T.Y.B.Sc. (IT)

SEMESTER - V

LINUX ADMINISTRATION

Prof. Suhas Pednekar
Vice-Chancellor,
University of Mumbai,

Prof. Ravindra D. Kulkarni
Pro Vice-Chancellor,
University of Mumbai,

Prof. Prakash Mahanwar
Director,
IDOL, University of Mumbai,

Programme Co-ordinator : **Shri Mandar Bhanushe**
Head, Faculty of Science and Technology,
IDOL, University of Mumbai, Mumbai

Course Co-ordinator **:** **Mr. Sumedh Shejole**
Asst. Professor, B.Sc. I.T.
IDOL, University of Mumbai, Mumbai

Course Writers **:** **Mr. Bhavesh Shah**
Assistant Professor,
Vidyalankar School of Information Technology,
Wadala, Mumbai,

: **Mr. Jayesh Shinde**
Asst. Professor,
University Dept. of Information Technology,
Santacruz, Mumbai

September 2021, Print - I

Published by : Director
Institute of Distance and Open Learning ,
University of Mumbai,
Vidyanagari, Mumbai - 400 098.

DTP Composed and Printed by : Mumbai University Press
Vidyanagari, Santacruz (E), Mumbai - 400098

CONTENTS

SYLLABUS
T.Y.B.SC. INFORMATION TECHNOLOGY
SEMESTER - V
LINUX ADMINISTRATION

Unit : I

Introduction: Introduction to UNIX, Linux, GNU and Linux distributions

Duties of the System Administrator : The Linux System Administrator, Installing and Configuring Servers, Installing and Configuring Application
Software, Creating and Maintaining User Accounts, Backing Up and Restoring Files, Monitoring and Tuning Performance, Configuring a Secure System, Using Tools to Monitor Security

Booting and shutting down: Boot loaders-GRUB, LILO, Bootstrapping, Init process, rc scripts, Enabling and disabling services,

The File System: Understanding the File System Structure, Working with Linux-Supported File Systems, Memory and Virtual File Systems, Linux Disk Management
Network Configuration Files:

Unit : II

System Configuration Files: System wide Shell Configuration Scripts, System Environmental Settings, Network Configuration Files, Managing the init Scripts, Configuration Tool, Editing Your Network Configuration

TCP/IP Networking: Understanding Network Classes, Setting Up a Network Interface Card (NIC), Understanding Subnetting, Working with Gateways and Routers, Configuring Dynamic Host Configuration Protocol, Configuring the Network Using the Network

The Network File System: NFS Overview, Planning an NFS Installation, Configuring an NFS Server, Configuring an NFS Client, Using Automount Services, Examining NFS Security

Unit : III

Connecting to Microsoft Networks: Installing Samba, Configuring the Samba Server, Creating Samba Users 3, Starting the Samba Server, Connecting to a Samba Client, Connecting from a Windows PC to the Samba Server

Additional Network Services: Configuring a Time Server, Providing a Caching Proxy Server, Optimizing Network Services

Internet Services: Secure Services, SSH, scp, sftp Less Secure Services (Telnet, FTP, sync, rsh, rlogin, finger, talk and ntalk, Linux Machine as a Server, Configuring the xinetd Server, Comparing xinetd and Standalone, Configuring Linux Firewall Packages,

Unit : IV

Domain Name System: Understanding DNS, Understanding Types of Domain Servers, Examining Server Configuration Files, Configuring a Caching DNS Server, Configuring a Secondary Master DNS Server, Configuring a Primary Master Server, Checking Configuration

Configuring Mail Services: Tracing the Email Delivery Process, Mail User Agent (MUA), Introducing SMTP, Configuring Sendmail, Using the Postfix Mail Server, Serving Email with POP3 and IMAP, Maintaining Email Security

Configuring FTP Services: Introducing vsftpd, Configuring vsftpd, Advanced FTP Server Configuration, Using SFTP

Unit : V

Configuring a Web Server: Introducing Apache, Configuring Apache, Implementing SSI, Enabling CGI, Enabling PHP, Creating a Secure Server with SSL

Providing Web Services: Creating Mailing Lists, Setting Up Web-Based Email, Configuring an RSS Feed, Adding Search Functionality,

Unit : VI

Optimizing Internet Services: Optimizing LDAP Services, Optimizing DNS Services, Optimizing Mail Services, Optimizing FTP Services, Optimizing Web Services

System Administration: updating system, upgrading and customizing kernel, Administering Users and Groups Installing and Upgrading Software Packages

Books:

1. Beginning Linux by Neil Mathew 4th Edition
2. Red hat Linux Networking and System Administration by Terry Collings

References:

1. UNIX: Concepts and techniques, S. Das, Tata McGraw-Hill,
2. Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill
3. Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill

Term Work:
Assignments: Should contain at least 6 assignments (one per unit) covering the Syllabus.

Practical:

1. Installation of Red HAT/Fedora Linux operating system.
    a. Partitioning drives
    b. Configuring boot loader (GRUB/LILO)
    c. Network configuration
    d. Setting time zones
    e. Creating password and user accounts
    f. Shutting down
2. Software selection and installation
3. Programming Shell scripts for Linux administration
4. Linux system administration
    a. Becoming super user
    b. Temporarily changing user identity with su command
    c. Using graphical administrative tools
    d. Administrative commands
    e. Administrative configuration files
5. Connecting to the internet and configuring samba
    a. Setting up dial-up PPP
    b. Creating a dial- up connection with the internet configuration wizard
    c. Launching PPP connection
    d. Setting up linux as a proxy server
    e. Configuring mozilla or firefox to use as a proxy
6. Setting up local area network
    a. LAN topologies
    b. LAN equipment
    c. Networking with TCP/IP
    d. Configuring TCP/IP
    e. Adding windows computer’s to user LAN
    f. IP address classes
7. Server setup and configuration
    a. Setting up NFS file server

   b. Setting up Samba file server
   c. The Apache web server
   d. Setting up FTP server
   e. Setting up proxy server
8. Understanding COMPUTER SECURITY: Firewall and security configurations
   a. LINUX security checklist
   b. Securing linux with IP table firewalls
   c. Configuring an IP table firewall
   d. Securing Linux features
9. Programming using C.
10. Implementing Socket programs.
11. Setting up hardware devices including sound card and printers and others (USB devices etc).
12. Working with X-windows
   a. Switching between text and graphical consoles
   b. set up my video card, monitor and mouse for the X-server.
   c. Install KDE, change default desktop to KDE (or Gnome)
   d. Accessing X-window remotely.
   e. Installing TrueType fonts from my MS Windows partition.
   f. Display and Control a Remote Desktop using VNC.

1

INTRODUCTION

Unit Structure :

1.0 OBJECTIVES

In this chapter, you discover what Linux is and how it relates to its inspiration, UNIX. What is the GNU project? This chapter also discusses the duties and responsibilities of Linux system administrator.

1.1 INTRODUCTION

In recent years Linux has become a phenomenon. Major hardware vendors like IBM and Dell support Linux, and major software vendors like Oracle support their software running on Linux. Linux truly has become a viable operating system, especially in the server market. Linux owes its success to systems and applications that preceded it: UNIX and GNU software.

1.1.1 What is Unix?

As a version of Unix, the history of Linux naturally begins with Unix. The story begins in the late 1960s, when a concerted effort to develop new operating system techniques occurred. In 1968, a consortium of researchers from General Electric, AT&T Bell Laboratories, and the Massachusetts Institute of Technology carried out a special operating system research project called MULTICS (the Multiplexed Information and Computing Service). MULTICS incorporated many new concepts in multitasking, file management, and user interaction.

In 1969, Ken Thompson, Dennis Ritchie, and the researchers at AT&T Bell Laboratories developed the Unix operating system, incorporating many of the features of the MULTICS research project. They tailored the system for the needs of a research environment, designing it to run on minicomputers. From its inception, Unix was an affordable and efficient multiuser and multitasking operating system. They developed a new operating system, which was

1. Simple and elegant.
2. Written in the C programming language instead of in assembly code.
3. Able to recycle code.

Throughout the next couple of decades the development of UNIX continued. More things became possible to do and more hardware and software vendors added support for UNIX to their products. UNIX was initially found only in very large environments with mainframes and minicomputers (note that a PC is a "micro" computer). You had to work at a university, for the government or for large financial corporations in order to get your hands on a UNIX system.

But smaller computers were being developed, and by the end of the 80's, many people had home computers. By that time, there were several versions of UNIX available for the PC architecture, but none of them were truly free and more important: they were all terribly slow, so most people ran MS DOS or Windows 3.1 on their home PCs.

Many UNIX-like systems are available commercially, such as IBM’s AIX, HP’s HP-UX, and Sun’s Solaris. Some have been made available for free, such as FreeBSD and Linux.

Unix Philosophy

The UNIX operating system, and hence Linux, encourages a certain programming style. Following are a few characteristics

shared by typical UNIX programs and systems :

- **Simplicity:** Many of the most useful UNIX utilities are very simple and, as a result, small and easy to understand. KISS, “Keep It Small and Simple,” is a good technique to learn. Larger, more complex systems are guaranteed to contain larger, more complex bugs, and debugging is a chore that we’d all like to avoid.
- **Focus:** It’s often better to make a program perform one task well than to throw in every feature along with the kitchen sink. A program with “feature bloat” can be difficult to use and difficult to maintain. Programs with a single purpose are easier to improve as better algorithms or interfaces are developed.
- **Reusable Components:** Make the core of your application available as a library. Well-documented libraries with simple but flexible programming interfaces can help others to develop variations or apply the techniques to new application areas. Examples include the *dbm database library*, which is a suite of reusable functions rather than a single database management program.
- **Filters:** Many UNIX applications can be used as filters. That is, they transform their input and produce output. As you’ll see, UNIX provides facilities that allow quite complex applications to be developed from other UNIX programs by combining them in novel ways.
- **Open File Formats:** The more successful and popular UNIX programs use configuration files and data files that are plain ASCII text or XML. It enables users to use standard tools to change and search for configuration items and to develop new tools for performing new functions on the data files. A good example of this is the *ctags* source code cross-reference system, which records symbol location information as regular expressions suitable for use by searching programs.
- **Flexibility:** You can’t anticipate exactly how ingeniously users will use your program. Try to be as flexible as possible in your programming. Try to avoid arbitrary limits on field sizes or number of records. If you can, write the program so that it’s network-aware and able to run across a network as well as on a local machine. Never assume that you know everything that the user might want to do.

1.1.2 The GNU Project and Free Software Foundation :

In the early 1980s, AT&T began to recognize the commercial value of Unix. Consequently, they asserted proprietary rights to it and began charging a substantial license fee. Many who had contributed code to Unix believed that AT&T had unfairly appropriated their contributions. Not content merely to whine, MIT

researcher Richard Stallman launched the GNU (GNU is not Unix) project, which focused on creating a Unix-like operating system that could be freely distributed. As a vehicle in support of the GNU, Stallman and others created the Free Software Foundation (FSF) in 1984.

The GNU Project has already provided the software community with many applications that closely mimic those found on UNIX systems. All these programs, so-called GNU software, are distributed under the terms of the GNU General Public License (GPL).

The FSF promotes free software, but free software is not necessarily cost free software. The FSF intends the word free in the sense of freedom. Free software is software with which you can do the following:

1. Use for any purpose.
2. Study to learn how it works, and adapt to meet your needs.
3. Copy and redistribute.
4. Distribute as part of an improved software system.

As a practical matter, these freedoms require access to the software's source code, which is why some refer to Open Source Software (OSS), rather than free software. The Apache Web server and Linux are, in the opinion of many, the most significant OSS products created to date.

A few major examples of software from the GNU Project distributed under the GPL follow:

- GCC: The GNU Compiler Collection, containing the GNU C compiler
- G++: A C++ compiler, included as part of GCC
- GDB: A source code–level debugger
- GNU make: A version of UNIX make
- Bison: A parser generator compatible with UNIX yacc
- bash: A command shell
- GNU Emacs: A text editor and environment

Many other packages have been developed and released using free software principles and the GPL, including spreadsheets, source code control tools, compilers and interpreters, Internet tools, graphical image manipulation tools such as the Gimp, and two complete object-based environments: GNOME and KDE.

1.1.3 What is Linux?

As you may already know, Linux is a freely distributed implementation of a UNIX-like kernel, the low-level core of an operating system. Because Linux takes the UNIX system as its inspiration, Linux and UNIX programs are very similar. In fact, almost all programs written for UNIX can be compiled and run on Linux. Also, some commercial applications sold for commercial versions of UNIX can run unchanged in binary form on Linux systems.

Linux was developed in the early 1990s by Linus Torvalds at the University of Helsinki, with the help of UNIX programmers from across the Internet. It began as a hobby inspired by Andy Tanenbaum's Minix, a small UNIX-like system, but has grown to become a complete system in its own right. The intention is that the Linux kernel will not incorporate proprietary code but will contain nothing but freely distributable code.

Linux is a fast and stable open source operating system for personal computers (PCs) and workstations that features professional-level Internet services, extensive development tools, fully functional graphical user interfaces (GUIs), and a massive number of applications ranging from office suites to multimedia applications. As an operating system, Linux performs many of the same functions as Unix, Macintosh, Windows, and Windows NT. However, Linux is distinguished by its power and flexibility, along with being freely available. Most PC operating systems, such as Windows, began their development within the confines of small, restricted PCs, which have only recently become more versatile machines. Such operating systems are constantly being upgraded to keep up with the ever-changing capabilities of PC hardware. Linux, on the other hand, was developed in a different context. Linux is a PC version of the Unix operating system that has been used for decades on mainframes and minicomputers and is currently the system of choice for network servers and workstations. Linux brings the speed, efficiency, scalability, and flexibility of Unix to your PC, taking advantage of all the capabilities that PCs can now provide. Technically, Linux consists of the operating system program, referred to as the kernel, which is the part originally developed by Linus Torvalds. But it has always been distributed with a massive number of software applications, ranging from network servers and security programs to office applications and development tools. Linux has evolved as part of the open source software movement, in which independent programmers joined together to provide free, high-quality software to any user.

Current applications of Linux systems

Today Linux has joined the desktop market. Linux developers concentrated on networking and services in the

beginning, and office applications have been the last barrier to be taken down. We don't like to admit that Microsoft is ruling this market, so plenty of alternatives have been started over the last couple of years to make Linux an acceptable choice as a workstation, providing an easy user interface and MS compatible office applications like word processors, spreadsheets, presentations and the like.

On the server side, Linux is well-known as a stable and reliable platform, providing database and trading services for companies like Amazon, the well-known online bookshop, US Post Office, the German army and many others. Especially Internet providers and Internet service providers have grown fond of Linux as firewall, proxy and web server, and you will find a Linux box within reach of every UNIX system administrator who appreciates a comfortable management station. Clusters of Linux machines are used in the creation of movies such as "Titanic", "Shrek" and others. In post offices, they are the nerve centers that route mail and in large search engine, clusters are used to perform internet searches. These are only a few of the thousands of heavy-duty jobs that Linux is performing day-to-day across the world.

It is also worth to note that modern Linux not only runs on workstations, mid- and high-end servers, but also on "gadgets" like PDA's, mobiles, a shipload of embedded applications and even on experimental wristwatches. This makes Linux the only operating system in the world covering such a wide range of hardware.

1.1.4 Linux Distributions

As we have already mentioned, Linux is actually just a kernel. You can obtain the sources for the kernel to compile and install it on a machine and then obtain and install many other freely distributed software programs to make a complete Linux installation. These installations are usually referred to as Linux systems, because they consist of much more than just the kernel. Most of the utilities come from the GNU Project of the Free Software Foundation. As you can probably appreciate, creating a Linux system from just source code is a major undertaking. Fortunately, many people have put together ready-to-install distributions (often called flavors), usually downloadable or on CD-ROMs or DVDs, that contain not just the kernel but also many other programming tools and utilities. These often include an implementation of the X Window System, a graphical environment common on many UNIX systems. The distributions usually come with a setup program and additional documentation (normally all on the CD[s]) to help you install your own Linux system. Some well-known distributions, particularly on the Intel x86 family of processors, are Red Hat Enterprise Linux and its community-developed cousin Fedora, Novell SUSE Linux and the free open

SUSE variant, Ubuntu Linux, Slack ware, Gentoo, and Debian GNU/Linux. Currently, distrowatch.com lists numerous Linux distributions. Check out the Distro Watch site at http://distrowatch.com for details on many more Linux distributions.

1.1.5 Understanding the Difference between Windows and Linux

Throughout this book, topic by topic, we'll examine the specific contrasts between the two systems. In some chapters, you'll find that we don't derive any comparisons because a major difference doesn't really exist. But before we attack the details, let's take a moment to discuss the primary architectural differences between the two operating systems.

Single Users vs. Multiple Users vs. Network Users

Windows was designed according to the "one computer, one desk, one user" vision of Microsoft's cofounder Bill Gates. For the sake of discussion, we'll call this philosophy single-user. In this arrangement, two people cannot work in parallel running (for example) Microsoft Word on the same machine at the same time.

Linux borrows its philosophy from UNIX. It required a design that allowed for multiple users to log into the central machine at the same time. Various people could be editing documents, compiling programs, and doing other work at the exact same time. The operating system on the central machine took care of the "sharing" details so that each user seemed to have an individual system. This multiuser tradition continues today on all versions of Linux. Today, the most common implementation of a multiuser setup is to support servers – systems dedicated to running large programs for use by many clients. Each member of a department can have a smaller workstation on the desktop, with enough power for day-to-day work. When they need to do something requiring significantly more processing power or memory, they can run the operation on the server.

