a server has extended functionalities. Appropriate modules are written based on the requirement.

In order to add a new module, just install that particular module and restart the Apache web server. If you don’t want a particular functionality, simply remove it so that the server becomes light and faster. It consumes lesser system resources and hence safeguards the server from security holes. Apache server also gets easily integrated with other third party open source applications, like MySQL and PHP. This makes it even more powerful.

**What goes on in the background while client and server communicate?**

Most of us who deal with the internet surfing, websites and the world wide web are familiar with the Internet Protocol, aka the IP address. Every computer, laptop, mobile phone, and tablet has a unique IP address so that a successful information flow could be maintained. The information cannot be sent to any device in absence of an IP address, just like a letter cannot be sent to a house without an official postal address. It is responsible for determining the geographical location of any computer device residing in this world.

It is practically impossible to remember each technical IP address of every device. The solution given to the issue is – giving a separate domain name to each site. It is the job of DNS server to translate the user-friendly domain names to their original IP addresses (or vice versa) and serve it to the clients.

For example, you typed [www.google.com](http://www.google.com/) and you obviously do not know its original IP address. The DNS server helps with the translation part so that the server ‘understands’ what the client is actually asking for. It will take out with the relevant site that has an IP address, say 216.58.216.164, which will display the Google’s page in front of you. Please note that Google has several IP addresses, depending on which server is currently serving you. Your purpose is served without worrying about any hassle of background working!

If the client types [www.google.com](http://www.google.com/) at the browser’s address field, following request appears to the server:

***GET / HTTP/1.1***

***Host: www.google.com***

The GET method specifies which page (or program) needs to be retrieved. The forward slash ‘/’ denotes the root directory, and HTTP 1.1 denotes the HTTP version. HTTP is a stateless protocol that governs the communication between the client (requests) and the server (responses). If the request is successful, i.e. when the page is retrieved and is ready for display, following code would be executed in the background:

***HTTP/1.1 200 OK***

***Date: Sun, 10 Jun 2017 19:19:19 GMT***

***Server: Apache***

***Expires: Wed, 15 Jan 1987 15:00:00 GMT***

***Cache-Control: no-cache, must-revalidate, max-age=0***

***Pragma: no-cache***

***Last-Modified: Sun, 10 Jun 2017 19:19:19 GMT***

***Vary: Accept-Encoding,User-Agent***

***Content-Type: text/html; charset=UTF-8***

***Content-Length: 7560***

The status 200 determines that the requested page is successfully found and is ready for display. The header Content-Type lets the client browser know about the type of data so that it knows in advance the ways to handle it. Content-Length lets the client browser know the length of the response page body. Following code is executed in case the page is unable to get displayed:

***HTTP/1.1 404 Not Found***

The exact message is displayed depending on what string characters the programmer has written, and in what exact page-design. There are statuses other than just 200 and 404, and so depending on the situation certain status show up as per relevance. The information written here is just a tip of an iceberg. The programmers who actually do the coding part for servers are familiar with every crucial detail. To learn more about this, one must show enthusiasm towards websites’ programming.

**Secure Web server**

Running a secure Web server is something you should consider if the daily use of your websites will include an exchange of confidential, private information from your users. Regular Web servers send and receive traffic in unencrypted form. Unfortunately, this makes them vulnerable to man-in-the-middle attacks, where a potential attacker could use sniffer tools to log packets en route from clients to the server and derive sensitive information from them.

This mode of security is completely unacceptable for websites that must deal in personal data, like bank accounts, medical or financial records, or others.

The secure Web server eliminates this threat by offering two key advantages:

*•* It allows users to verify the identity of the server.

*•* It allows users to conduct safe transactions with your server by encrypting the authentication and the session.

**To achieve this, the Apache Web server uses secure communication protocols like the Secure Socket Layer (SSL) or the Transport Layer Security (TLS) to protect the flow of data.**

**Encrypted session**

Before we setup a secure server, we should first understand how encrypted communication between the server and the client is conducted.

Let us outline the details of a typical secure session:

*•* A client tries to connect to port 443 on the secure Web server.

*•* The client sends a list of available encryption methods it supports; if the client cannot support encryption, for instance very old browsers, the connection attempt will be unsuccessful. Modern browsers support both SSL and TLS without any problems.

*•* The server will choose the strongest available encryption method that both sides can support.

*•* The server will then send back to the client its certificate and the public encryption key. The certificate is a sort of an ID, telling the client important information about the server. To make this information credible, the certificate must be signed by a reputable Certificate Authority

(CA), like EquiFax, Thawte or others. The public key will be used by the client to generate its own encryption hash should it choose to accept the server’s certificate.