Both Linux and Windows are indeed capable of providing services such as databases over the network. We can call users of this arrangement network users, since they are never actually logged into the server, but rather, send requests to the server. The server does the work and then sends the results back to the user via the network. The catch in this case is that an application must be specifically written to perform such server/client duties.

The Monolithic Kernel and the Micro-Kernel

In operating systems, there are two forms of kernels. You have a monolithic kernel that provides all the services the user applications need. And then you have the micro-kernel, a small core set of services and other modules that perform other functions.

Linux, for the most, part adopts the monolithic kernel architecture; it handles everything dealing with the hardware and system calls. Windows works off a micro-kernel design. The kernel provides a small set of services and then interfaces with other executive services that provide process management, input/output (I/O) management, and other services. It has yet to be proved which methodology is truly the best way.

Separation of the GUI and the Kernel

Taking a cue from the Macintosh design concept, Windows developers integrated the GUI with the core operating system. One simply does not exist without the other. The benefit with this tight coupling of the operating system and user interface is consistency in the appearance of the system.

On the other hand, Linux (like UNIX in general) has kept the two elements – user interface and operating system – separate. The X Window System interface is run as a user-level application, which makes it more stable. If the GUI (which is complex for both Windows and Linux) fails, Linux's core does not go down with it. The process simply crashes, and you get a terminal window.

So which approach is better – Windows or Linux – and why? That depends on what you are trying to do. The integrated environment provided by Windows is convenient and less complex than Linux, but out of the box, it lacks the X Window System feature that allows applications to display their windows across the network on another workstation. Windows' GUI is consistent, but cannot be turned off, whereas the X Window System doesn't have to be running (and consuming valuable memory) on a server.

The Network Neighbourhood

The native mechanism for Windows users to share disks on servers or with each other is through the Network Neighbourhood. In a typical scenario, users attach to a share and have the system assign it a drive letter. As a result, the separation between client and server is clear. The only problem with this method of sharing data is more people-oriented than technology-oriented: People have to know which servers contain which data.

Linux, using the Network File System (NFS), has supported the concept of mounting since its inception. In fact, the Linux Automounter can dynamically mount and unmount partitions on an as-needed basis. Under Linux NFS, users never have to know server names or directory paths, and their ignorance is your bliss. No more questions about which server to connect to. Even better, users need not know when the server configuration must change. Under Linux, you can change the names of servers and adjust this information on client-side systems without making any

announcements or having to re-educate users. Anyone who has ever had to reorient users to new server arrangements is aware of the repercussions that can occur.

The Registry vs. Text Files

Think of the Windows Registry as the ultimate configuration database – thousands upon thousands of entries, only a few of which are completely documented. In other words, the Windows Registry system is, at best, difficult to manage. Although it's a good idea in theory, most people who have serious dealings with it don't emerge from battle without a scar or two.

Linux does not have a registry. This is both a blessing and a curse. The blessing is that configuration files are most often kept as a series of text files. This setup means you're able to edit configuration files using the text editor of your choice rather than tools like regedit. The curse of a no-registry arrangement is that there is no standard way of writing configuration files. Each application can have its own format.

In reality, having text files hold configuration information usually turns out to be an efficient method. Once set, they rarely need to be changed; even so, they are straight text files and thus easy to view when needed. Even more helpful is that it's easy to write scripts to read the same configuration files and modify their behavior accordingly. This is especially helpful when automating server maintenance operations, which is crucial in a large site with many servers.

1.2 DUTIES OF SYSTEM ADMINISTRATOR

Linux is a multi-user, multitasking operating system from the ground up. In this regard the system administrator has flexibility — and responsibility — far beyond those of other operating systems. Red Hat has employed innovations that extend these duties even for the experienced Linux user. This chapter briefly looks at those responsibilities, which are covered in more detail in later chapters.

1.2.1 The Linux System Administrator

Using Linux involves much more than merely sitting down and turning on the machine. Often you hear talk of a "steep learning curve" but that discouraging phrase can be misleading. Linux is quite different from the most popular commercial operating systems in a number of ways. Make no mistake: Every computer in the world has a system administrator. It may be — and probably is — true that the majority of system administrators are those who decided what software and peripherals were bundled with the machine when it was shipped. That status quo remains because

the majority of users who acquire computers for use as appliances probably do little to change the default values. But the minute a user decides on a different wallpaper image or adds an application that was acquired apart from the machine itself, he or she has taken on the role of system administration. By its very nature as a modern, multi-user operating system, Linux requires a degree of administration greater than that of less robust, home-market systems. This means that even if you use just a single machine connected to the Internet by a dial-up modem — or not even connected at all — you have the benefits of the same system employed by some of the largest businesses in the world, and will do many of the same things that IT professionals employed by those companies are paid to do. Administering your system does involve a degree of learning, but it also means that in setting up and configuring your own system you gain skills and understanding that raise you above mere “computer user” status.

By definition, the Linux system administrator is the person who has “root” access, which is to say the one who is the system’s “super user” (or root user). A standard Linux user is limited to whatever he or she can do with the underlying engine of the system. But the root user has unfettered access to everything — all user accounts, their home directories, and the files therein; all system configurations; and all files on the system. A certain body of thought says that no one should ever log in as “root,” because system administration tasks can be performed more easily and safely through other, more specific means, which we discuss in due course. Because the system administrator has full system privileges, your first duty is to know what you’re doing, lest you break something.

Linux system administrators are likely to understand the necessity of active system administration more than those who run whatever came on the computer, assuming that things came properly configured from the factory. The user or enterprise that decides on Linux has decided, also, to assume the control that Linux offers, and the responsibilities that this entails.

The word duty implies a degree of drudgery; in fact, it’s a manifestation of the tremendous flexibility of the system measured against the responsibility to run a tight organization. These duties do not so much constrain you, the system administrator, as free you to match the job to the task. Let’s take a brief look at them.

1.2.2 Installing and Configuring Servers

The server can be described as a computer which offers some type of service to clients. The server may provide file or printer sharing, File Transfer Protocol (FTP) or Web access, or email-processing tasks. Don’t think of a server as a standalone

workstation; think of it as a computer that specifically performs these services for many users. Whenever a server is connected to other machines outside your physical control, there are security implications to consider. You want your users to have easy access to the things they need, but you don't want to open up the system you're administering to the whole wide world.

Linux distributions used to ship with all imaginable servers turned on by default. Just installing the operating system on the computer would install and configure — with default parameters — all the services available with the distribution. This was a reflection of an earlier, more innocent era in computing when people did not consider vandalizing other people's machines to be good sportsmanship. Unfortunately, the realities of this modern, more dangerous world dictate that all but the most essential servers remain turned off unless specifically enabled and configured. This duty falls to the system administrator. You need to know exactly which servers you need and how to employ them, and to be aware that it is bad practice and a potential security nightmare to enable services that the system isn't using and doesn't need.

1.2.3 Installing and Configuring Application Software

Although it is possible for individual users to install some applications in their home directories – drive space set aside for their own files and customizations – these applications may not be available to other users without the intervention of the user who installed the program or the system administrator. Besides, if an application is to be used by more than one user, it probably needs to be installed higher up in the Linux file hierarchy, which is a job that only the system administrator can perform. The administrator can even decide which users may use which applications by creating a "group" for that application and enrolling individual users in that group.

Configuration and customization of applications is to some extent at the user's discretion, but not entirely. "Skeleton" configurations – administrator determined default configurations – set the baseline for user employment of applications. If there are particular forms, for example, that are used throughout an enterprise, the system administrator would set them up or at least make them available by adding them to the skeleton configuration. The same applies to configuring user desktops and in even deciding what applications should appear on user desktop menus. For instance, your company may not want to grant users access to the games that ship with modern Linux desktops. You may also want to add menu items for newly installed or custom applications. The system administrator brings all this to pass.

1.2.4 Creating and Maintaining User Accounts

Not just anyone can show up and log on to a Linux machine. An account must be created for each user and no one but the system administrator can do this. It also involves decisions that either you or your company must make. You might want to let users select their own passwords, which would no doubt make them easier to remember but which probably would be easier for a malefactor to crack. You might want to assign passwords, which is more secure in theory but increases the likelihood that users will write them down on a conveniently located scrap of paper — a risk if many people have access to the area where the machine(s) is located. You might decide that users must change their passwords periodically — something you can configure Red Hat Enterprise Linux to prompt users about. To what may specific users have access? It might be that there are aspects of your business that make Web access desirable, but you don't want everyone spending their working hours surfing the Web. If your system is at home, you may wish to limit your children's access to certain Web sites.

These and other issues are part of the system administrator's duties in managing user accounts. Whether the administrator or his or her employer establishes policies governing accounts, these policies should be delineated — preferably in writing for a company — for the protection of all concerned.

1.2.5 Backing Up and Restoring Files

There is a considerable need to back up important files so that the system can be up and running again with minimal disruption in the event of hardware, security, or administration failure. Only the system administrator may do this. Because of its built-in security features, Linux doesn't allow even users to back up their own files to removable disks. It's not enough to know that performing backups is your job. You need to formulate a strategy for making sure your system is not vulnerable to catastrophic disruption. If you have a high-capacity tape drive and several good sets of restore disks, you might make a full system backup every few days. If you are managing a system with scores of users, you might find it more sensible to back up user accounts and system configuration files.

Once you decide *what* to back up, you need to decide *how* frequently to perform backups, whether to maintain a series of *incremental* backups — adding only files that have changed since the last backup — or *multiple* full backups, and *when* these backups should be performed. Do you trust an automated, unattended process?

Restoring files from your backup media is no less important than backing them up in the first place. Be certain you can restore

your files if the need arises by testing your restore process at least once during a noncritical time. Periodically testing your backup media is also a good idea.

Backing up is only part of the story. You also need to formulate a plan for bringing the system back up after a failure. A system failure could be caused by any number of problems, either related to hardware or software (application, system configuration) trouble, and could range from a minor inconvenience to complete shutdown.

If you are the administrator of servers and workstations for a business, you should have a disaster recovery plan in place. Such a plan takes into account the type of data and services provided and how much fault tolerance your systems require — that is, how long your systems could be down and what effect that would have on your company's ability to conduct business. Based on the level of fault tolerance you require, your disaster recovery plan should list as many possible failures as you can anticipate and detail the steps required to restore your systems.

1.2.6 Monitoring and Tuning Performance

The default installation of Red Hat Enterprise Linux goes a long way toward capitalizing on existing system resources. There is no "one size fits all" configuration, however. Linux is infinitely configurable, or close to it.

System tuning is an ongoing process aided by a variety of diagnostic and monitoring tools. Some performance decisions are made at installation time, while others are added or tweaked later. A good example is the use of the *hdparm* utility, which can increase throughput in IDE drives considerably.

Proper monitoring allows you to detect a misbehaving application that consumes more resources than it should or fails to exit completely upon closing. Through the use of system performance tools, you can determine when hardware — such as memory, added storage, or even something as elaborate as a hardware RAID — should be upgraded for more cost-effective use of a machine in the enterprise or for complicated computational tasks such as three-dimensional rendering. Possibly most important, careful system monitoring and diagnostic practices give you a heads-up when a system component is showing early signs of failure, so that you can minimize any potential downtime.

In any case, careful system monitoring plus wise use of the built-in configurability of Linux allows you to squeeze the best possible performance from your existing equipment, from customizing video drivers to applying special kernel patches or

simply turning off unneeded services to free memory and processor cycles.

1.2.7 Configuring a Secure System

The system administrator's task, first and foremost, is to make certain that no data on the machine or network is likely to become corrupted, whether by hardware or power failure, misconfiguration or user error (to the extent that the latter can be avoided), or malicious or inadvertent intrusion from elsewhere. This means doing all the tasks described throughout this chapter, and doing them well, with a full understanding of their implications.

Depending on how a Linux machine is connected, and to what; the sensitivity of the data it contains; and the uses to which it is put, security can be as simple as turning off unneeded services, monitoring the Red Hat security mailing list to make sure that all security advisories are followed, regularly using system utilities to keep the system up to date, and otherwise engaging in good computing practices to make sure that the system runs robustly. It's almost a full-time job, involving levels of security permissions within the system and systems to which it is connected; elaborate firewalls to protect not just Linux machines but machines that, through their use of non-Linux software, are far more vulnerable; and physical security — making sure that no one steals the machine itself! For any machine connected to another machine, security means hardening against attacks and making certain that no one else uses your machine as a platform for launching attacks against others. If you run Web, FTP, or mail servers, it means giving access to only those who are entitled to it, while locking out everyone else. It means making sure that passwords are not easily guessed and not made available to unauthorized persons. It means that disgruntled former employees no longer have access to the system and that no unauthorized person may copy files from your machines.

Security is an ongoing process. The only really secure computer is one that contains no data, is unplugged from networks and power supplies, has no keyboard attached, and resides in a locked vault. While this is theoretically true, it implies that security diminishes the usefulness of the machine. In the chapters that follow, you learn about the many tools that Red Hat provides to help you guard against intrusion, even to help you prevent intrusion into non-Linux machines that may reside on your network. Linux is designed from the beginning with security in mind. In all your tasks you should maintain that same security awareness.

Your job as system administrator is to strike the right balance between maximum utility and maximum safety, all the while bearing in mind that confidence in a secure machine today means nothing about the machine's security tomorrow.

1.2.8 Using Tools to Monitor Security

People who, for purposes of larceny or to amuse themselves, like to break into computers — they're called crackers — are a clever bunch. If there is a vulnerability in a system, they will find it. Fortunately, the Linux development community is quick to find potential exploits and to create ways of slamming the door shut before crackers can enter. Fortunately, too, Red Hat is diligent in making available new, patched versions of packages in which potential exploits have been found. Your first and best security tool, therefore, is making sure that whenever a security advisory is issued, you download and install the repaired package. This line of defense can be annoying but it is nothing compared to rebuilding a compromised system. As good as the bug trackers are, sometimes their job is reactive. Preventing the use of your machine for nefarious purposes and guarding against intrusion are, in the end, your responsibility alone. Red Hat equips you with tools to detect and deal with unauthorized access of many kinds. If your machine is connected to the Internet, you will be amazed at the number of attempts made to break into your machine. You'll be struck by how critical the issue of security is.

1.3 SUMMARY

- In 1969, Ken Thompson, Dennis Ritchie, and the researchers at AT&T Bell Laboratories developed the Unix operating system, which was
  - → Simple and elegant.
  - → Written in the C programming language instead of in assembly code.
  - → Able to recycle code.
- MIT researcher Richard Stallman launched the GNU (GNU is not Unix) project, which focused on creating a Unix-like operating system that could be freely distributed.
- The GNU Project has already provided the software community with many applications that closely mimic those found on UNIX systems. All these programs, so-called GNU software, are distributed under the terms of the GNU General Public License (GPL).
- Linux is a freely distributed implementation of a UNIX-like kernel, the low-level core of an operating system.
- Linux was developed in the early 1990s by Linus Torvalds at the University of Helsinki, with the help of UNIX programmers from across the Internet. The intention is that the Linux kernel will not incorporate proprietary code but will contain nothing but freely distributable code.

- Linux is a fast and stable open source operating system for personal computers (PCs) and workstations that features professional-level Internet services, extensive development tools, fully functional graphical user interfaces (GUIs), and a massive number of applications ranging from office suites to multimedia applications.
- Many people have put together ready-to-install distributions (often called flavors), usually downloadable or on CD-ROMs or DVDs, that contain not just the kernel but also many other programming tools and utilities.
- The primary architectural differences between Microsoft Windows and Linux are
  - → Single Users vs. Multiple Users vs. Network Users
  - → The Monolithic Kernel and the Micro-Kernel
  - → Separation of the GUI and the Kernel
  - → The Network Neighbourhood
  - → The Registry vs. Text Files
- By definition, the Linux system administrator is the person who has “root” access, which is to say the one who is the system’s “super user” (or root user).
- Duties of System Administrator
  - → Installing and Configuring Servers
  - → Installing and Configuring Application Software
  - → Creating and Maintaining User Accounts
  - → Backing Up and Restoring Files
  - → Monitoring and Tuning Performance
  - → Configuring a Secure System
  - → Using Tools to Monitor Security

1.4 REVIEW QUESTIONS

1. Write a short note on the history of Unix and the invention of Linux.
2. Write a short note on GNU project and Free Software Foundation.
3. Explain the primary architectural difference between Windows and Linux.
4. Explain the duties of Linux System Administrator?

5. Define Linux Administrator? Explain Backing Up and Restoring Files in Linux?
6. Explain Monitoring and Tuning Performance in Linux.
7. Explain how to Configure a Secure System?

1.5 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

Beginning Linux Programming 4th Edition by Neil Mathew, Richard Stone. Wiley Publishing

Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill

Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill

Red Hat Linux Networking and System Administration 3rd Edition by Terry Collins and Kurt Wall.

Sybex – RHCE Red Hat Certified Engineer Study Guide

Red Hat Certified Technician & Engineer by Asghar Ghori.

www.thegeekstuff.com

www.tlpd.org

www.linuxtopia.org

2

BOOTING AND SHUTTING DOWN

Unit Structure :

2.0 OBJECTIVES

In this chapter, we discuss the bootstrapping of the Linux operating system with GRUB and LILO. We then step through the processes of starting up and shutting down the Linux environment. We discuss the scripts that automate this process, as well as the parts of the process for which modification is acceptable.

2.1 INTRODUCTION

As operating systems have become more complex, the process of starting up and shutting down has become more comprehensive. Anyone who has undergone the transition from a straight DOS-based system to a Windows 2003/XP-based system has experienced this transition firsthand. Not only is the core operating system brought up and shut down, but also an impressive list of services must be started and stopped. Like Windows, Linux comprises an impressive list of services that are turned on as part of the boot procedure.

All Red Hat systems, whether Fedora Core or Enterprise Linux, use a similar procedure for starting up the operating system. As the system boots, the operating system loads programs in an orderly fashion. You are able to make changes to the programs that load and their configurations after the system has booted. The changes you make will then affect the boot process the next time and all subsequent times that your system boots. The process of shutting down the system also follows a consistent, orderly method that you can customize as you desire. For a clear understanding of how your system works, it is good to know the methodology behind the orderly process of bringing your system up as well as shutting it down. By knowing this process in depth, you can make any desired changes to the configuration files and gain total control over the functionality of your system. You will also be able to easily find problems that may be keeping your system from booting properly and quickly correct them. This chapter gives you the details about what happens when your start and shut down your system.

2.2 The Boot Process

The act of turning on a computer system and causing its operating system to be loaded is called **booting.** The name comes from an image of the computer pulling itself up from its bootstraps, but the act itself slightly more realistic.

When a PC is booted, the **BIOS** will do various tests to check that everything looks all right, and will then start the actual booting. This process is called the power on self test, or **POST** for short.

The BIOS is the only way to communicate with the system components until the operating system is up and running and able to take over system management functions. Unlike the operating system, which is installed on a user-writable disk, such as a floppy, CD-ROM, or hard drive, the system BIOS is typically installed on a read-only memory (ROM) chip physically attached to the system board. This ROM chip is a type of chip usually referred to as an electronically erasable programmable read-only memory (EEPROM) chip, meaning that it is not normally writable by the end user. It is possible to rewrite an EEPROM BIOS chip, but this requires a program from the chip manufacturer and is not a process that should be taken lightly as any errors here could make your system totally unusable. After the BIOS loads, it performs some diagnostics on the hardware, checks the installed components to be sure they are functioning, and checks the system RAM. Next, the BIOS tries to find a system drive from which it can load the boot program to begin the process of starting the operating system.

It will choose a **disk drive** (typically the first CD-ROM drive, if there is a CD-ROM inserted, otherwise the first hard disk, if one is installed in the computer; the order might be configurable, however) and will then read its very first sector. This is called the **boot sector;** for a hard disk, it is also called the **master boot record (MBR)**, since a hard disk can contain several partitions, each with their own boot sectors.

The boot sector contains a small program (small enough to fit into one sector) whose responsibility is to read the actual operating system from the disk and start it. When booting Linux from a CD-ROM disk, the boot sector contains code that just reads the first few hundred blocks (depending on the actual kernel size, of course) to a predetermined place in memory. On a Linux boot CD-ROM, there is no file system; the **kernel** is just stored in consecutive sectors, since this simplifies the boot process. It is possible, however, to boot from a floppy with a file system, by using **LILO**, the Linux Loader, or **GRUB**, the GR and Unifying Boot loader.

When booting from the hard disk, the code in the master boot record will examine the partition table (also in the master boot record), identify the active partition (the partition that is marked to be bootable), read the boot sector from that partition, and then start the code in that boot sector. The details vary, however, since it is generally not useful to have a separate partition for just the kernel image, so the code in the partition's boot sector can't just read the disk in sequential order, it has to find the sectors wherever the file system has put them. There are several ways around this problem, but the most common way is to use a boot loader like LILO or GRUB.

When booting, the boot loader will normally go right ahead and read in and boot the default kernel. It is also possible to configure the boot loader to be able to boot one of several kernels, or even other operating systems than Linux, and it is possible for the user to choose which kernel or operating system is to be booted at boot time. LILO, for example, can be configured so that if one holds down the alt, shift, or ctrl key at boot time (when LILO is loaded), LILO will ask what is to be booted and not boot the default right away. Alternatively, the boot loader can be configured so that it will always ask, with an optional timeout that will cause the default kernel to be booted.

After the Linux kernel has been read into the memory, by whatever means, and is started for real, roughly the following things happen:

- The Linux kernel is installed compressed, so it will first uncompress itself. The beginning of the kernel image contains a small program that does this.
- After this, the kernel checks what other hardware there is (hard disks, floppies, network adapters, etc), and configures some of its device drivers appropriately; while it does this, it outputs messages about its findings.
- Then the kernel will try to mount the root file system. The file system type is detected automatically. The root file system is usually mounted read-only. This makes it possible to check the file system while it is mounted; it is not a good idea to check a file system that is mounted read-write.
- After this, the kernel starts the program init (located in /sbin/init) in the background (this will always become first process with PID 1). init does various start-up chores. It will at least start some essential background daemons.
- init references the inittab file to determine the default run level to boot into and starts a getty for virtual consoles and serial lines. getty is the program which lets people log in via virtual consoles and serial terminals. init may also start some other programs, depending on how it is configured.
- After this, the boot is complete, and the system is up and running normally.

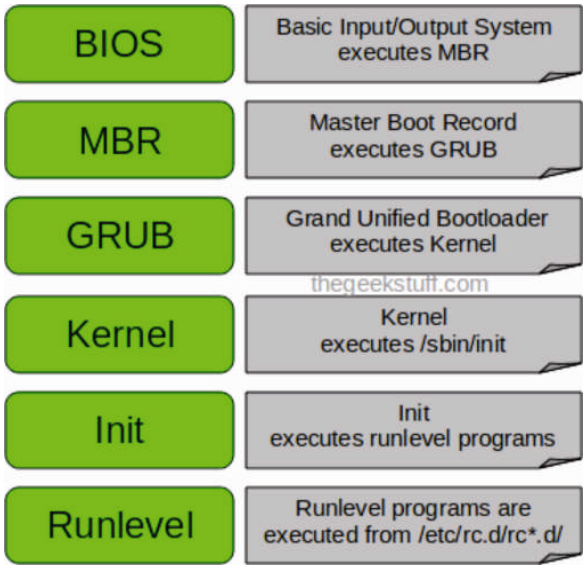


Figure 2-1: Pictorial view of the Booting Process of a Linux system

2.3 BOOT LOADERS

For any operating system to boot on standard PC hardware, you need what is called a boot loader. If you have only dealt with Windows on a PC, you have probably never needed to interact directly with a boot loader. The boot loader is the first software program that runs when a computer starts. It is responsible for handing over control of the system to the operating system. Typically, the boot loader will reside in the Master Boot Record (MBR) of the disk, and it knows how to get the operating system up and running. The main choices that come with Linux distributions are GRUB (Grand Unified Bootloader) and LILO (Linux Loader). GRUB is the most common boot loader that ships with the newer distributions of Linux and it has a lot more features than LILO. Both LILO and GRUB can be configured to boot other non-native operating systems.

2.3.1 GRUB

Most modern Linux distributions use GRUB as the default boot loader during installation. GRUB is the default boot loader for Fedora, Red Hat Enterprise Linux (RHEL), OpenSUSE, Mandrake, Ubuntu, and a host of other Linux distributions. GRUB aims to be compliant with the Multiboot Specification and offers many features. The GRUB boot process happens in stages. Each stage is taken care of by special GRUB image files, with each preceding stage helping the next stage along. Two of the stages are essential, and any of the other stages are optional and dependent on the particular system setup.

Stage 1

The image file used in this stage is essential and is used for booting up GRUB in the first place. It is usually embedded in the MBR of a disk or in the boot sector of a partition. The file used in this stage is appropriately named stage 1. A Stage 1 image can next either load Stage 1.5 or load Stage 2 directly.

Stage 2

The Stage 2 images actually consist of two types of images: the intermediate stage 1.5 (optional image) and the actual stage 2 image file. The Stage 1.5 images serve as a bridge between Stage 1 and Stage 2. The Stage 1.5 images are file system–specific; that is, they understand the semantics of one file system or the other. The Stage 1.5 images have names of the form – x_stage_1_5 – where x can be a file system of type e2fs, reiserfs, fat, jfs, minix, xfs, etc. For example, the Stage 1.5 image that will be required to load an operating system (OS) that resides on a File Allocation Table (FAT) file system will have a name like fat_stage1_5. The Stage 1.5 images allow GRUB to access several file systems.

When used, the Stage 1.5 image helps to locate the Stage 2 image as a file within the file system.

Next comes the actual stage2 image. It is the core of GRUB. It contains the actual code to load the kernel that boots the OS, it displays the boot menu, and it also contains the GRUB shell from which GRUB commands can be entered. The GRUB shell is interactive and helps to make GRUB flexible. For example, the shell can be used to boot items that are not currently listed in GRUB's boot menu or to bootstrap the OS from an alternate supported medium.

Other types of Stage 2 images are the stage2_eltorito image, the nbgrub image, and the pxegrub image. The stage2_eltorito image is a boot image for CD-ROMs. The nbgrub and pxegrub images are both network-type boot images that can be used to bootstrap a system over the network (using Bootstrap Protocol [BOOTP], Dynamic Host Configuration Protocol [DHCP], Preboot Execution Environment [PXE], Etherboot, or the like). A quick listing of the contents of the /boot/grub directory of most Linux distributions will show some of the GRUB images.

Since you only have to install GRUB once on the MBR or partition of your choice, you have the luxury of simply editing a text file, (/boot/grub/menu.1st), in order to make changes to your boot loader. When you are done editing this file, you can reboot and select any new kernel that you added to the configuration. The configuration file looks like the following (please note that line numbers 1–16 have been added to the output to aid readability):

```
[root@fedora-serverA  ~]# cat /boot/grub/menu.lst
1) # grub.conf generated by anaconda
2) # Note that you do not have to re-run grub after making changes to this file
3) # NOTICE:  You have a /boot partition.  This means that
4) #          all kernel and initrd paths are relative to /boot/, eg.
5) #          root (hd0,0)
6) #          kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
7) #          initrd /initrd-version.img
8) #boot=/dev/sda
9) default=0
10) timeout=5
11) splashimage=(hd0,0)/grub/splash.xpm.gz
12) hiddenmenu
13) title Fedora (2.6.25-14.fc9.i686)
14)   root (hd0,0)
15)   kernel /vmlinuz-2.6.25-14.fc9.i686 ro  root=UUID=7db5-4c27 rhgb quiet
16)       initrd   /initrd-2.6.25-14.fc9.i686.img
```

Figure 2-2: The /boot/grub/grub.conf GRUB configuration file

The entries in the preceding sample configuration file for GRUB are discussed here:

- Lines 1 – 8, All lines that begin with the hash sign (#) are comments and are ignored.
- Line 9, **default:** This directive tells GRUB which entry to automatically boot. The numbering starts from zero. The preceding sample file contains only one entry – the entry titled Fedora (2.6.25-14.fc9.i686).
- Line 10, **timeout:** This means that GRUB will automatically boot the default entry after five seconds. This can be interrupted by pressing any key on the keyboard before the counter runs out.
- Line 11, **splashimage:** This line specifies the name and location of an image file to be displayed at the boot menu. This is optional and can be any custom image that fits GRUB's specifications.
- Line 12, **hiddenmenu:** This entry hides the usual GRUB menu. It is an optional entry.
- Line 13, **title:** This is used to display a short title or description for the following entry it defines. The title field marks the beginning of a new boot entry in GRUB.
- Line 14, **root:** You should notice from the preceding listing that GRUB still maintains its device naming convention (e.g., (hd0,0) instead of the usual Linux /dev/sda1).
- Line 15, **kernel:** Used for specifying the path to a kernel image. The first argument is the path to the kernel image in a partition. Any other arguments are passed to the kernel as boot parameters.
- Line 16, **initrd:** The initrd option allows you to load kernel modules from an image, not the modules from /lib/modules.

2.3.2 LILO

LILO, short for Linux Loader, is a boot manager. It allows you to boot multiple operating systems, provided each system exists on its own partition. In addition to booting multiple operating systems, with LILO, you can choose various kernel configurations or versions to boot. This is especially handy when you're trying kernel upgrades before adopting them. Configuring LILO is straightforward: A configuration file (/etc/lilo.conf) specifies which partitions are bootable and, if a partition is Linux, which kernel to load. When the /sbin/lilo program runs, it takes this partition information and rewrites the boot sector with the necessary code to present the options as specified in the configuration file. At boot time, a prompt (usually lilo:) is displayed, and you have the option of specifying the

operating system. (Usually, a default can be selected after a timeout period.) LILO loads the necessary code, the kernel, from the selected partition and passes full control over to it.

LILO is what is known as a two-stage boot loader. The first stage loads LILO itself into memory and prompts you for booting instructions with the lilo: prompt or a colourized boot menu. Once you select the OS to boot and press enter, LILO enters the second stage, booting the Linux operating system. As was stated earlier in the chapter, LILO has somewhat fallen out of favour with most of the newer Linux distributions. Some of the distributions do not even give you the option of selecting or choosing LILO as your boot manager!

2.3.3 Difference between LILO and GRUB

LILO	GRUB
LILO stand for Linux Loader	GRUB stands for Grand Unified Bootloader
LILO provides a textual interface	GRUB provides a graphical interface
LILO has no interactive command interface	GRUB has interactive command interface
LILO supports only up to 16 different boot selections	GRUB supports an unlimited number of boot entries
LILO cannot boot from network	GRUB can boot from network
LILO stores information regarding the location of the operating systems it can to load physically on the MBR. If you change your LILO config file, you have to rewrite the LILO stage one boot loader to the MBR	With GRUB, if the configuration file is configured incorrectly, it will simply default to the GRUB command-line interface
The LILO bootstrap process involves locating the kernel by in essence (it's more complicated than this) pointing to the first logical-sector of the Kernel file	The GRUB bootstrap process is more filesystem aware and can locate a kernel file in a filesystem without having to specify a logical-sector
LILO must be written again every time you change the configuration file	GRUB does not have to be written every time you change the configuration file

2.4 BOOTSTRAPPING

Since you are already familiar with the boot processes of other operating systems and thus already know the boot cycle of your hardware. This section will cover the process of bootstrapping the operating system. We'll begin with the Linux boot loader (usually GRUB for PCs).

Kernel Loading

Once GRUB has started and you have selected Linux as the operating system to boot, the first thing to get loaded is the kernel. Keep in mind that no operating system exists in memory at this point, and PCs (by their unfortunate design) have no easy way to access all of their memory. Thus, the kernel must load completely into the first megabyte of available random access memory (RAM). In order to accomplish this, the kernel is compressed. The head of the file contains the code necessary to bring the CPU into protected mode (thereby removing the memory restriction) and decompress the remainder of the kernel.

Kernel Execution

With the kernel in memory, it can begin executing. It knows only whatever functionality is built into it, which means any parts of the kernel compiled as modules are useless at this point. At the very minimum, the kernel must have enough code to set up its virtual memory subsystem and root file system (usually, the ext3 file system). Once the kernel has started, a hardware probe determines what device drivers should be initialized. From here, the kernel can mount the root file system. (You could draw a parallel of this process to that of Windows being able to recognize and access its C drive.) The kernel mounts the root file system and starts a program called init, which is discussed in the next section.

2.5 The init process

The **init** process is the first non-kernel process that is started, and, therefore, it always gets the process ID number of 1. The */sbin/init* program is the first system process that runs after the kernel has configured the system devices and mounted the system drives. The /init program is like the project manager of the system because it manages the remaining steps of booting the system and is the parent or grandparent of all the rest of the automatically started system boot processes. Basically, the init program coordinates the order of the many scripts it will run to complete system setup. The first script /init runs is the */etc/rc.d/rc.sysinit* script. This script starts system swap, checks the file systems, and performs other system initialization. Then the init command refers to the */etc/inittab* script to get information about how to start the system, which system initialization script to run and bring the system to the runlevel indicated in the inittab script.

The term **runlevel** has been used a few times so far in this chapter and now is a good time to learn more about runlevels and why they are used. There are typically eight runlevels on Linux systems, but we are only interested in the seven used on Fedora Core or Red Hat Enterprise Linux systems. Each of the runlevels has a set of processes associated with that runlevel that will be started by entering that runlevel. The runlevels on a Fedora Core or Enterprise Linux system and their purpose are:

- 0 — Halt
- 1 — Single-user mode
- 2 — Multiuser mode, but without Network File System (NFS)
- 3 — Full multiuser mode (without a graphical user interface, GUI)
- 4 — Not used (user-definable)
- 5 — Full multiuser mode (with a GUI)
- 6 — Reboot

When it is told to enter a runlevel, init executes a script, as dictated by the /etc/inittab file. The default runlevel that the system boots into is determined by the initdefault entry in the /etc/inittab file. If, for example, the entry in the file is **id:3:initdefault:** this means that the system will boot into runlevel 3. But if, on the other hand, the entry in the file is **id:5:initdefault:** this means the system will boot into runlevel 5, with the X Window subsystem running with a graphical login screen.

2.6 RC SCRIPTS

In the preceding section, we mentioned that the /etc/inittab file specifies which scripts to run when runlevels change. These scripts are responsible for either starting or stopping the services that are particular to the runlevel. Because of the number of services that need to be managed, rc scripts are used. The main one, */etc/rc.d/rc*, is responsible for calling the appropriate scripts in the correct order for each runlevel. For each runlevel, a subdirectory exists in the */etc/rc.d* directory. These runlevel sub-directories follow the naming scheme of rc X .d, where X is the runlevel. For example, all the scripts for runlevel 3 are in */etc/rc.d/rc3.d*. In the runlevel directories, symbolic links are made to scripts in the */etc/rc.d/init.d* directory. Instead of using the name of the script as it exists in the /etc/rc.d/init.d directory, however, the symbolic links are prefixed with an S, if the script is to start a service, or with a K, if the script is to stop (or kill) a service. (Note that these two letters are case-sensitive. You must use uppercase letters, or the startup scripts will not recognize them.) In many cases, the order in which these scripts are run makes a difference. For example, you can’t start services that rely on a configured

network interface without first enabling and configuring the network interface. To enforce order, a two-digit number is suffixed to the S or K. Lower numbers execute before higher numbers; for example, /etc/rc.d/rc3.d/ S10network runs before /etc/rc.d/rc3.d/S55sshd (S10network configures the network settings, and S55sshd starts the Secure Shell [SSH] server).