*•* The client receives the certificate. In most browsers, the certificate is first compared to an existing list of authorities. If the digital signature matches, the certificate will be accepted. If no match is found for the certificate, the browser might use the Online Certificate Status Protocol (OCSP) to connect to CAs in real time in an attempt to verify the certificate. Generally, the use of OCSP is not enabled by default in most browsers, in order to speed up the authentication process. If no match is found still, the client will be issued a warning by the browser, informing it that the certificate could not be verified. The user now must decide whether he/she can take the risk and accept the certificate.

In addition to being self-signed (i.e. no CA signature), the typical issues arising with certificate prompts include a mismatch between the site you are trying to access and the one registered in the certificate, dubious credentials or an expired certificate.

*•* Regardless of what may occur, if the client accepts the connection, it will send back a hash encrypted with the server’s public key. This hash will be used to encrypt all communication between the server and the client throughout the session. Only the client will be able to decrypt the communications - or rather, anyone who possesses the private key.

But if the client side is fairly secure and the server’s certificate is valid, the communication is safe.

**Understanding the *httpd.conf* File**

Apache, like many other Linux applications, has configuration files in with runtime settings that can be configured and controlled by the system administrators. Apache uses the file named httpd.conf, which is located in the conf/ directory where you installed Apache. Let’s take a look at the important directives of this file now:

**ServerType standalone** By using the *standalone* directive, the ServerType directive enables your server to run as a daemon, as a service using *inetd*, or as a *standalone* server. For the purposes of this book, you will run the server as standalone.

**ServerRoot "/www"** The ServerRoot directive defines where Apache is installed. If you are running Apache as a local server on that particular machine, this directive should not be modified.

If you wish to run Apache through the network while it is installed on another machine, you would modify this path for the network path relative to the server handling the public HTTP requests.

**PidFile /www/logs/httpd.pid** A PID file is the file responsible for reporting the process ID to the system. This should not be modified from the default configuration.

**ScoreBoardFile /www/logs/httpd.scoreboard** The ScoreboardFile is used to store internal server process information and should not be modified. It might not be needed by your system, but just in case, leave this directive alone.

**Timeout 300** The Timeout directive prevents a hung request from never ending. If the request exceeds this setting, it will die and free the process for other connections.

**KeepAlive On** The KeepAlive directive enables the processes to stay alive by allowing a client to make more than one request during a network session instead of opening and closing them each time a process is handled. Leaving this setting to On will reduce the cycles required to operate your web server by allowing multiple uses per thread. This directive is used for one-to-one connections.

**MaxKeepAliveRequests 100** This directive defines the maximum number of requests to allow during a persistent connection. If you want to allow unlimited requests, set this to 0; however, you could make your system vulnerable. You should modify this number only if you are having connection errors during peak hours.

**KeepAliveTimeout 15** The KeepAliveTimeout directive defines the number of seconds to wait for a particular user’s request before it frees the process to be used by another user. The default number is 15 and should not be modified unless you fully understand what will happen to your server by doing so.

**MinSpareServers 5** This directive indicates the number of minimum spare servers to spawn as a backup to catch any traffic bursts to your server. A good starting point for a standard server is 5. You might want to evaluate this as your traffic increases.

**MaxSpareServers 10** This directive indicates the *maximum* number of spare servers to spawn as a backup to catch any traffic bursts to your server. A good starting point for a standard server is 10. Once again, you might want to evaluate this as your traffic increases. Also, you should note that this number should be larger than the MinSpareServers directive.

**StartServers 5** This directive indicates the number of servers to start initially. As traffic starts to utilize this number of servers, the MinSpareServers and MaxSpareservers directives will come into effect.

**MaxClients 150** The MaxClients directive is one you should analyze very carefully. This directive will prevent your server from becoming too busy; however, if you have the hardware

Understanding the *httpd.conf* File to handle more, we recommend setting this to a higher number, depending on the traffic load of your website. If more than 150 users try to connect, they will be locked out of the site until users start to time out.

**NOTE** We have not had to modify this number, even with 30,000 visitors per day on LinuxForum.com. You might want to consider increasing this number if your server has a high traffic load.

**MaxRequestsPerChild 0** This directive defines the number of requests each child process is allowed to process before it dies. The default setting is 0, which means unlimited. If you are having performance issues, you might want to consider assigning a value to this setting.

**Listen 12.34.56.78:80** By default, Apache will listen on all IP addresses assigned to a machine. Therefore, this directive is commented out and not used by default. If you want to limit which IP addresses can be accessed via the Internet, you can assign them here. One IP address is allowed per line, for example:

Listen 12.34.56.78:80

Listen 12.34.56.79:80

**BindAddress \*** This directive is also disabled by default. It is used to support virtual hosts and addresses. If you use the Listen directive, you should also use this directive.