The scripts pointed to in the */etc/rc.d/init.d* directory are the workhorses; they perform the actual process of starting and stopping services. When */etc/rc.d/rc* runs through a specific runlevel's directory, it invokes each script in numerical order. It first runs the scripts that begin with a K and then the scripts that begin with an S. For scripts starting with K, a parameter of stop is passed. Likewise, for scripts starting with S, the parameter start is passed.

2.7 ENABLING AND DISABLING A SERVICE

At times, you may find that you simply don't need a particular service to be started at boot time. This is especially important if you are configuring the system as a server and need only specific services and nothing more. As described in the preceding sections, you can cause a service not to be started by simply renaming the symbolic link in a particular runlevel directory; rename it to start with a K instead of an S. Once you are comfortable working with the command line, you'll quickly find that it is easy to enable or disable a service.

The startup runlevels of the service/program can also be managed using the **chkconfig** utility.

Using the chkconfig utility you can perform enable and disable a service. In the following example using the chkconfig utility is used to list, enable and disable the carpald.sh program.

```
[root@fedora-serverA ~]# chkconfig --list carpald
Carpald         0:off 1:off 2:off 3:on 4:off 5:on 6:off

[root@serverA ~]# chkconfig --level 2 carpald on

[root@fedora-serverA ~]# chkconfig --list carpald
carpald         0:off 1:off 2:on 3:on 4:off 5:on 6:off

 [root@fedora-serverA ~]# chkconfig carpald off

[root@fedora-serverA ~]# chkconfig --list carpald
carpald 0:off 1:off 2:off 3:off 4:off 5:off 6:off

[root@fedora-serverA ~]# chkconfig --del carpald
```

2.8 SHUTTING DOWN THE SYSTEM

Most Linux administrators do not like to shut down their Linux servers. It spoils their uptime (“uptime” is a thing of pride for Linux system admins). Thus, when a Linux box has to be rebooted, it is usually for unavoidable reasons. Perhaps something bad has happened or the kernel has been upgraded.

It is important to follow the correct procedures when you shut down a Linux system. If you fail do so, your file systems probably will become trashed and the files probably will become scrambled. This is because Linux has a disk cache that won't write things to disk at once, but only at intervals. This greatly improves performance but also means that if you just turn off the power at a whim the cache may hold a lot of data and that what is on the disk may not be a fully working file system (because only some things have been written to the disk). Another reason against just flipping the power switch is that in a multi−tasking system there can be lots of things going on in the background, and shutting the power can be quite disastrous. By using the proper shutdown sequence, you ensure that all background processes can save their data.

The command for properly shutting down a Linux system is **shutdown**. The shutdown command stops all services, processes and daemons in a sequential and consistent fashion. It broadcasts the message to all logged in users and waits for one minute, by default, for users to log off, after which time it begins stopping services, processes and daemons. It unmounts file systems and proceeds as per the options specified at the command line.

If your system has many users, use the command *shutdown −h +time message*, where time is the time in minutes until the system is halted, and message is a short explanation of why the system is shutting down.

Rebooting means booting the system again. This can be accomplished by first shutting it down completely, turning power off, and then turning it back on. A simpler way is to ask shutdown to reboot the system, instead of merely halting it. This is accomplished by using the **−r** option to shutdown, for example, by giving the command *shutdown −r now*.

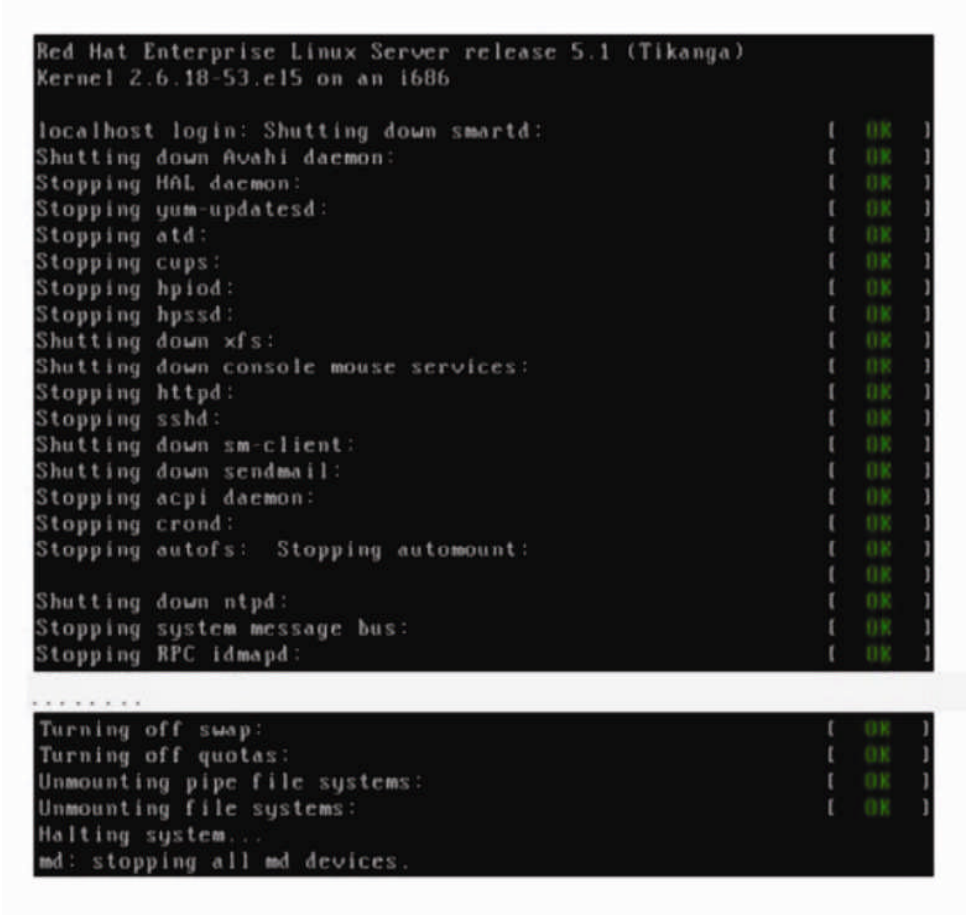


Figure 2-3: Messages when shutdown command is initiated

2.9 SUMMARY

- By knowing the booting process in depth, you can make any desired changes to the configuration files and gain total control over the functionality of your system. You will also be able to easily find problems that may be keeping your system from booting properly and quickly correct them.
- The boot process can be summarized as follows:

POST → BIOS → Active Partition → MBR → BOOT Loader → Kernel → initrd → init → insertion of kernel modules to support the most essential hardware needed for booting → mounting of root file system from the secondary storage → /etc directory → init in /sbin → fstab → initab → rc.sysinit → init levels → rcN.d → Naming convention (K/S),2 digit integer → chronological sequence (N in rcN.d stands for run level bet 0 and 6)

- The boot loader is the first software program that runs when a computer starts. It is responsible for handing over control of the system to the operating system.

- The main choices that come with Linux distributions are GRUB (Grand Unified Bootloader) and LILO (Linux Loader).
- GRUB is the default boot loader for Fedora, Red Hat Enterprise Linux (RHEL), OpenSUSE, Mandrake, Ubuntu, and a host of other Linux distributions.
- The GRUB boot process happens in stages. Each stage is taken care of by special GRUB image files, with each preceding stage helping the next stage along. Two of the stages are essential, and any of the other stages are optional and dependent on the particular system setup.
- LILO, short for Linux Loader, is a boot manager. It allows you to boot multiple operating systems, provided each system exists on its own partition.
- The process of bootstrapping the operating system is divided into - kernel loading and kernel execution.
- The init process is the first non-kernel process that is started, and, therefore, it always gets the process ID number of 1. The /sbin/init program is the first system process that runs after the kernel has configured the system devices and mounted the system drives.
- The /etc/inittab file specifies which scripts to run when runlevels change. These scripts are responsible for either starting or stopping the services that are particular to the runlevel. Because of the number of services that need to be managed, rc scripts are used.
- The command for properly shutting down a Linux system is shutdown. The shutdown command stops all services, processes and daemons in a sequential and consistent fashion. It broadcasts the message to all logged in users and waits for one minute, by default, for users to log off, after which time it begins stopping services, processes and daemons. It unmounts file systems and proceeds as per the options specified at the command line.

2.10 REVIEW QUESTIONS

i. Explain the boot process in detail?

ii. What is a boot loader? Explain LILO in brief?

iii. Explain GRUB in detail?

iv. Explain the process of bootstrapping?

v. Explain the init process, runlevels and rc scripts?

vi. How can a service be enabled or disabled?

2.11 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- Beginning Linux Programming 4th Edition by Neil Mathew, Richard Stone. Wiley Publishing
- Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill
- Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill
- Red Hat Linux Networking and System Administration 3rd Edition by Terry Collins and Kurt Wall.
- Sybex – RHCE Red Hat Certified Engineer Study Guide
- Red Hat Certified Technician & Engineer by Asghar Ghori.
- www.thegeekstuff.com
- www.tlpd.org
- www.linuxtopia.org

3

THE FILE SYSTEM

Unit Structure :

3.0 OBJECTIVES

In this chapter, we try to understand the file system hierarchy standard, the file system structure that Linux follows and the various file systems that Linux supports. This chapter explains memory and virtual file systems and ends with a discussion on Linux disk management.

3.1 INTRODUCTION

Understanding the organization, or layout, of the file system is one of the most important aspects of system administration. For administrators, programmers, users, and installed software, knowing how and where the files are stored on the system is critical for proper system operation. A standard should be in place that specifies locations for specific types of data. Linux files are organized in a logical fashion for ease of administration. This logical division of files in maintained in hundreds of directories that are located in larger containers called file systems. Fortunately, Red Hat has chosen to follow the standards outlined in the Filesystem Hierarchy Standard (FHS). There are two types of file systems – disk-based and memory-based. Disk-based file systems are created on physical media such as a hard disk and memory-based file systems, also called virtual file systems, are created at system boot up and destroyed at shut down.

A **file system** is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk. The word is also used to refer to a partition or disk that is used to store the files or the type of the file system.

In simpler terms, the file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent on how the operating system organizes files on storage devices. Providing a common file system structure ensures users and programs can access and write files.

3.2 FILE SYSTEM HIERARCHY STANDARD

Most UNIX file system types have a similar general structure, although the exact details vary quite a bit. The central concepts are super block, inode, data block, directory block, and indirection

block. The **super block** contains information about the file system as a whole, such as its size (the exact information here depends on the file system). An **inode** contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A **directory** entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file. There is space only for a few data block numbers in the inode, however, and if more are needed, more space for pointers to the data blocks is allocated dynamically. These dynamically allocated blocks are **indirect blocks**; the name indicates that in order to find the data block, one has to find its number in the indirect block first. Like UNIX, Linux chooses to have a single hierarchical directory structure. Everything starts from the root directory, represented by /, and then expands into sub−directories instead of having so−called 'drives'.

In the Windows environment, one may put one's files almost anywhere: on C drive, D drive, E drive etc. Such a file system is called a hierarchical structure and is managed by the programs themselves (program directories), not by the operating system. On the other hand, Linux sorts directories descending from the root directory, according to their importance to the boot process. If you're wondering why Linux uses the frontslash / instead of the backslash \ as in Windows it's because it's simply following the UNIX tradition. If you install a program in Windows, it usually stores most of its files in its own directory structure. A help file for instance may be in C:\Program Files\[program name]\ or in C:\Program Files\[program−name]\help or in C:\Program Files\ [program−name] \humpty\dumpty\doo. In Linux, programs put their documentation into /usr/share/doc/[program−name], man(ual) pages into /usr/ share/ man /man[1−9] and info pages into /usr/share/info. They are merged into and with the system hierarchy.

The FHS provides specific requirements for the placement of files in the directory structure. Placement is based on the type of information contained in the file. Two categories of file information exist:

- shareable or unshareable files,
- variable or static files.

Shareable files are files that can be accessed by other hosts, and **unshareable** files can be accessed only by the local system. **Variable** files contain information that can change at any time on their own, without anyone actually changing the file. A log file is an example of such a file. A **static** file contains information that does not change unless a user changes it. Program documentation and binary files are examples of static files.

Categorizing files in this manner helps correlate the function

of each file with the permissions assigned to the directories which hold them. How the operating system and its users interact with a file determines the directory in which it is placed, whether that directory is mounted with read-only or read/write permissions, and the level of access each user has to that file. The top level of this organization is crucial; access to the underlying directories can be restricted, otherwise security problems could arise if, from the top level down, access rules do not adhere to a rigid structure. This chapter is not as detailed as the FHS. A system administrator should also read the full FHS for a complete understanding.

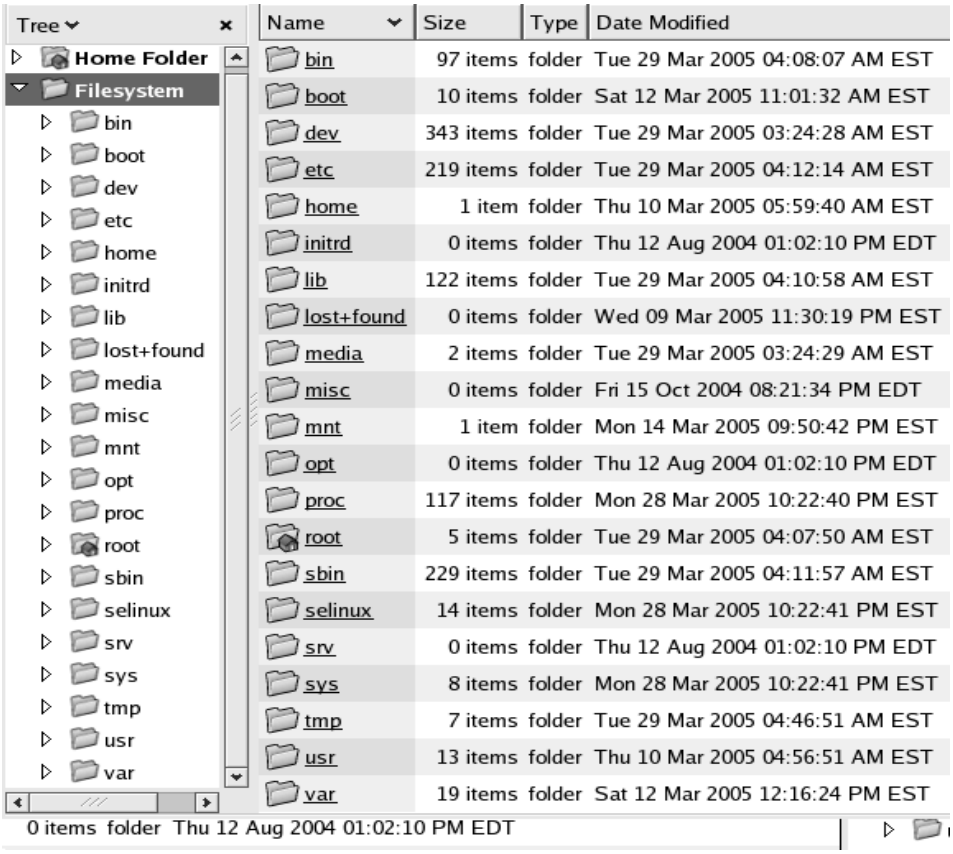


Figure 3-1: The file system organization for a typical Fedora and RHEL systems

3.3 Understanding the File System Structure

Figure 3-1 shows the organization of the file system on a typical Fedora Core and Red Hat Enterprise Linux system. As shown in the illustration, the file system is organized in a flat, hierarchical file system. Linux's method of mounting its file systems in a flat, logical, hierarchical method has advantages over the file system mounting method used by Windows. Linux references everything relative to the root file system point /, whereas Windows has a different root mount point for every drive. Linux file system

starts with /, the root directory. All other directories are 'children' of this directory. The partition which the root file system resides on is mounted first during boot and the system will not boot if it doesn't find it.

The Linux file system structure is like an inverted tree with the root of the tree at the top and branches and leaves at the bottom. The top-level is referred to as root and represented by the forward slash (/) character. This is the point where the entire file system structure is ultimately connected to. Two file systems – / and /boot are created, by default, when Linux is installed. The main directories under / and other file systems are shown in figure 3-2 as given below. Some of these directories hold static data while others contain dynamic (or variable) information. The static data refers to file contents that are not usually modified. The dynamic or variable data refers to file contents that are modified as required. Static directories normally contain commands, library routines, kernel files, device files etc. and dynamic directories hold log files, status files, configuration files, temporary files, etc.

A brief description of disk-based and virtual-based file systems is provided in the following sub-sections.