**LoadModule *name file*** The LoadModule directive loads the module into Apache. If you performed the DSO installation method, you will see all of the modules that you entered into your configuration options here. This section should be coordinated with the following AddModule section.

**AddModule *header file name*** This directive should be coordinated with the previous

LoadModule section and it defines the C header files of the modules.

**Port 80** By default, Apache is designed to listen on port 80, which is the standard web server port. If you do not want to listen on port 80, define your port here.

**NOTE** If you change the port number, make sure to update your firewall settings.

**NOTE** If you change this port number, you must define what port you are using in your URL request—for example: <http://yourdomain.com:8080>.

You might also notice the following code for your SSL installation:

<IfDefine SSL>

Listen 80

Listen 443

</IfDefine>

This code will be executed if you start Apache with SSL:

apachectl startssl

**User nobody** Apache will run as a user and group defined in the configuration file. For security purposes, Apache is set to nobody or apache because these users do not have any permission to execute commands or to write or delete files that that they do not explicitly own.

**NOTE** Do not be fooled by this setting. If you chmod a file or directory to 777, or chown a directory to the same user and group the server runs as, the web server will have ability to write to it and potentially cause harm to your system.

**Group nobody** This directive is the same as the User directive except it defines the group that the server runs as.

**ServerAdmin *root@yourdomain.com*** The ServerAdmin directive is simply a contact e-mail address for the administrator of the server. It is not required, and we would hesitate to put a legitimate or an important e-mail address in this setting.

**ServerName *www.yourdomain.com*** By default, Apache will deliver any web pages located in the default DocumentRoot directive to any name that resolves to your IP address. For example, if you have yourdomain.com and mydomain.com pointed to your IP address, a web page from the Document Root will be displayed.

**TIP** If you get a message that says [alert] httpd: Could not determine the server's fully qualified domain name, using 127.0.0.1 for ServerName when you start your Apache Web Server, uncomment this line and enter your hostname. Additionally, you should put the fully qualified domain name in the /etc/hosts file.

**DocumentRoot "/www/htdocs"** This is the directory where your website files are located.

Try to remember this term because you will see it in many locations, including PHP.

**Directory** The Directory directive enables you to specify certain options for a directory.

A Directory directive has an opening and closing tag. The directory you are configuring is defined in the opening tag. See the following example:

<Directory /path/to/directory>

Options FollowSymLinks

AllowOverride None

</Directory>

**UserDir** If you enabled mod\_user, you can allow system users to have their own websites on your server. By default, these directives are allowed to be utilized only if the mod\_user module is installed by using the IfModule settings:

<IfModule mod\_userdir.c>

UserDir public\_html

</IfModule>

An example of a system user website is as follows:

http://www.yourdomain.com/~username

The user’s website files will be located in their home directory under a subdirectory called public\_html.

**DirectoryIndex *files*** The DirectoryIndex directive defines which page to be displayed by default. For example, currently it is defined to load index.html if one exists.

This is one of the directives that we feel are commonly configured improperly by default in the Apache configuration. Every time we have set up an Apache Web Server, we have had to modify this setting. We add index.htm, default.html, default.htm, default.php, and index.php to each of our web server configurations.

<IfModule mod\_dir.c>

DirectoryIndex index.html

</IfModule>

**AccessFileName .htaccess** The AccessFileName is the name of a file that is placed inside a directory and enables you to control the directives within that directory on the web server.

These files can have the same directives that the Directory directive can have.

Worth noting are the default settings in the httpd.conf file that prevent web clients from accessing these file types:

<Files ~ "^\.ht">

Order allow,deny

Deny from all

Satisfy All

</Files>

**NOTE** It is important that the AccessFileName file begin with a period (.) as long as you leave the default settings listed in the preceding code for the Files directive. If you alter the Files directive listed and you use .ht prefixes, these sensitive files can become accessible via the web browser, and potential hackers might see what types of settings you use.

**UseCanonicalName On** This setting is important to the operation of CGI- and PHP-style scripts that use SERVER\_NAME environment variables. It handles the way self-referencing URLs are returned from the server. When this directive is turned on, it will display the ServerName directive and port in the form of a canonical name.

**TypesConfig** The TypesConfig directive defines the file that is responsible for translating MIME types for files. This directive, by default, falls into the IfModule setting.

<IfModule mod\_mime.c>

TypesConfig /www/conf/mime.types

</IfModule>

**DefaultType text/plain** DefaultType defines the MIME type to send a file as if the file is not defined in the mime.types file defined by TypesConfig.

**MIMEMagicFile /www/conf/magic** MIMEMagicFile is a module that enables certain hints from a file’s contents to determine which type of file it is; therefore it is known as *magic*. This directive also falls into an IfModule setting.

<IfModule mod\_mime\_magic.c>

MIMEMagicFile /www/conf/magic

</IfModule>

**HostnameLookups Off** This directive enables you to perform hostname lookups on each IP address that connects to your Apache Web Server. We recommend you leave this setting disabled and use the logresolve utility provided by the Apache developers to do hostname lookups after the log file has been rotated.

**ErrorLog /www/logs/error\_log** The ErrorLog directive defines a file to log errors for the applicable server or virtual host.

**LogLevel warn** The LogLevel directive defines the minimum level of an error to log. The default setting, warn, works well.

**LogFormat** LogFormat enables you to format your log files to your desire. The standard log format produces the following output:

127.0.0.1 - frank [17/Mar/2004:09:30:25 -0500] "GET /index.php

HTTP/1.0" 200 2326

**CustomLog /www/logs/access\_log common** CustomLog is an alternative to TransferLog and the other log file types. We prefer to append the combined format instead of common format to the end of the log file name. The combined log file is more verbose and has more information:

CustomLog /www/logs/access\_log combined

**ServerSignature On** The ServerSignature setting enables the server type, version, and possibly some of the configuration options to be displayed when requested by an error document or an outside source. We feel that displaying this information might be a security risk so we turn it off.

**Alias /icons/ "/www/icons/"** An Alias directive enables you to store files and directories outside of the DocumentRoot and then alias them into a website.

**ScriptAlias /cgi-bin/ "/www/cgi-bin/"** ScriptAlias is much like Alias except the directories defined here are known to be executable, for example, cgi-bin.

**IndexOptions FancyIndexing** IndexOptions enables you to display a directory’s contents as icons—if the files listed in the DirectoryIndex directive are not available. Having this setting enabled by default is a bad idea and using *FancyIndexing* should be explicitly defined in a Directory option.

**NameVirtualHost \*** NameVirtualHost is a directive for allowing name-based virtual hosts.

We enable this setting with a simple \* as the value. We will discuss this later in the “Configuring

Apache Virtual Hosts” section.

**VirtualHost** Virtual hosts enable you to run multiple websites on a single machine and a single IP address or multiple IP addresses. We will discuss this later in the “Configuring

Apache Virtual Hosts” section.

**Using Apache Virtual Hosts**

Apache has an excellent built-in feature called virtual hosts. This feature enables you to host multiple domain names from the same server and either use a single IP address for all of the virtual hosts or assign a group of hosts to an IP address while assigning another group to a different

IP address and so on. This widely flexible feature is well designed and very configurable.

In fact, virtual hosts enable web-hosting companies to pack many clients onto a single machine with little overhead.

Apache’s virtual hosting features are quite extensive; in fact, there are too many to list in this chapter. Here is a list of the most important features you might find uses for in this book:

● IP or name-based virtual hosts

● Each host can have their own DocumentRoot

● Separate logging facilities for each host

● Ability to share directories between virtual hosts by using ScriptAlias and Alias directives

● Ability to allow override options per directory or virtual host Using Apache Virtual Hosts

When using SSL-enabled virtual hosts, you must take some things into consideration. First, you must have a separate IP address or SSL port per IP address for each unique SSL virtual host. If you decide to use different ports, the port must be specified in the URL—for example, https://www.yoursite.com:444 if you are using port 444 for an alternate. Second, you can use a single IP address with one single SSL virtual host and multiple non-SSL virtual hosts. SSL virtual hosts are beyond the scope of this book, so if you want to learn more about them, check the httpd.conf examples or read the mod\_ssl documentation.

**Directives Supported by Virtual Hosts**

We have already mentioned that Apache virtual hosts are widely configurable; therefore, in this section we will provide a list of directives that can be used in virtual hosts.

Here is the list of documented directives available in Apache virtual hosts:

● AccessConfig

● AccessFileName

● ContentDigest

● DefaultType

● <Directory>

● <DirectoryMatch>

● DocumentRoot

● EBCDICConvert

● EBCDICConvertByType

● EBCDICKludge

● ErrorDocument

● ErrorLog

● FileETag

● <Files>

● <FilesMatch>

● <Group> (Requires suEXEC installed)

● HostnameLookups

● IdentityCheck

● LimitInternalRecursion

● LimitRequestBody

● <Location>

● <LocationMatch>

● LogLevel

● Options

● ResourceConfig

● RLimitCPU

● RLimitMEM

● RLimitNPROC

● ServerAdmin

● ServerAlias

● ServerName

● ServerPath

● ServerSignature

● UseCanonicalName

● User (Requires suEXEC installed)

● Let’s put some of these directives to use in configuring an Apache virtual host.