3.3.1 The Root File System (/) – Disk-based

The / directory is called the *root* directory and is the top-level file system in the FHS and contains many higher-level directories holding specific information. The primary purpose of the / directory is booting the system and correcting any problems that might be preventing the system from booting. According to the FHS, the / directory must contain, or have links to, the following directories:

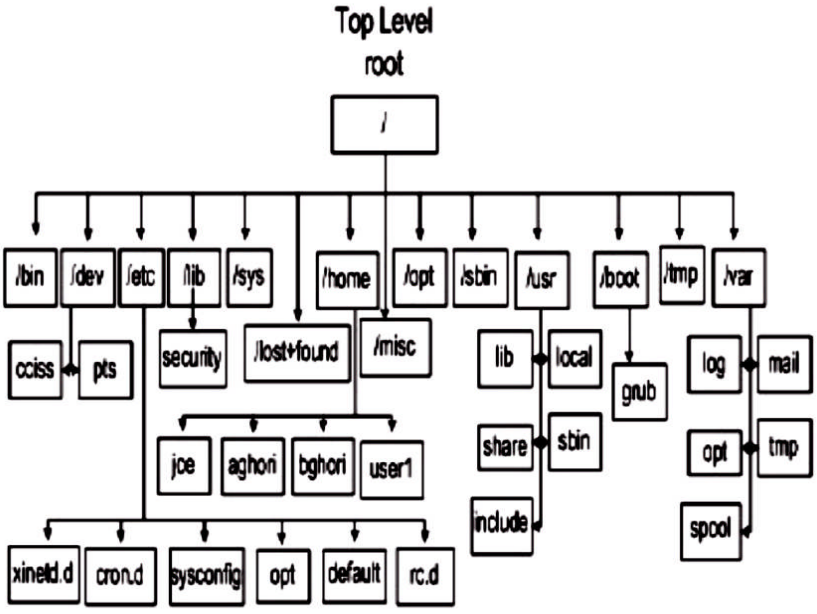


Figure 3-2: File System Tree

The Binary Directory (/bin)

The *binary* directory contains crucial user executable commands. This directory cannot contain sub-directories. This directory holds static data files.

The Library Directory (/lib)

The *library* directory contains shared library files required by programs. It contains sub-directories that hold library routines. The */lib* directory holds static data files.

The System Binary Directory (/sbin)

Most commands required at system boot up are located in the *system binary* directory. In addition, most commands requiring root privileges to run are also located here. In other words, this directory contains crucial system administration commands that are not intended for regular users. This directory is not included in normal users' default search path because of the nature of commands it contains. The */sbin* directory holds static files.

The Etcetera Directory (/etc)

The *etcetera* directory holds most system configuration files. Some of the more common sub-directories under */etc* are: *sysconfig, default, opt, cups, lvm, xinetd.d, mail, rc.d, skel, kde* and *gnome.* These sub-directories contain, in that sequence, configuratin files for various system services, user account defaults, additional software installed on the system, printers, Logical Volume Manager, internet services, mail subsystem, system startup and shutdown scripts, user profile templates and the last two hold KDE and GNOME desktop configuration files. The */etc* directory contains dynamic data files.

The lost + found Directory (/lost + found)

This directory is used to hold files that become orphan after a system crash. An *orphan* file is a file that has lost its name. This directory is automatically created in a file system when the file system is created, and holds dynamic information.

The /root Directory

This is the default home directory for the *root* user. Don't confuse this with the / directory, which has the same name.

The /srv Directory

This directory holds server or site-specific data associated with databases, websites, etc.

The Boot File System (/boot) – Disk-based

The */boot* file system contains Linux kernel(s), boot loader(s) and boot configuration file(s) in addition to other files required to boot. The default size of this file system is 100MB, and is altered

only when an update to the kernel is performed. The */boot* file system contains static data files.

The Variable File System (/var) – Disk-Based

/var contains data that frequently change while the system is up and running. Files olding log, status, spool and other dynamic data are typically located in this file system. Some common sub-directories under */var* are briefly discussed below:

The /var/log Directory

Most system log files are located here. This directory contains system logs, boot logs, failed user logs, user logs, installation logs, cron logs, mail logs, etc.

The /var/spool/mail Directory

This is the location for user mailboxes.

The /var/opt Directory

For additional software installed in */opt*, this directory contains log, status and other variable data files for that software.

The /var/spool Directory

Directories that hold print jobs, cron jobs, email messages and other queued items before being sent out, are located here.

The /var/tmp Directory

Large temporary files or temporary files that need to exist for extended periods of time than what is allowed in */tmp*, are stored here. These files survive system reboots and are not automatically deleted.

The UNIX System Resources File System (/usr) – Disk-Based

This file system contains general files related to the system. Some of the more important sub-directories under */usr* are briefly discussed below:

The /usr/bin Directory

Contains additional user executable commands.

The /usr/sbin Directory

Contains additional system administration commands.

The /usr/local Directory

System administrator repository to keep commands and tools that they download from the web, develop in-house or obtain elsewhere. These commands and tools are not generally included with original Linux software distribution. In particular, */usr/local/bin* holds executable files, */usr/local/etc* contains their configuration files and */usr/local/man* holds associated man pages.

The /usr/include Directory

Contains header files for the *C* language.

The /usr/share Directory

Directory location for man pages, documentation, sample templates, configuration files, etc. that may be shared on multi-vendor Linux/UNIX platforms with heterogeneous hardware architectures.

The /usr/lib Directory

Contains library files pertaining to programming sub-routines.

The Temporary File System (/tmp) – Disk-Based

This file system is a repository for temporary files. Many programs create temporary files as they run or being installed. Some programs delete temporary files that they create after they are finished, while other do not.

The Optional File System (/opt) – Disk-Based

This file system holds additional software packages installed on the system. A sub-directory is created for each installed software.

The Home File System (/home) – Disk-Based

The */home* file system is designed to hold user *home* directories. Each user account is assigned a home directory for storing personal files. Each home directory is owned by the user the directory is assigned to. No other user usually has access to other users' home directories.

The Devices File System (/dev) – Virtual

The */dev* (devices) file system contains device files for hardware and virtual devices. Linux kernel communicates with system hardware and virtual devices through corresponding device files located in here. There are two types of device files: *character* special device files (a.k.a. *raw* device files) and *block* special device files. The kernel accesses devicesusing one or both types of device files.

Character devices are accessed in a serial manner where streams of bits are transferred during kernel and device

communication. Examples of such devices are serial printers, mouse, keyboard, terminals, floppy disks, hard disk devices, tape drives, etc. Block devices are accessed in a parallel fashion meaning that data is transferred between the kernel and the device in blocks (parallel) when communication between the two takes place. Examples of block devices are hard disk devices, CD/DVD drives, floppy disks, parallel printers, etc.

Some key directories under */dev* are *disk, pts* and *VolGroup00*, and contain device files for hard disks (*disk*), pseudo terminals *(pts)* and root volume group (*VolGroup00*). The */dev* file system holds static data files.

The Media File System (/media) – Virtual

This virtual file system is used to automatically mount removable media such as floppy, CD, DVD, USB and Zip disks.

The Mount File System (/mnt) – Virtual

The */mnt* directory is reserved for temporarily mounted file systems, such as NFS file system mounts. For all removable storage media, the */media* directory is used.

The Process File System (/proc) – Virtual

Information about the current state of the running kernel is maintained in this file system. This information includes details on CPU, memory, partitioning, interrupts, I/O addresses, DMA channels and running processes, and is represented by various files. These files do not actually store information, rather, they point to the information in the memory. This file system is automatically maintained by the system. The */proc* file system contains dynamic data files.

The System File System (/sys) – Virtual

Information about the currently configured hardware is stored and maintained in this file system. This file system is automatically maintained by the system.

The SELinux File System (/selinux) – Virtual

If SELinux packages are installed, this file system stores all current settings for SELinux.

3.4 WORKING WITH STANDARD LINUX-SUPPORTED FILESYSTEMS

Linux is a very flexible operating system that has a long history of interoperability with other systems on a number of different hardware platforms. A consequence of this friendliness to other operating systems is that Linux can read and write to several different file systems that originated with other operating systems

much different from Linux. This section details the different file systems supported and where they originated. One reason that Linux supports so many file systems is the design of its Virtual File Systems (VFS) layer. The VFS layer is a data abstraction layer between the kernel and the programs in userspace that issue file system commands. The VFS layer avoids duplication of common code between all file systems. It provides a fairly universal backward compatible method for programs to access all of the different forms of file support. Only one common, small API set accesses each of the file system types, to simplify programming file system support.

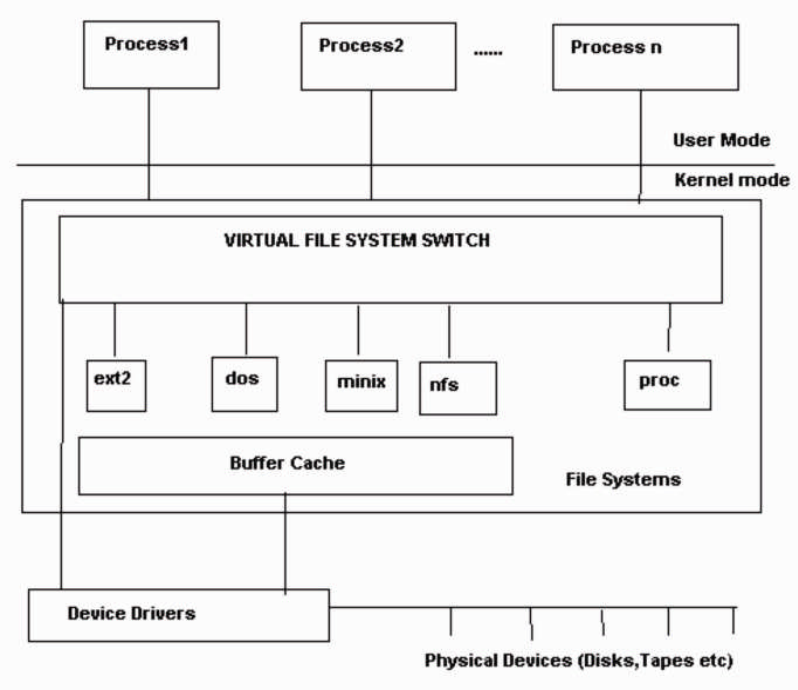


Figure 3-3: Virtual File System

3.4.1 Ext2 (Second Extended File System)

The *Second Extended File System* is probably the most widely used filesystem in the Linux community. It provides standard Unix file semantics and advanced features. Moreover, thanks to the optimizations included in the kernel code, it is robust and offers excellent performance. *Ext2fs* was first developed and integrated in the Linux kernel and is now actively being ported to other operating systems. Ext2fs is able to manage filesystems created on really big partitions. While the original kernel code restricted the maximal filesystem size to 2 GB, recent work in the VFS layer have raised this limit to 4 TB. Thus, it is now possible to use big disks without the need of creating many partitions. Ext2fs reserves some blocks for the super user (root). Normally, 5% of the blocks are reserved. This allows the administrator to recover easily from situations where user processes fill up filesystems. In Linux, the Ext2fs kernel code contains many performance optimizations, which tend to

improve I/O speed when reading and writing files. Users can take advantage of new features without reformatting their old ext2 file systems. ext2 has the added bonus of being designed to be POSIX-compliant. New features that are still in the development phase are access control lists, undelete, and on-the-fly compression.

3.4.2 Ext3 (Third Extended File System)

The *extended 3 file system* is a new file system introduced in Red Hat 7.2. *ext3* provides all the features of ext2, and also features journaling and backward compatibility with ext2. The backward compatibility enables you to still run kernels that are only ext2-aware with ext3 partitions. You can also use all of the ext2 file system tuning, repair, and recovery tools with ext3. You can upgrade an ext2 file system to an ext3 file system without losing any of your data. This upgrade can be done during an update to the operating system. After an unexpected power failure or system crash (also called an unclean system shutdown), each mounted ext2 file system on the machine must be checked for consistency. This takes a long time on large file systems. The journaling provided by the ext3 file system means that this sort of file system check is no longer necessary after an unclean system shutdown. On an ext3 system, the system keeps a record of uncommitted file transactions and applies only those transactions when the system is brought back up. So, a complete file system check is not required, and the system will come back up much faster. The only time a consistency check occurs using ext3 is in certain rare hardware failure cases, such as hard drive failures. A cleanly unmounted ext3 file system can be mounted and used as an ext2 file system. This capability (backward compatibility) can come in handy if you need to revert to an older kernel that is not aware of ext3. The kernel sees the ext3 file system as an ext2 file system.

3.4.3 Reiserfs

The *Reiser* file system is a journaling file system designed for fast server performance, especially in directories containing thousands of files. Reiserfs is a file system using a variant on classical balanced tree algorithms. The results when compared to the ext2fs conventional block allocation based file system running under the same operating system and employing the same buffering code suggest that these algorithms are more effective for files. reiserfs also does not have fixed space allocation for inodes, which saves about 6 percent of your disk space.

3.4.4 SystemV

Linux currently provides read support for *SystemV* partitions, and write support is experimental. The SystemV file system driver currently supports AFS/EAFS/EFS, Coherent FS, SystemV/386 FS, Version 7 FS, and Xenix file systems.

3.4.5 UFS

UFS is used in Solaris and early BSD operating systems. Linux provides read support, and write support is experimental.

3.4.6 FAT

FAT is one of a few different file systems used with Windows over the years. Almost every computer user has used FAT at one time or another, since it was the sparse base operating system at the heart of all Windows operating systems. FAT was originally created for QDOS and used on 360K (double density, double-sided) floppy disks. Its address space has since been extended from 12 bit to 32 bit, so it can handle very large file systems. There have been four versions of FAT since its beginnings: FAT12, FAT16, VFAT, and FAT32. Nowadays, it’s possible to create FAT32 file systems over a terabyte in size.

3.4.7 NTFS (New Technology File System)

NTFS is the next generation of HPFS. It comes with all versions of Microsoft operating systems beginning with Windows NT. Unlike FAT, it is a b-tree file system, meaning it has a performance and reliability advantage, including journaling, and support for encryption and compression, over FAT. It makes NTFS drives appear indistinguishable from standard FAT drives, providing the ability to navigate, view and execute programs on them.

3.4.8 IBM JFS

JFS is IBM's journaled file system technology, currently used in IBM enterprise servers, and is designed for high−throughput server environments. Linux support for JFS was written by IBM. IBM has contributed quite a bit of code to the Linux cause and is a staunch supporter of Linux.

3.4.9 SGI XFS

XFS is the next−generation file system for Silicon Graphics[TM] systems, from desktop workstations to supercomputers. XFS provides full 64−bit file capabilities that scale easily to handle extremely large files and file systems that grow to million terabytes. The XFS file system integrates access control lists, volume management, guaranteed rate I/O, and journaling technology for fast, reliable recovery. File systems can be backed up while still in use, significantly reducing administrative overhead. It has been available for use on Linux since May 2001.

3.5 WORKING WITH NON-STANDARD LINUX-SUPPORTED FILESYSTEMS

Support for these file systems needs to be explicitly compiled into the Linux kernel, since kernel support for them is not configured by default.

3.5.1 FREEVxFS

This is a commercial filesystem developed by Veritas Inc. You can see it in HP−UX, SCO UnixWare, Solaris and probably other systems. It has very interesting features: Extent based allocation, Journaling, access control lists (ACLs), upto 2 terabyte large file support, online backup (snapshot filesystem), BSD style quotas and many more. Three VxFS versions are available:

Version 1: This is original VxFS, not commonly in use.

Version 2: Support for filesets and dynamic inode allocation.

Version 4: Latest version, supports large files and quotas.

GNU utilities available for Linux called VxTools can read VxFS versions 2 and 4. The tools included in the VxTools package are vxmount, vxumount, vxls, vxcat, vxidump, vxcd, and vxpwd. Currently there is only read support in Linux for VxFS file systems.

3.5.2 GFS

GFS is Sistina's Global File System. It is a clustered journaling file system for SANs that enables multiple servers to have read/write access to a single file system on shared SAN devices. GFS is scalable, since storage devices and servers can be added without taking the system down or taking the disks offline. It also makes a single image of all the data in the SAN, so that if a server fails it can be removed and replaced while the load is rebalanced amongst the remaining servers. In a proper cluster setup, all nodes in the cluster share the same storage devices through a fiber channel, SCSI hookup, or network block device. Each node sees the file system as being local to their machine, and GFS synchronizes files across the cluster. GFS is fully symmetric, so no server is a bottleneck or single point of failure. GFS uses regular UNIX-style file semantics.

3.6 MEMORY AND VIRTUAL FILESYSTEMS

These file systems do not exist on disk in the same way that traditional file systems do. They either exist entirely in system memory or they are virtual, because they are an interface to system devices, for example.

3.6.1 CRAMFS

cramfs is designed to cram a file system onto a small flash memory device, so it is small, simple, and able to compress things well. The largest file size is 16 MB, and the largest file system size is 256 MB. Since cramfs is so compressed, it isn't instantly updateable. The mkcramfs tool needs to be run to create or update a cramfs disk image. The image is created by compressing files one page at a time, so this enables random page access. The metadata is not compressed, but it has been optimized to take up much less space than other file systems. For example, only the low 8 bits of the GID are stored. This saves space but also presents a potential security issue.

3.6.2 TMPFS

tmpfs is structured around the idea that whatever is put in the /tmp file system is accessed again shortly. tmpfs exists solely in memory, so what you put in /tmp doesn't persist between reboots. Mounting a special-purpose file system on /tmp as an in-memory file system is a performance boost but is rarely done in Linux because of the performance available from the traditional Linux file system. But for those who feel that they need the performance gains from storing /tmp in memory, this option is now available in Linux.

3.6.3 RAMFS

ramfs is basically cramfs without the compression.

3.6.4 ROMFS

This is a *read-only file system* that is mostly used for the initial ramdisks of installation disks. It was designed to take up very little space, so you could fit a kernel and some useful code into a small boot disk, without having the file system overhead taking up too much precious space in memory or on the disk. The kernel on the disk has only this file system linked into it, and it can load any modules it needs later, after bootup. After the kernel is loaded, it can call other programs to help determine what SCSI drivers are needed, if any, or what IDE or floppy drives should be accessed after bootup. This method is perfect for rescue diskettes or installation diskettes, where only a very bare minimum kernel needs to be loaded into memory, so after the initial boot it can then load from a CD-ROM whatever ext2 modules or other drivers are necessary to mount the system's regular drives. The romfs file system is created with a program called genromfs.

3.6.5 PROC

Unlike most file systems, */proc* contains neither text nor binary files. Instead, it houses virtual files; hence, /proc is normally referred to as a virtual file system. These virtual files are typically zero bytes in size, even if they contain a large amount of

information. The /proc file system is not used for storage per se. Its main purpose is to provide a file-based interface to hardware, memory, running processes, and other system components. Its contents are created at system boot and destroyed when the system is shut off. You can retrieve real-time information on many system components by viewing the corresponding /proc file. Some of the files within /proc can also be manipulated (by both users and applications) to configure the kernel. The files and sub-directories in the /proc filesystem contain hardware and software information.

/proc/1

A directory with information about process number 1. Each process has a directory below /proc with the name being its process identification number.

/proc/cpuinfo

Information about the processor, such as its type, make, model, and performance.

/proc/devices

List of device drivers configured into the currently running kernel.

/proc/dma

Shows which DMA channels are being used at the moment.

/proc/filesystems

Filesystems configured into the kernel.

/proc/interrupts

Shows which interrupts are in use, and how many of each there have been.

/proc/ioports

Which I/O ports are in use at the moment.

/proc/kcore

An image of the physical memory of the system. This is exactly the same size as your physical memory, but does not really take up that much memory; it is generated on the fly as programs access it. (Remember: unless you copy it elsewhere, nothing under /proc takes up any disk space at all.)

/proc/kmsg

Messages output by the kernel. These are also routed to syslog.

/proc/ksyms

Symbol table for the kernel.

/proc/loadavg

The `load average' of the system; three meaningless indicators of how much work the system has to do at the moment.

/proc/meminfo

Information about memory usage, both physical and swap.

/proc/modules

Which kernel modules are loaded at the moment.

/proc/net

Status information about network protocols.

/proc/self

A symbolic link to the process directory of the program that is looking at /proc. When two processes look at /proc, they get different links. This is mainly a convenience to make it easier for programs to get at their process directory.

/proc/stat

Various statistics about the system, such as the number of page faults since the system was booted.

/proc/uptime

The time the system has been up.

/proc/version

The kernel version.

3.6.6 /dev/pts

/dev/pts is a lightweight version of devfs. Instead of having all the device files supported in the virtual file system, it provides support for only virtual pseudoterminal device files. /dev/pts was implemented before devfs.

3.6.7 devfs

The *Device File System (devfs)* is another way to access “real” character and block special devices on your root file system. The old way used major and minor numbers to register devices. devfs enables device drivers to register devices by name instead. devfs is deprecated in the 2.6 kernel in favor of udev.

3.6.8 sysfs

sysfs is a virtual file system that acts as an interface to the kernel’s internal data structures. Information is stored in the /sys directory and can be used to get details about a system’s hardware and to change kernel parameters at runtime. Information in the /sys directory is similar to the information provided in the /proc directory and can be accessed in a similar fashion.

3.7 LINUX DISK MANAGEMENT

This section explains some basics about disk partitioning and disk management under Linux. To see how your Linux disks are currently partitioned and what file systems are on them, look at the */etc/fstab* file. In Figure 3-4, you can see what a simple /etc/fstab file looks like.

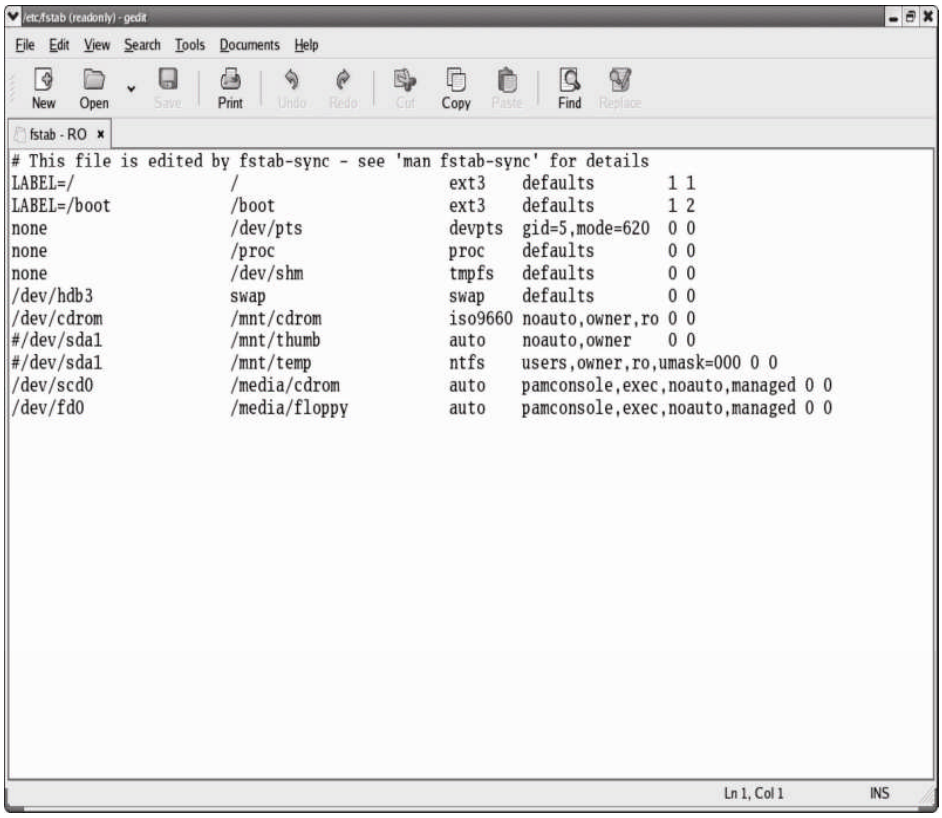


Figure 3-4: The contents of the /etc/fstab file.

3.7.1 Disk Partitioning on an x86 Machine

When disk partitioning on an x86 PC, you need to be mindful of the limitations present in the x86 architecture. You are allowed to create four primary partitions. Primary partitions are the only partitions that are bootable. You can create more partitions if you make extended partitions. Extended partitions are set into a primary partition. So, if you choose to make extended partitions, you are allowed to make only three primary partitions for operating system use, and the fourth partition is dedicated to hosting the extended partitions.

3.7.2 Mounting Other OS Partitions/Slices

Not only can Linux read other operating systems' file systems; it can mount disk drives from other systems and work with their partition tables. However, it is necessary to compile two options into the kernel to do this. You must have the file system support and the file partitioning support turned on in the kernel.

Usually file system support is compiled as a module by default, but disk partition support usually has to be explicitly compiled. Some common partitioning schemes that Linux supports are x86 partitions, BSD disklabel, Solaris x86, Unixware, Alpha, OSF, SGI, and Sun. Mounting other operating systems' partitions is helpful if you need to put a Sun hard disk into a Linux machine, for example. You may need to do this if the original Sun system has gone bad, and you need to recover the information that was on its disk, or if it's the target of a forensic computer crime investigation, and you need to copy the disk contents to another machine to preserve evidence. This method takes advantage of the fact that copying a large amount of data is much faster across a SCSI connection than across a network. If you need to copy a large amount of raw disk data across a network, you can use the Network Block Device, which enables other machines to mount a disk on your machine as if it were on their machine.

3.7.3 Metadevices

Virtual block devices that are made up of other block devices are referred to as a metadevice. An example of a metadevice is a disk array that makes many disks look like one large disk. When a disk that's mounted as a regular block device dies, then the data on it becomes unavailable. If a disk dies in a metadevice, the metadevice is still up. As long as the criteria are met for the minimum number of working devices in the metadevice, the metadevice still functions.

3.7.4 Logical Volumes

Logical Volume Manager (LVM) enables you to be much more flexible with your disk usage than you can be with conventional old-style file partitions. Normally if you create a partition, you have to keep the partition at that size indefinitely. For example, if your system logs have grown immensely, and you've run out of room on your /var partition, increasing a partition size without LVM is a big pain. You would have to get another disk drive, create a /var mount point on there too, and copy all your data from the old /var to the new /var disk location. With LVM in place, you could add another disk, create a physical volume, and then add the physical volume to the volume group that contains the /var partition. Then you'd use the LVM file system resizing tool to increase the file system size to match the new partition size. Normally, you might think of disk drives as independent entities, each containing some data space. When you use LVMs, you need a new way of thinking about disk space. First, you have to understand that space on any disk can be used by any file system. A Volume Group is the term used to describe various disk spaces (either whole disks or parts of disks) that have been grouped together into one volume. It works in the following manner. First you need to have a physical volume which is then divided into Volume groups that are then combined to

form logical volumes. Logical volumes are akin to the historic idea of partitions. You can then use a file system creation tool such as fdisk to create a file system on the logical volume. The Linux kernel sees a logical volume in the same way it sees a regular partition. When the system is installed, LVM is enabled by default and you will need to use the LVM tools to make changes to your logical volumes. You can, if you desire, choose not to use logical volumes during the system installation. The basic syntax for using the lvm command is:

lvm <command> file

There are many commands available when using LVM. You can obtain a complete listing of the commands by entering lvm help at a command prompt.

3.7.5 RAID

RAID is an acronym for Redundant Array of Inexpensive, or Independent Disks. There are two types of RAID that can be used on computer systems. These types are hardware RAID and software RAID. In addition, there are six different RAID levels commonly used regardless of whether hardware or software RAID is used. A brief explanation of hardware and software RAID is in order. Following this explanation is a description of the RAID levels.
Hardware Raid – In hardware RAID the disks have their own RAID controller with built-in software that handles the RAID disk setup, and I/O. The controller is typically a card in one of the system's expansion slots, or it may be built onto the system board. The hard RAID interface is transparent to Linux, so the hardware RAID disk array looks like one giant disk. The operating system does not control the RAID level used, it is controlled by the hardware RAID controller. Most dedicated servers use a hardware RAID controller.
Software RAID – In software RAID there is no RAID controller card. The operating system is used to set up a logical array, and the operating system controls the RAID level used by the system. Software RAID must be configured during system installation.

As mentioned earlier, there are six RAID levels that can be used, but in actual practice usually only three of them are used. And of these three, one doesn't provide redundancy even though it is identified as a RAID level. The three most commonly used RAID levels are:

RAID level 0 — This RAID level requires at least two disks and uses a method called striping that writes data across both drives. There is no redundancy provided by this level of RAID, since the loss of either drive makes it impossible to recover the data. This level of RAID does give a speed increase in writing to the disks.

RAID level 1 — This RAID level requires at least two disks and uses a method called mirroring. With mirroring, the data is written to both of the drives. So, each drive is an exact mirror of the other one, and if one fails the other still holds all the data. There are two variants to level 1 with one variant using a single disk controller that writes to both disks as described above. The other variant uses two disk controllers, one for each disk. This variant of RAID level 1 is known as duplexing.

RAID level 5 — This RAID level, which is the most widely used, requires at least three disks and uses striping to write the data across the two disks similarly to RAID level 1. But unlike RAID level 1, this level of RAID uses the third disk to hold parity information that can be used to reconstruct the data from either, but not both, of the two disks after a single disk failure.

The commands discussed here are only useful when using software RAID. Hardware RAID is invisible to the operating system. There are some system files that you can use to get information about RAID on your system. You can look in /etc/raidtab to get information about the system's RAID configuration. RAID devices are identified in Fedora Core and Enterprise Linux as *md* devices. The */etc/raidtab* file lists which block devices are associated with the md device. You can also look at the contents of the */proc/mdstat* file to get information about the running status of your md devices. Also available to you are several command-line tools. You can use lsraid to list and query md devices as well. This command is similar to the ls command and more information is available by reading the *lsraid* man page. You can also use the man command with the following RAID commands:

raidstart — This command will start an existing RAID device.

raidstop — This command will stop an existing RAID device.

raidreconf — This command is used to add disks to an existing array or to convert an array to a new type.

3.8 SUMMARY

- The file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent on how the operating system organizes files on storage devices. Providing a common file system structure ensures users and programs can access and write files.
- The FHS provides specific requirements for the placement of files in the directory structure. Placement is based on the type of

information contained in the file. Two categories of file information exist:

- shareable or unshareable files,
- variable or static files.

- Linux's method of mounting its file systems in a flat, logical, hierarchical method has advantages over the file system mounting method used by Windows. Linux references everything relative to the root file system point /, whereas Windows has a different root mount point for every drive. Linux file system starts with /, the root directory.
- The Linux file system structure is like an inverted tree with the root of the tree at the top and branches and leaves at the bottom. The top-level is referred to as root and represented by the forward slash (/) character.
- The primary purpose of the / directory is booting the system and correcting any problems that might be preventing the system from booting.
- According to the FHS, the / directory must contain, or have links to, the following directories:
  - bin
  - lib
  - sbin
  - etc
  - lost+found
  - root
  - srv
  - boot
  - var
  - usr
  - tmp
  - opt
  - home
  - dev
  - media
  - mnt
  - proc
  - sys
  - selinux

- Linux is a very flexible operating system that has a long history of interoperability with other systems on a number of different hardware platforms.
- Linux supports so many file systems because of the design of its Virtual File Systems (VFS) layer. The VFS layer is a data abstraction layer between the kernel and the programs in userspace that issue file system commands.
- Linux supports different disk and memory/virtual file systems - ext2, ext3, reiserfs, systemV, ufs, fat, ntfs, jfs, xfs, freevxfs, gfs, cramfs, tmpfs, ramfs, romfs, proc, devfs, sysfs.
- Logical Volume Manager (LVM) enables you to be much more flexible with your disk usage than you can be with conventional old-style file partitions.
- A Volume Group is the term used to describe various disk spaces (either whole disks or parts of disks) that have been grouped together into one volume.
- RAID is an acronym for Redundant Array of Inexpensive, or Independent Disks.
- In hardware RAID the disks have their own RAID controller with built-in software that handles the RAID disk setup, and I/O.
- In software RAID there is no RAID controller card. The operating system is used to set up a logical array, and the operating system controls the RAID level used by the system.
- The following commands can be used with software raid - lsraid, raidstart, raidstop, raidreconf.

3.9 REVIEW QUESTIONS

(1) Write a short note on Filesystem Hierarchy Standard.

(2) Explain the file system structure of Linux in brief?

(3) Write a short note on linux-supported filesystems.

(4) Write a short note on /proc file system.

(5) Write a short note on logical volumes.

(6) Write a short note on RAID.

3.10 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- Beginning Linux Programming 4th Edition by Neil Mathew, Richard Stone. Wiley Publishing
- Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill

- Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill
- Red Hat Linux Networking and System Administration 3rd Edition by Terry Collins and Kurt Wall.
- Sybex – RHCE Red Hat Certified Engineer Study Guide
- Red Hat Certified Technician & Engineer by Asghar Ghori.
- www.thegeekstuff.com
- www.tlpd.org
- www.linuxtopia.org

4

EXAMINING SYSTEM CONFIGURATION FILES

Unit Structure :

4.0 OBJECTIVES

In this chapter, we try to examine the various system and network configuration files that the Linux system administrator needs to configure after installing Linux. Also we examine the various files in the /etc/sysconfig directory along with managing the init scripts.

4.1 INTRODUCTION

Linux is designed to serve many users at the same time, providing an interface between the users and the system with its

resources, services, and devices. Users have their own shells through which they interact with the operating system, but you may need to configure the operating system itself in different ways.

The system configuration files in the /etc directory are the first place a system administrator goes after installing a system to set it up. The /etc directory is probably the most often visited directory by a system administrator after his or her own home directory and /var/log. All of the systemwide important configuration files are found either in /etc or in one of its many subdirectories. An advantage to keeping all system configuration files under /etc is that it's easier to restore configurations for individual programs, as opposed to having all the system's configurations rolled up into a monstrous registry hive as some operating systems do.

Because these files are so important and their contents so sensitive (everything from users' hashed passwords to the host's SSH key are stored in /etc), it is important to keep the file permissions set properly on everything in /etc. Almost all files should be owned by root, and nothing should be world-writable.

The /etc/sysconfig directory contains configuration scripts written and configured by Red Hat and Red Hat administration tools as well as files containing variable settings used by system startup scripts. /etc/sysconfig contains both system and networking configuration files. Putting these files in /etc/sysconfig distinguishes them from other /etc configuration files not designed by Red Hat.

The Red Hat system configuration files can fall within a few different functions. Some specify system duties, such as logging and automatically running programs with cron. Some set default configurations for important programs such as Sendmail and Bash. And many other system configuration files are responsible for arranging the appearance of the system, such as setting the colours that show up when a directory listing is shown and the banners that pop up when someone logs in.

4.2 SYSTEMWIDE SHELL CONFIGURATION SCRIPTS

These files determine the default environment settings of system shells and what functions are started every time a user launches a new shell. These configuration files affect all shells used on the system. An individual user can also set up a default configuration file in his or her home directory that affects only his or her shells. This ability is useful in case the user wants to add some extra directories to his or her path or some aliases that only he or she can use. When used in the home directory, the names are the same, except they have a . (period) in front of them. So /etc/bashrc

affects bash shells systemwide, but /home/host/.bashrc affects only the shells that the user *host* starts.