**Configuring Apache Virtual Hosts**

It is now time to start planning and configuring your virtual hosts. By the end of this section, you will be very familiar with how well designed and flexible the Apache Web Server really is.

**Got DNS?**

For this example, we are going to assume you have two domain names with DNS set up to point to the IP address of your server. If you do not have DNS set up or you do not own domain names yet, you can simply edit your system’s hosts file and trick your system into resolving a fake name for your server’s IP address.

On Linux, the HOSTS file is located in /etc/hosts, and you will need to add the following lines to it, assuming your server’s IP address is 123.456.789.1.

**NOTE** The IP addresses used here are fictional will not work on a real server. You *must* substitute

*123.456.789.1* with your real IP addresses!

123.456.789.1 yourdomain.com www.yourdomain.com

123.456.789.1 yourotherdomain.com www.yourotherdomain.com

On Windows systems, you should search your C:\Windows directory for a file named hosts.

It is usually located in C:\Windows\System32\Drivers\etc\hosts. In this file, you will add the following lines:

123.456.789.1 yourdomain.com www.yourdomain.com

123.456.789.1 yourotherdomain.com www.yourotherdomain.com

Now your system is tricked into resolving the domain names yourdomain.com, www.yourdomain

.com, yourotherdomain.com, and www.yourotherdomain.com to the IP address of the server.

**TIP** Keep this previous trick in mind the next time you register a domain name. You can set up your HOSTS files so you can start building your website while you wait for DNS to register and resolve.

**Preparation**

Before you start digging into the configuration files, you need to prepare the system by creating the DocumentRoot and Logging directories. A rule of thumb that usually works well is to use the domain name for the parent directory. You can store your DocumentRoot directories anywhere you would like. We prefer to use /home. Follow these steps:

**1.** Create the directories:

mkdir -p /home/www.yourdomain.com/public\_html/cgi-bin

mkdir -p /home/www.yourdomain.com/logs

mkdir -p /home/www.yourotherdomain.com/public\_html/cgi-bin

mkdir -p /home/www.yourotherdomain.com/logs

**NOTE** The traditional method of using cgi-bin directories is to place them outside of your document root and use the cgi-bin ScriptAlias and Alias directives. However, this method works and can be used if you understand that there might be minimal risks involved.

**2.** Now you have your directories created, you should change the permissions of these directories because you are probably running as root. It is good practice to assign the files to a user other than root. This prevents your users from logging in to the server as root to edit files and manage websites. Change the permissions as follows:

chown -R *someuser.somegroup* /*home*/*www.yourdomain.com*/

chown -R *someuser.somegroup* /*home*/*www.yourotherdomain.com*/

**3.** You need to create a directory that will store your virtual host files. You have the ability to simply append the virtual host configurations directly to the httpd.conf file; however, you can also have Apache include a directory of configuration files when it starts up. Keeping a separate directory of individual virtual host files seems more practical than editing a configuration file of 500+ lines each time you need to manage one of those hosts. Make the directory now: mkdir /www/conf/vhosts

**4.** Modify the httpd.conf file to include this new directory of configuration files you will be creating; open the /www/conf/httpd.conf file and add the following line to the end of it: include conf/vhosts

**5.** Because you are editing the httpd.conf file, you need to enable one more setting for your virtual hosts. Locate the NameVirtualHost directive line and remove the comment symbol (#); then change it to the following setting:

NameVirtualHost \*

The previous setting enables virtual hosts to be configured for any IP address.

You should be all set to create your virtual host configurations now.

**Virtual Host Configuration Files**

In the previous section, we mentioned that creating separate files for each host makes virtual hosts much easier to manage. Let’s set up the first file for www.yourdomain.com and name this file www.yourdomain.com.conf in the /www/conf/vhosts directory. The file will contain the following contents:

**<VirtualHost \*>**

**ServerName www.yourdomain.com**

**ServerAlias yourdomain.com**

**DocumentRoot /home/www.yourdomain.com/public\_html**

**CustomLog /home/www.yourdomain.com/logs/access\_log combined**

**ErrorLog /home/www.yourdomain.com/logs/error\_log**

**</VirtualHost>**

**NOTE** When using the backslash to continue directives, you should avoid allowing any additional spaces or characters after the backslash (\), which could cause parse errors in the configuration files.