Shell Config Scripts: bashrc, csh.cshrc, zshrc
Bashrc is read by bash; csh.cshrc is read by tcsh; and zshrc is read by zsh. These files are read every time a shell is launched, not just upon login, and they determine the settings and behaviours of the shells on the system.

profile: This file is read by all shells except tcsh and csh upon login. bash falls back to reading it if there is no bash_profile. Zsh looks for zprofile, but if there is none, it reads profile as well. /etc/profile is a good place to set paths because it is where you set environmental variables that are passed to child processes in the shell. If you want to change the default path of your shells in /etc/profile, you can add another path statement in the path manipulation section of /etc/profile.

Do not add too many paths to this section because users can set their own paths using a .profile in their home directories. Adding more default paths than are necessary can pose a security risk. For example, a user named *katie* may want to run her own version of *pine*, which she keeps in her home directory. In that case, she may want to have /home/$USER or /home/katie at the beginning of her path so that when she types pine, the version in her home directory is found by the shell first, before finding the copy of pine in /usr/bin/pine. Generally, putting /home/$USER or any other directory whose contents are not controlled by root in /etc/profile is not a good idea.

The reason for this warning is that a rogue user or cracker can compile a backdoor, a way to enter the system unexpectedly, or corrupted version of a program and somehow get it in a user's home directory, perhaps even by mailing it to the user. If users' paths are set to check their home directories first, they may think that they are running a system program but instead are unknowingly running an alternate version. On the other hand, if this path modification is set only in katie's .profile, only she runs this risk. She should also be aware of this risk since she has to perform the extra step of adding this path modification herself.

Another useful variable to change in the system profile is the number of user commands saved in the .history file in the user's directory. This command history is especially useful, since you can scroll through your previous commands by using the up and down arrows. To change the number of commands saved in the .history file, modify this line:

HISTSIZE=1000

bash, tcsh, zsh, and Their Config File Read Orders
The shells read a few configuration files when starting up. It is good to know which files are read in what order, so that you know where to set variables that will only apply to certain users.

- bash — bash reads the following files on startup: /etc/profile, all the files in /etc/profile.d, ~/.bash_profile, ~/.bash_login, and ~/.profile. Upon logout, bash reads ~/.bash_logout.
- tcsh — tcsh reads the following files when starting up: /etc/csh.cshrc, then /etc/csh.login. After these come the config files in the user's home directory: ~/.tcshrc (or if not present, ~/.cshrc), ~/.history, ~/.login, ~/.cshdirs.
- zsh — zsh reads the following when starting up: /etc/zshenv, ~/.zshenv, /etc/zprofile, ~/.zprofile, /etc/zshrc, ~/.zshrc, and /etc/zlogin. Non-login shells also read ~/.bashrc. Upon logout, zsh reads the ~/.zlogout and /etc zlogout files.

4.3 SYSTEM ENVIRONMENTAL SETTINGS

The files discussed in this section deal with system environmental settings.

/etc/motd
This file contains the message that users see every time they log in. It's a good place to communicate messages about system downtime and other things that users should be aware of. On the other hand, you can put amusing quotes here to entertain your users. Usually, the motd contains a message like:

Welcome to Institute of Distance & Open Learning's LINUX system.
This system is monitored. Unauthorized use prohibited.
System downtime scheduled this Sunday night from 10 pm to 1 am.

motd is a plain-text file, which you can edit with any text editor. You can use it to display any message you want users to see when they login. If you don't have this file in your /etc directory you can easily create it.

NOTE: The message in this file is only visible to users who login using the console and not GUI.

issue
Whatever is in this file shows up as a prelogin banner on your console. By default, this file tells which version of Red Hat is running on the system and the kernel version. The default file looks like this:

Red Hat Linux release 7.2 (Enigma)
Kernel \r on an \m

So when you log in, you see this message (or something similar, depending on the kernel running on your system):

Red Hat Linux release 7.2 (Enigma)
Kernel 2.6.10-1.770_FC3 on an i686

issue.net

This file generally contains the same thing as */etc/issue*. It shows up when you attempt to telnet into the system. Because it shows up to people who are connecting to your system over the Internet, displaying a warning such as “Access is being monitored. Unauthorized access is prohibited” is good practice.

aliases

/etc/aliases is the email aliases file for the Sendmail program, and Postfix uses */etc/postfix/aliases*. By default, it contains many system account aliases. The aliases file sends mail for all the basic system accounts such as bin, daemon, and operator to root’s mailbox. Whenever you make changes to this file, you need to run the *newaliases* command to have the changes take affect in Sendmail.

fstab

fstab contains important information about your file systems, such as what file system type the partitions are, where they are located on the hard drive, and what mount point is used to access them. This information is read by vital programs such as *mount, umount,* and *fsck. mount* runs at start time and mounts all the file systems mentioned in the fstab file, except for those with noauto in their line. If a partition you want to access is not listed in this file, you have to mount it manually. This can get tedious, so it’s better to list all of your file systems in fstab. When *fsck* is run at bootup, it also checks all the file systems listed in fstab for consistency. It then fixes corrupted file systems, usually because they were not unmounted properly when the system crashed or suddenly lost power.

The *fstab* file has six fields, and each field represents a different configuration value. The first field describes the file system, which can be a partition name, the label of a disk partition, a logical volume, or a remote file system. The second field is the mount point used to access the file system. The third field describes the file system type. The fourth field is the place for any mount options you may need. The fifth field is 0 or 1 to determine whether dump backs up this file system. The final field sets the order in which fsck checks these file systems.

```
# This file is edited by fstab-sync - see 'man fstab-sync' for details
/dev/VolGroup00/LogVol00 /                       ext3    defaults        1 1
LABEL=/boot              /boot                   ext3    defaults        1 2
none                     /dev/pts                devpts  gid=5,mode=620  0 0
none                     /dev/shm                tmpfs   defaults        0 0
none                     /proc                   proc    defaults        0 0
none                     /sys                    sysfs   defaults        0 0
/dev/VolGroup00/LogVol01 swap                    swap    defaults        0 0
/dev/hdc                 /media/cdrecorder       auto
pamconsole,exec,noauto,fscontext=system_u:object_r:removable_t,managed 0 0
```

Figure 4-1: A typical fstab file

grub.conf

GRUB stands for the Grand Unified Bootloader. It is the default boot loader used by Fedora Core and Red Hat Enterprise Linux. GRUB offers a nice graphical interface, giving you a basic choice between which installed operating systems or kernels you want to run. The /etc/grub.conf file is a symbolic link to the actual file that is located in /boot/grub/grub.conf. Figure 4-2 shows a typical grub.conf file.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,1)
#          kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=5
splashimage=(hd0,1)/grub/splash.xpm.gz
hiddenmenu
password --md5 $1$ANJi7kLJ$/NODBfkCTkMAPxZgC8WK10

title Fedora Core (2.6.10-1.770_FC3)
        root (hd0,1)
        kernel /vmlinuz-2.6.10-1.770_FC3 ro root=/dev/VolGroup00/LogVol00 rhgb
quiet
        initrd /initrd-2.6.10-1.770_FC3.img
title Fedora Core (2.6.10-1.766_FC3)
        root (hd0,1)
        kernel /vmlinuz-2.6.10-1.766_FC3 ro root=/dev/VolGroup00/LogVol00 rhgb
quiet
        initrd /initrd-2.6.10-1.766_FC3.img
title Fedora Core (2.6.9-1.724_FC3)
        root (hd0,1)
        kernel /vmlinuz-2.6.9-1.724_FC3 ro root=/dev/VolGroup00/LogVol00 rhgb
quiet
        initrd /initrd-2.6.9-1.724_FC3.img
#title Fedora Core (2.6.9-1.667)
#       root (hd0,1)
#       kernel /vmlinuz-2.6.9-1.667 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
#       initrd /initrd-2.6.9-1.667.img
title Other
        rootnoverify (hd0,0)
        chainloader +1
```

Figure 4-2: A typical GRUB configuration file

The default=0 line indicates that the first title section should be booted by default. GRUB starts its counting at 0 instead of 1. The title line contains the label that will be shown in the boot menu for that kernel. The root line specifies that Linux will be booted off the first hard drive. The kernel line indicates the kernel's location on the file system. In the Other title section, notice that GRUB is calling a chain loader to be used for loading a different operating system; in this case it is actually Windows XP. GRUB uses a chain loader to load any operating system that it doesn't support.

cron files

cron is a daemon that executes commands according to a preset schedule that a user defines. It wakes up every minute and checks all cron files to see what jobs need to be run at that time. cron files can be set up by users or by the administrator to take care of system tasks. Basically, users edit their crontab files by telling cron what programs they'd like run automatically and how often they'd like to run them. User crontab files are stored in */var/spool/cron/*. They are named after the user they belong to. Please note that you should never manually edit the files in the */var/spool/cron* directory.

System cron files are stored in the following subdirectories of the /etc directory:

- cron.d
- cron.daily
- cron.hourly
- cron.monthly
- cron.weekly

crontab in the /etc directory is sort of the master control file set up to run all the scripts in the cron.daily directory on a daily basis, all the scripts in the cron.hourly directory on an hourly bases, and so on with cron.monthly and cron.weekly.

cron.d is where system maintenance files that need to be run on a different schedule than the other /etc cron files are kept. By default, a file in cron.d called *sysstat* runs a system activity accounting tool every 10 minutes, 24 × 7.

syslog.conf

The *syslog* daemon logs any notable events on your local system. It can store these logs in a local file or send them to a remote log host for added security. It can also accept logs from other machines when acting as a remote log host. These options and more, such as how detailed the logging should be, are set in the *syslog.conf* file. Figure 4-3 is an excerpt that demonstrates the syntax and logic of the *syslog.conf* file.

The first entry specifies that all messages that are severity-level info or higher should be logged in the */var/log/messages* file. Also indicated by the first entry is that any mail, news, private authentication, and cron messages should be logged elsewhere. Having separate log files makes it easier to search through logs if they are separated by type or program. The lines following this one specify the other places where those messages should be logged. For example, authentication privilege messages contain somewhat sensitive information, so they are logged to /var/log/secure, all mail messages are logged to /var/log/maillog and so on.

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                         /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none        /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                      /var/log/secure

# Log all the mail messages in one place.
mail.*                                          -/var/log/maillog


# Log cron stuff
cron.*                                          /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                   /var/log/spooler

# Save boot messages also to boot.log
local7.*                                        /var/log/boot.log


# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none;cron.none  /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                       /var/log/secure

# Log all the mail messages in one place.
mail.*                                           /var/log/maillog


# Log cron stuff
cron.*                                           /var/log/cron
```

Figure 4-3: an excerpt from the /etc/syslog.conf file

ld.so.conf
This configuration file is used by ldconfig, which configures dynamic linker runtime bindings. It contains a listing of directories that hold shared libraries. Shared library files typically end with **.so**, whereas static library files typically end with **.a**, indicating they are an archive of objects. You may need to edit this file if you've installed a program that has installed a shared library to a different library directory that is not listed in the *ld.so.conf file*. In this case, you get an error at runtime that the library does not exist.

```
include ld.so.conf.d/*.conf
/usr/X11R6/lib
/usr/lib/qt-3.3/lib
/usr/lib/mysql
```

Figure 4-4: A typical ld.so.conf file

logrotate.conf
logrotate.conf and the files within the *logrotate.d* directory determine how often your log files are rotated by the *logrotate* program. Log rotation refers to the process of deleting older log files and replacing them with more recent ones. *logrotate* program can automatically rotate, compress, remove, and mail your log files. Log files can be rotated based on size or on time, such as daily, weekly, or monthly. For every program that has a separate log rotation configuration file in *logrotate.d*, and uses *syslogd* for logging, there should be a *logrot config* file for all log entries in /*etc*/syslog.conf, as well as log files produced by external applications, such as Apache. This is because syslog needs to save log entries for these programs in separate files so that their log files can be rotated independently of one another.

4.4 EXAMINING THE /ETC/SYSCONFIG/ DIRECTORY

We now take a look at some of the files found in the */etc/sysconfig/* directory, their functions, and their contents. This information is not intended to be complete, as many of these files have a variety of options used only in very specific or rare circumstances. The */usr/share/doc/initscripts-version-number/sysconfig.txt* file contains a more authoritative listing of the files found in the */etc/sysconfig* directory and the configuration options available. These files are used to pass configuration information to scripts that run when the system starts. It is possible that your system may be missing some of the configuration files described here, or it may have more of the files and directories, depending on whether the corresponding programs that need the files are installed or not. Also, if the service that uses the

configuration file is not started, the configuration file will not be read.

4.4.1 Files in the /etc/sysconfig/ Directory

/etc/sysconfig/apmd

The */etc/sysconfig/apmd* file is used by apmd as a configuration for what things to start, stop, change on suspend, or resume. It provides information to the apmd during startup if apmd is set to start, depending on whether your hardware supports Advanced Power Management (APM) or whether you choose to use it. APM is a monitoring daemon that works with power management code within the Linux kernel. It can alert you to a low battery if you are using Red Hat Linux on a laptop, among other things.

/etc/sysconfig/authconfig

The */etc/sysconfig/authconfig* file provides settings to */usr/sbin/authconfig,* which is called from */etc/rc.sysinit* for the kind of authorization to be used on the host. The basic syntax for lines in this file is:

USE <service name> = <value>

Some sample lines from the file are shown here.

- **USEMD5=value**, where *value* is one of the following:
  - *yes* — MD5 is used for authentication.
  - *no* — MD5 is not used for authentication.
- **USEKERBEROS=value**, where *value* is one of the following:
  - *yes* — Kerberos is used for authentication.
  - *no* — Kerberos is not used for authentication.
- **USELDAPAUTH=value**, where *value* is one of the following:
  - *yes* — LDAP is used for authentication.
  - *no* — LDAP is not used for authentication.

/etc/sysconfig/clock

The */etc/sysconfig/clock* file controls the interpretation of values read from the system clock. Currently, the correct values are as follows:

- **UTC=value**, where *value* is one of the following Boolean values:
  - *true* — Indicates that the hardware clock is set to Universal Time.
  - Any other value indicates that it is set to local time.
- **ARC=value**, where *value* is the following:
  - *true* — Indicates the ARC console's 42-year time offset is in effect.
  - Any other value indicates that the normal UNIX epoch is assumed (for Alpha-based systems only).

- **ZONE=filename** — Indicates the time zone file under */usr/share/zoneinfo* that */etc/localtime* is a copy of, such as: ZONE="Asia/Kolkata". Identifies the time zone file copied into */etc/localtime*. Time zone files are stored in */usr/share/zoneinfo.*

/etc/sysconfig/crond
This file contains settings for the cron daemon. You typically do not need to make any changes to this file.

/etc/sysconfig/desktop
The */etc/sysconfig/desktop* file specifies the desktop manager to be run and is used by the */etc/X11/xinit/Xclients* script, for example:

DESKTOP="GNOME"

/etc/sysconfig/grub
The /etc/*sysconfig*/grub file is used to pass arguments to GRUB at boot time. The information passed is the drive to boot from and whether to use lba mode.

/etc/sysconfig/harddisks
The */etc/sysconfig/harddisks* file allows you to tune your hard drive(s). But be careful do not make changes to this file lightly. If you change the default values stored here, you could corrupt all of the data on your hard drive(s). The */etc/sysconfig/harddisks* file may contain the following:

- **USE_DMA=1**, where setting this to 1 enables DMA. However, with some chipsets and hard-drive combinations, DMA can cause data corruption. Check with your hard-drive documentation or manufacturer before enabling this.
- **Multiple_IO=16**, where a setting of 16 allows for multiple sectors per I/O interrupt. When enabled, this feature reduces operating system overhead by 30 to 50 percent. Use with caution.
- **EIDE_32BIT=3** enables (E)IDE 32-bit I/O support to an interface card.
- **LOOKAHEAD=1** enables drive read-lookahead.
- **EXTRA_PARAMS=** specifies where extra parameters can be added.

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

/etc/sysconfig/i18n
The */etc/sysconfig/i18n* file sets the default language, for example:

LANG="en_US"

/etc/sysconfig/iptables

The */etc/sysconfig/iptables* file stores information used by the kernel to set up packet-filtering services at boot time or whenever the service is started. You should not modify this file by hand unless you are familiar with how to construct iptables rules. The simplest way to add rules is to use the */usr/sbin/lokkit* command from a terminal prompt if you aren't running an X server. If you are running an X server, you can type *system-config-securitylevel* from a terminal prompt or select *Applications ➪ System Settings ➪ Security Level* from the main menu to start the graphical application to create your firewall. Using these applications automatically edits this file at the end of the process. If you wish, you can manually create rules using */sbin/iptables* and then type */sbin/service iptables save* to add the rules to the */etc/sysconfig/iptables* file. Once this file exists, any firewall rules saved there are persisted through a system reboot or a service restart.

/etc/sysconfig/irda
The */etc/sysconfig/irda* file controls how infrared devices on your system are configured at startup. The following values may be used:

- **IRDA=value**, where *value* is one of the following Boolean values:
  - *yes* — irattach will be run, which periodically checks to see whether anything is trying to connect to the infrared port, such as another notebook computer attempting to make a network connection.
  - *no* — irattach will not be run, preventing infrared device communication.
- **DEVICE=value**, where *value* is the device (usually a serial port) that handles infrared connections.
- **DONGLE=value**, where *value* specifies the type of dongle being used for infrared communication. This setting exists for people who use serial dongles rather than real infrared ports. This line is commented out by default because notebooks with real infrared ports are far more common than computers with add-on dongles.
- **DISCOVERY=value**, where *value* is one of the following Boolean values:
  - *yes* — Starts irattach in discovery mode, meaning it actively checks for other infrared devices.

- *no* — Does not start irattach in discovery mode.