Let’s break down this configuration file for better understanding. The following line is the opening tag for the virtual host. Everything between this line and the last line, </VirtualHost>, will contain the settings for this virtual host that you are configuring. Take special notice of the setting \* in the open tag. The \* indicates that this virtual host will respond to the request for this domain name on any IP address it is used for:

<VirtualHost \*>

The next directive is the name of your server, hence the name ServerName:

ServerName www.yourdomain.com

Next, you define the ServerAlias directive. This enables you to point yourdomain.com to the same virtual host. This is important because web users have a tendency to access your site without the leading www, and you do not want to lose any users by an improperly configured server:

ServerAlias yourdomain.com

The following DocumentRoot directive is the path to the directory where the website’s files are located:

DocumentRoot /home/www.yourdomain.com/public\_html

The following CustomLog directive is the path to the log file that will contain the access information about the virtual host. In the “Understanding the httpd.conf File” section of this chapter, we discussed the options and settings for this directive.

CustomLog /home/www.yourdomain.com/logs/access\_log \combined

The ErrorLog directive was also discussed in the “Understanding the httpd.conf File” section in this chapter and it contains error information regarding the domain:

ErrorLog /home/www.yourdomain.com/logs/error\_log

Finally, you close the Virtual Host configuration for this domain name by using the closing tag:

</VirtualHost>

Now that you have a good understanding of the virtual host configuration, let’s make another virtual host for your other domain name. Create a file located at /www/conf/www.yourotherdomain

.com.conf and enter the following information:

<VirtualHost \*>

ServerName www.yourotherdomain.com

ServerAlias yourotherdomain.com

DocumentRoot /home/www.yourotherdomain.com/public\_html

CustomLog /home/www.yourotherdomain.com/logs/access\_log\combined

ErrorLog /home/www.yourotherdomain.com/logs/error\_log

</VirtualHost>

The next task you will perform is to create an index.html file that will display a message indicating which virtual host is being displayed when you access the domain name.

In your /home/www.yourdomain.com/public\_html directory, create a file named index.html with the following contents:

<html>

<head><title>YourDomain.com</title></head>

<body>Welcome to YourDomain.com</body>

</html>

In your /home/www.yourotherdomain.com/public\_html directory, create a file named index.html with the following contents:

<html>

<head><title>YourOtherDomain.com</title></head>

<body>Welcome to YourOtherDomain.com</body>

</html>

Before you go any further, read the next section, “A Lesson in Testing Configuration File Changes.” After you have completed those steps, access your domain names configured in this section via your web browser and you should see the appropriate files for each virtual host.

**A Lesson in Testing Configuration File Changes**

Because adding your virtual hosts has been the first real editing you have done with your configuration files, you need to go through a small routine that prevents you from taking your web server offline.

Because Apache will not read configuration files on the fly, any changes you make to the configuration files will require the server to be restarted to take effect. The kind developers of

Apache have taken measures to prevent you from taking your server offline in the event you

“fat-fingered” your way through the configuration file.

You might have noticed in the “Becoming Familiar with Apache Programs” section of this chapter that we gave you a list of commands to run. The command for apachectl contains a special setting that enables you to test your configuration files while the server is still running and prevent you from taking your server offline.

Let’s issue the apachectl configtest command now and check the output:

**Good syntax output:**

Processing config directory: /www/conf/vhosts/

Processing config file: /www/conf/vhosts/www.yourdomain.com.conf

Syntax OK

In this example, everything parsed as expected by the server and you are clear to start, stop, or restart your server as needed to make the changes take effect.

**Bad syntax output:**

Processing config directory: /www/conf/vhosts/

Processing config file: /www/conf/vhosts/www.yourdomain.com.conf

Syntax error on line 2 of /www/conf/vhosts/www.yourdomain.com.conf:

Invalid command 'Oops', perhaps mis-spelled or defined by a module not included in the server configuration

In this example, you have a bad token on line 2 of your www.yourdomain.com.conf file. (We purposely entered the text *Oops* because we knew that it would cause a syntax error in Apache.)

After you get the Syntax OK output from your configuration files, you are clear to restart the server. We prefer to completely stop the server and then start it instead of using the restart command because we have had experiences in the past where the changes would not take effect by using the restart command.

apachectl stop

apachectl start

Alternatively, if you are using SSL:

apachectl stop

apachectl startssl

Now is a good time for you to check your website and ensure the changes have taken effect.

**Apache Log File**

In order to effectively manage a web server, it is necessary to get feedback about the activity and performance of the server as well as any problems that may be occurring. The Apache HTTP Server provides very comprehensive and flexible logging capabilities.

The Apache HTTP Server provides a variety of different mechanisms for logging everything that happens on your server, from the initial request, through the URL mapping process, to the final resolution of the connection, including any errors that may have occurred in the process. In addition to this, third-party modules may provide logging capabilities, or inject entries into the existing log files, and applications such as CGI programs, or PHP scripts, or other handlers, may send messages to the server error log.

**Access Log File**

The server access log records all requests processed by the server. The location and content of the access log are controlled by the [CustomLog](https://httpd.apache.org/docs/1.3/mod/mod_log_config.html#customlog) directive. The [LogFormat](https://httpd.apache.org/docs/1.3/mod/mod_log_config.html#logformat) directive can be used to simplify the selection of the contents of the logs. This section describes how to configure the server to record information in the access log.

Of course, storing the information in the access log is only the start of log management. The next step is to analyze this information to produce useful statistics. Log analysis in general is beyond the scope of this document, and not really part of the job of the web server itself. For more information about this topic, and for applications which perform log analysis, check the [Open Directory](http://dmoz.org/Computers/Software/Internet/Site_Management/Log_Analysis/) or [Yahoo](http://dir.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/Servers/Log_Analysis_Tools/).

Various versions of Apache httpd have used other modules and directives to control access logging, including mod\_log\_referer, mod\_log\_agent, and the TransferLog directive. The CustomLog directive now subsumes the functionality of all the older directives.

The format of the access log is highly configurable. The format is specified using a [format string](https://httpd.apache.org/docs/1.3/mod/mod_log_config.html#formats) that looks much like a C-style printf(1) format string. Some examples are presented in the next sections. For a complete list of the possible contents of the format string, see the [mod\_log\_config documentation](https://httpd.apache.org/docs/1.3/mod/mod_log_config.html).

### ***Common Log Format***

A typical configuration for the access log might look as follows.

LogFormat "%h %l %u %t \"%r\" %>s %b" common

CustomLog logs/access\_log common

This defines the nickname common and associates it with a particular log format string. The format string consists of percent directives, each of which tell the server to log a particular piece of information. Literal characters may also be placed in the format string and will be copied directly into the log output. The quote character (") must be escaped by placing a back-slash before it to prevent it from being interpreted as the end of the format string. The format string may also contain the special control characters "\n" for new-line and "\t" for tab.

The CustomLog directive sets up a new log file using the defined nickname. The filename for the access log is relative to the [ServerRoot](https://httpd.apache.org/docs/1.3/mod/core.html#serverroot) unless it begins with a slash.

The above configuration will write log entries in a format known as the Common Log Format (CLF). This standard format can be produced by many different web servers and read by many log analysis programs. The log file entries produced in CLF will look something like this:

127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache\_pb.gif HTTP/1.0" 200 2326

Each part of this log entry is described below.

127.0.0.1 (%h)

This is the IP address of the client (remote host) which made the request to the server. If [HostnameLookups](https://httpd.apache.org/docs/1.3/mod/core.html#hostnamelookups) is set to On, then the server will try to determine the hostname and log it in place of the IP address. However, this configuration is not recommended since it can significantly slow the server. Instead, it is best to use a log post-processor such as [logresolve](https://httpd.apache.org/docs/1.3/programs/logresolve.html) to determine the hostnames. The IP address reported here is not necessarily the address of the machine at which the user is sitting. If a proxy server exists between the user and the server, this address will be the address of the proxy, rather than the originating machine.

- (%l)

The "hyphen" in the output indicates that the requested piece of information is not available. In this case, the information that is not available is the RFC 1413 identity of the client determined by identd on the clients machine. This information is highly unreliable and should almost never be used except on tightly controlled internal networks. Apache httpd will not even attempt to determine this information unless [IdentityCheck](https://httpd.apache.org/docs/1.3/mod/core.html#identitycheck) is set to On.

frank (%u)

This is the userid of the person requesting the document as determined by HTTP authentication. The same value is typically provided to CGI scripts in the REMOTE\_USER environment variable. If the status code for the request (see below) is 401, then this value should not be trusted because the user is not yet authenticated. If the document is not password protected, this entry will be "-" just like the previous one.

[10/Oct/2000:13:55:36 -0700] (%t)

The time that the server finished processing the request. The format is:

[day/month/year:hour:minute:second zone]  
day = 2\*digit  
month = 3\*letter  
year = 4\*digit  
hour = 2\*digit  
minute = 2\*digit  
second = 2\*digit  
zone = (`+' | `-') 4\*digit

It is possible to have the time displayed in another format by specifying %{format}t in the log format string, where format is as in strftime(3) from the C standard library.

"GET /apache\_pb.gif HTTP/1.0" (\"%r\")

The request line from the client is given in double quotes. The request line contains a great deal of useful information. First, the method used by the client is GET. Second, the client requested the resource /apache\_pb.gif, and third, the client used the protocol HTTP/1.0. It is also possible to log one or more parts of the request line independently. For example, the format string "%m %U%q %H" will log the method, path, query-string, and protocol, resulting in exactly the same output as "%r".

200 (%>s)

This is the status code that the server sends back to the client. This information is very valuable, because it reveals whether the request resulted in a successful response (codes beginning in 2), a redirection (codes beginning in 3), an error caused by the client (codes beginning in 4), or an error in the server (codes beginning in 5). The full list of possible status codes can be found in the [HTTP specification](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) (RFC2616 section 10).

2326 (%b)

The last entry indicates the size of the object returned to the client, not including the response headers. If no content was returned to the client, this value will be "-". To log "0" for no content, use %B instead.

#### **Combined Log Format**

Another commonly used format string is called the Combined Log Format. It can be used as follows.

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" combined  
CustomLog log/acces\_log combined

This format is exactly the same as the Common Log Format, with the addition of two more fields. Each of the additional fields uses the percent-directive %{header}i, where header can be any HTTP request header. The access log under this format will look like:

127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache\_pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"

The additional fields are:

"http://www.example.com/start.html" (\"%{Referer}i\")

The "Referer" (sic) HTTP request header. This gives the site that the client reports having been referred from. (This should be the page that links to or includes /apache\_pb.gif).

"Mozilla/4.08 [en] (Win98; I ;Nav)" (\"%{User-agent}i\")

The User-Agent HTTP request header. This is the identifying information that the client browser reports about itself.

**Error Log File**

The server error log, whose name and location is set by the [ErrorLog](https://httpd.apache.org/docs/1.3/mod/core.html#errorlog) directive, is the most important log file. This is the place where Apache httpd will send diagnostic information and record any errors that it encounters in processing requests. It is the first place to look when a problem occurs with starting the server or with the operation of the server, since it will often contain details of what went wrong and how to fix it.

The error log is usually written to a file (typically error\_log on unix systems and error.log on Windows and OS/2). On unix systems it is also possible to have the server send errors to syslog or [pipe them to a program](https://httpd.apache.org/docs/1.3/logs.html#piped).

The format of the error log is relatively free-form and descriptive. But there is certain information that is contained in most error log entries. For example, here is a typical message.

[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client denied by server configuration: /export/home/live/ap/htdocs/test

The first item in the log entry is the date and time of the message. The second entry lists the severity of the error being reported.

[***Log Level***](https://httpd.apache.org/docs/1.3/mod/core.html#loglevel)***directive***

The [Log Level](https://httpd.apache.org/docs/1.3/mod/core.html#loglevel) directive is used to control the types of errors that are sent to the error log by restricting the severity level. The third entry gives the IP address of the client that generated the error. Beyond that is the message itself, which in this case indicates that the server has been configured to deny the client access. The server reports the file-system path (as opposed to the web path) of the requested document.

A very wide variety of different messages can appear in the error log. Most look similar to the example above. The error log will also contain debugging output from CGI scripts. Any information written to stderr by a CGI script will be copied directly to the error log.

It is not possible to customize the error log by adding or removing information. However, error log entries dealing with particular requests have corresponding entries in the [access log](https://httpd.apache.org/docs/1.3/logs.html#accesslog). For example, the above example entry corresponds to an access log entry with status code 403. Since it is possible to customize the access log, you can obtain more information about error conditions using that log file.

During testing, it is often useful to continuously monitor the error log for any problems. On unix systems, you can accomplish this using:

tail -f error\_log

This specifies [log message severity.](http://httpd.apache.org/docs/current/mod/core.html#loglevel) Default is “warn.”

[quote]LogLevel warn[/quote]

Table of Level Severities-

| Severity | Description | Example |
| --- | --- | --- |
| emerg | Emergencies — system is unusable | “Child cannot open lock file. Exiting” |
| alert | Immediate action required | “getpwuid: couldn’t determine user name from uid” |
| crit | Critical conditions | “socket: Failed to get a socket, exiting child” |
| error | Error conditions | “Premature end of script headers” |
| warn | Warning conditions | “child process 1234 did not exit, sending another SIGHUP” |
| notice | Normal but significant condition | “httpd: caught SIGBUS, attempting to dump core in …” |
| info | Informational | “Server seems busy…” |
| debug | Debug-level messages | “opening config file …” |
| trace1-8 | Trace messages | “proxy: FTP: … ” |

Note- regarding a particular level: Levels are listed in order of descending severity. When triggered, a configured log level will log all events of that level or greater.