/etc/sysconfig/kernel
The settings in this file specify whether new kernels loaded by the *up2date* utility should be booted by default. You can change the setting from yes to no to prevent the newly updated kernel from booting.

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

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

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

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

For more information about what parameters you can use in this file, type **man named**. By default, the file contains no parameters.

/etc/sysconfig/netdump
The */etc/sysconfig/netdump* file is the configuration file for the */etc/init.d/netdump* service. The netdump service sends both oops data and memory dumps over the network. In general, netdump is not a required service, so you should run it only if you absolutely need to. For more information about what parameters you can use in this file, type **man netdump**.

/etc/sysconfig/selinux
This file is a link to */etc/selinux/config* and is used to control **selinux** on the system. It contains two settings that control the state of

selinux — enforcing, permissive, or disabled — and the type of policy, either targeted or strict. A sample of this file is shown here.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
#permissive - SELinux prints warnings instead of enforcing.
# disabled - SELinux is fully disabled.
  SELINUX=permissive
# SELINUXTYPE= type of policy in use. Possible values are:
# targeted - Only targeted network daemons are protected.
# strict - Full SELinux protection.
  SELINUXTYPE=targeted
```

/etc/sysconfig/system-config-users

The */etc/sysconfig/system-config-users* file is the configuration file for the graphical application User Manager. This file is used to filter out system users such as *root*, *daemon,* and *lp.* This file is edited via the Preferences ➪ Filter system users and groups pull-down menu in the User Manager application and should not be edited manually.

/etc/sysconfig/samba

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

/etc/sysconfig/sendmail

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

- **DAEMON=value**, where *value* is one of the following Boolean values:
  - *yes* — Sendmail should be configured to listen to port 25 for incoming mail. *yes* implies the use of Sendmail's *-bd* options.
  - *no* — Sendmail should not be configured to listen to port 25 for incoming mail.

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

/etc/sysconfig/xinetd
The */etc/sysconfig/xinetd* file is used to pass arguments to the ***xinetd*** daemon at boot time. The ***xinetd*** daemon starts programs that provide Internet services when a request to the port for that service is received. For more information about what parameters you can use in this file, type **man xinetd**.

4.4.2 Directories in the /etc/sysconfig/ Directory

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

apm-scripts
This contains the Red Hat APM *suspend/resume* script. You should not edit this file directly. If you need customization, simply create a file called */etc/sysconfig/apm-scripts/apmcontinue*, and it will be called at the end of the script. Also, you can control the script by editing */etc/sysconfig/apmd*.

daemons
This directory is initially empty after the system installation. It is used to hold the configuration scripts for programs that the user may have installed. For example, the configuration files for the webmin program are placed in this directory during its installation.

networking
This directory is used by the Network Configuration tool (system-config-network), and its contents should not be edited manually.

network-scripts

This directory contains files used for network configuration.

- Network configuration files for each configured network interface, such as *ifcfg-eth0* for the eth0 Ethernet interface.
- Scripts used to bring up and down network interfaces, such as *ifup* and *ifdown*.
- Scripts used to bring up and down ISDN interfaces, such as *ifup-isdn* and *ifdown-isdn*.
- Various shared network function scripts that should not be edited directly.

4.5 EXAMINING THE NETWORK CONFIGURATION FILES

This section discusses the following topics:

- Files to change when setting up a system or moving the system
- Starting up network services from xinetd
- Starting up network services from the rc scripts
- Other important network configuration files in the /etc/sysconfig directory

4.5.1 Files to Change When Setting Up a System or Moving the System

Whenever you set up a system to work on a new network, either because you've just installed Red Hat or you're moving the machine from one location to another, a set of files needs to be modified to get it working on the new network. You need to:

- Set up the IP addresses of your network interfaces. Make changes to:
  - /etc/sysconfig/network-scripts/ifcfg-eth0
- Set up the hostname of your machine. Make changes to:
  - /etc/sysconfig/network
  - /etc/hosts
- Set up the DNS servers to reference. Make changes to:
  - /etc/resolv.conf
- Make a local file of hostname to IP address mappings. Make changes to:
  - /etc/hosts
- Set up the device order from which hostnames are looked up. Make changes to:
  - /etc/nsswitch.conf

Fedora Core and Red Hat Enterprise Linux provide a handy graphical tool, called the Network Configuration tool for configuring your network settings. Start up the Network Configuration tool while in X-Window, and enjoy an interface very similar to the Windows control panel for networks. If you use the Network Configuration tool to set up your network, you do not need to edit the files manually as explained in the next sections. Also, if you use DHCP to obtain your IP information, you do not need to do any manual configuration.

Setting Up the IP Address

The first thing you should do is set an IP address on your network interfaces. This step provides your computer with an identity on the network. If you haven't set the IP address already in the installation

process, you need to edit the configuration files by hand. To set the IP address on your first Ethernet interface eth0, edit the */etc/sysconfig/network-scripts/ifcfg-eth0* file. A copy of this file is shown below. Insert your interface's IP address on the line that says:

IPADDR=""

```
DEVICE="eth0"
BOOTPROTO="static"
BROADCAST=192.168.1.255
IPADDR="192.168.1.10"
NETMASK="255.255.255.0"
NETWORK=192.168.1.0
ONBOOT="yes"
USERCTL=no
```

File 4-5: The /etc/sysconfig/network-scripts/ifcfg-eth0 file

Setting Up the Hostname

Once you've picked your hostname, you need to put it into two different places: */etc/sysconfig/network* and */etc/hosts*. In */etc/sysconfig/network*, shown next, change the line that says:

HOSTNAME="idolpc"

This is the /etc/sysconfig/network file:

NETWORKING=yes
HOSTNAME="idolpc"
GATEWAY="192.168.1.1"
GATEWAYDEV="eth0"
FORWARD_IPV4="yes"

You also need to modify the /etc/hosts file. Change the first line in the file, which would look something like this by adding the hostname you want:

127.0.0.1 idolpc localhost.localdomain localhost locala localb localc

Setting Up the DNS Name Resolution

Setting the IP address should enable communication with the other hosts on the network. However, you won't be able to talk to them unless you know their IP addresses, because you haven't set up what DNS servers you should reference to map hostnames to IP addresses. The program that resolves hostnames to IP addresses reads a file called *resolv.conf*, so you need to put your DNS server IP addresses there. Generally, you need one name server, but you can include up to three, if you'd like. Specifying

more than one name server is important. If the first one on the list is not responding, your computer tries to resolve against the next one on the list, and so on, until it finds one that is responding.
Edit */etc/resolv.conf* to contain a list of name servers, like this:

nameserver 1.2.3.4
nameserver 1.2.3.5
nameserver 1.2.3.6

Making a Local File of Hostname to IP Address Mappings

Linux gives you the ability to store a list of hostnames and their corresponding IP addresses in */etc/hosts*, so that you don't have to look them up in **DNS** every time you use them. While you shouldn't do this with every hostname you ever use, one of the advantages gained by configuring often-used hostnames in this way includes the ability to alias a fully qualified hostname to a shorter version of itself. Another useful example occurs if you're monitoring several servers' network services from a monitoring host. If you're monitoring SSH connectivity to certain servers, for example, and your DNS server stops responding, then the monitoring software may report that all your hosts are down. This happens because the monitoring software tries to connect to the server via its hostname, and gets no response because DNS is not providing it with an IP address to connect to. In this case it looks as if your whole network fell over, when the real problem is that your DNS service is not responding properly. To keep this kind of scenario from happening, you should put the hostnames and IP addresses of all your monitored servers in */etc/hosts*. This way, your monitoring software looks into */etc/hosts* to get the proper IP addresses, instead of relying on DNS. The only caveat to keep in mind when putting hosts in */etc/hosts* is that if the hostname's IP address changes for whatever reason, the hosts file does not automatically update to reflect that change. If you start getting connection errors when connecting to a host in the */etc/hosts* file, you should do an **nslookup** on the host and update your */etc/hosts* file accordingly. Your */etc/hosts* file should contain IP address to hostname mappings that follow this format

IP_address canonical_hostname aliases

so that the lines look like this:

192.168.1.66	foo.xena.edu	foo
192.168.1.76	buffy.xena.edu	buffy
152.2.210.81	sunsite.unc.edu	sunsite

Setting Up Name Service Resolution Order

Once you have set up your DNS servers and hosts file, you need to tell your Linux server which method it should use first to look up hostnames. The place to set up this configuration is in the */etc/nsswitch.conf* file. Edit the following line:

hosts: files nisplus dns

The order of the words files, nisplus, and dns determines which method is checked first. **Files** refers to the */etc/hosts* file, **nisplus** refers to any nisplus servers you may have on your network, and **dns** refers to any DNS servers you have set up your machine to reference. The */etc/nsswitch.conf* file contains some other useful settings; for example, whether the server should authenticate users off the local password file or off the network's NIS plus service.

```
#       db                      Use the local database (.db) files
#       compat                  Use NIS on compat mode
#       hesiod                  Use Hesiod for user lookups
#       [NOTFOUND=return]       Stop searching if not found so far
#

# To use db, put the "db" in front of "files" for entries you want to be
# looked up first in the databases
#
# Example:
#passwd:    db files nisplus nis
#shadow:    db files nisplus nis
#group:     db files nisplus nis
bootparams: nisplus [NOTFOUND=return] files

ethers:     files
netmasks:   files
networks:   files
protocols:  files nisplus
rpc:        files
services:   files nisplus

netgroup:   files nisplus

publickey:  nisplus

automount:  files nisplus
aliases:    files nisplus
```

Figure 4-5: The /etc/nsswitch.conf file

4.5.2 Starting Up Network Services from xinetd

xinetd is the replacement for inetd. xinetd is started on bootup and listens on ports designated in the */etc/xinetd.conf* for incoming network connections. When a new connection is made, xinetd starts up the corresponding network service.

You should disable any unnecessary services from being started from xinetd as part of securing your machine. The way to do this is to edit that service's configuration file. *xinetd's* main configuration file is */etc/xinetd.conf*. At the end of the *xinetd.conf* file is a line that indicates that all the files in the */etc/xinetd.d* are also

included in the configuration. This means that you need to go through the files in that directory as well to turn off any services you don't want. So, to disable Telnet, you would look in */etc/xinetd.d* for a file called **telnet.** The telnet file is shown in figure 4-6. Edit the line in the config file that says disable = no, and change that to **disable = yes**. After that line is set to disable = yes, the service is disabled and does not start up the next time you boot up.

```
/etc/xinetd.d/telnet

# default: on
# description: The telnet server serves telnet sessions; it uses \
#       unencrypted username/password pairs for authentication.
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = yes
}
```

Figure 4-6: The Telnet config file in the xinetd.d directory

4.5.3 Starting Up Network Services from the rc Scripts

Network services that are not started out of xinetd are started out of the rc scripts at boot time. Network services started at the default boot level 3 (multi-user networked mode) are started out of the */etc/rc3.d* directory. If you look in that directory, you should see a file with the name of the service you want to stop or start. The script to start the service starts with an **S**, and the kill script starts with a **K**. For example, SSH is started from */etc/rc3.d /S55sshd*, and killed upon shutdown from */etc/rc6.d/K25sshd*. Runlevel 6 is the shutdown level, so that's why its kill script is located in the rc6.d directory.

4.5.4 Other Important Network Configuration Files in the /etc/sysconfig Directory

You can use the files listed in this section to create routes to other hosts, either on your own network or on outside networks. You also can use these files to set up firewall rules for your network to either allow or disallow connections to your network.

Static-routes

If you want to set up some static routes on your machine, you can do so in the static-routes file. This config file has lines in the following format:

network-interface net network netmask netmask gw gateway

Iptables
iptables is the current Fedora Core and Red Hat Enterprise Linux firewall. It supercedes the *ipchains* firewall. It can use *ipchains* rules as a component of its firewall filtering, but iptables and ipchains cannot be run at the same time. This is the file where the iptables rules are stored. When you install Fedora or Enterprise Linux, the installation asks if you would like to enable a host-based firewall. If you select to enable a host-based firewall, a default set of iptables rules installs according to your preferences.

Network Configuration Files in */etc/sysconfig/network-scripts*

You can use the files in this directory to set the parameters for the hardware and software used for networking. The scripts contained here are used to enable network interfaces and set other network-related parameters.

ifcfg-networkinterfacename
A few files fall into this specification. Red Hat specifies a separate configuration file for each network interface. In a typical Red Hat install, you might have many different network interface config files that all follow the same basic syntax and format. You could have *ifcfg-eth0* for your first **Ethernet interface**, *ifcfg-irlan0* for your **infrared network port**, *ifcfg-lo* for the **network loopback interface**, and *ifcfg-ppp0* for your **PPP network interface**.

ifup and ifdown
These files are symlinks to */sbin/ifup* and */sbin/ifdown*. These scripts are called when the network service is started or stopped. You call these scripts with the name of the interface that you want to bring up or down. If these scripts are called at boot time, then boot is used as the second argument. For instance, to bring your Ethernet interface down and then up again after boot, you would type:

ifup eth0
ifdown eth0

4.6 MANAGING THE INIT SCRIPTS

Init scripts determine which programs start up at boot time. Red Hat and other Unix distributions have different runlevels, so there are a different set of programs that are started at each runlevel. Usually Red Hat Linux starts up in multiuser mode with networking turned on. The runlevels available are:
0 — Halt
1 — Single-user mode
2 — Multiuser mode, without networking

3 — Full multiuser mode
4 — Not used
5 — Full multiuser mode (with an X-based login screen)
6 — Reboot
The system boots into the default runlevel set in */etc/inittab*.

Init scripts can be managed in the following ways:
- Managing rc scripts by hand
- Managing rc scripts using chkconfig

```
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Figure 4-7: The default /etc/inittab file.

4.6.1 Managing rc Scripts by Hand

If you want to configure which services are started at boot time, you need to edit the rc scripts for the appropriate runlevel. The default runlevel is 3, which is full multiuser mode without a graphical interface and runlevel 5 with a graphical interface. So, to change the services that are started in the default runlevel, you should edit the scripts found in */etc/rc3.d*, or */etc/rc5.d* depending on your system. When you look at a directory listing of the rc directories, notice that the files either start with S or K. The files that start with **S** are startup files, and the files that start with **K** are kill files. The S scripts are run in the numerical order listed in their filenames. Note that case is important. Scripts that do not start with a capital S do not run upon startup. One good way to keep scripts

from starting up at boot time without deleting them is to rename the file with a small s at the beginning instead of a capital S. This way you can always put the script back into the startup configuration by capitalizing the initial letter. When the system starts up, it runs through the scripts in the rc directory of the runlevel it's starting up in. So when the system starts up in runlevel 3, it runs the scripts in the */etc/rc3.d* directory. All of the files in **rc#.d** are symbolic links to */etc/init.d* scripts, and the names are used here only to affect what services start or stop and the ordering of those services. Editing the *rc3.d/httpd* file will affect *rc5.d/httpd*. When the system shuts down, the corresponding K or kill scripts are run to shut down the services started from the rc directory. In general, every S script should have a corresponding K script to kill the service at shutdown. When the system is shut down, it enters runlevel 6. So, most K scripts are in */etc/rc6.d*.

If you ever need to restart a service that's started from an rc directory, an easy way to do it properly is to run its startup script with the restart option. This procedure enables all the proper steps to be followed (configuration files read, lock files released, and so forth) when the service starts up again. So, to restart syslog, for example, run the following command from the rc directory:

```
[root@terry rc3.d]# ./S12syslog restart
Shutting down kernel logger:                               [  OK  ]
Shutting down system logger:                               [  OK  ]
Starting system logger:                                    [  OK  ]
Starting kernel logger:                                    [  OK  ]
```

4.6.2 Managing rc Scripts Using chkconfig

Fedora Core and Red Hat Enterprise Linux come with a useful tool called *chkconfig*. It helps the system administrator manage rc scripts and xinetd configuration files without having to manipulate them directly. It is inspired by the *chkconfig* command included in the **IRIX** operating system. Type ***chkconfig --list*** to see all the services chkconfig knows about, and whether they are stopped or started in each runlevel. An abridged example output is shown in the figure 4-8. The chkconfig output can be a lot longer than that listed here. The first column is the name of the installed service. The next seven columns each represent a runlevel, and tell you whether that service is turned on or off in that runlevel. Since xinetd is started on the system whose chkconfig output is excerpted, at the end of chkconfig's report is a listing of what xinetd started services are configured to begin at boot time. The listing is abridged, since a lot of services can be started from xinetd, and there's no need to show all of them. Figure 4-8 shows the output of *chkconfig --list*:

To turn a service off or on using chkconfig, use this syntax:

chkconfig -level*[0-6](you must choose the runlevel)* servicename off|on|reset

So, to turn off the gpm daemon turned on previously, type:

chkconfig --level 2 gpm off

To turn on xinetd, type:

chkconfig xinetd on

Run *chkconfig --list* again to see if the service you changed has been set to the state you desire. Changes you make with chkconfig take place the next time you boot up the system. You can always start, stop, or restart a service by running ***service (service name)*** from a terminal prompt.

```
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
rwhod           0:off   1:off   2:off   3:off   4:off   5:off   6:off
keytable        0:off   1:on    2:on    3:on    4:on    5:on    6:off
nscd            0:off   1:off   2:off   3:off   4:off   5:off   6:off
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
gpm             0:off   1:off   2:on    3:off   4:off   5:off   6:off
kudzu           0:off   1:off   2:off   3:on    4:on    5:on    6:off
kdcrotate       0:off   1:off   2:off   3:off   4:off   5:off   6:off
lpd             0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
sendmail        0:off   1:off   2:on    3:off   4:off   5:off   6:off
rhnsd           0:off   1:off   2:off   3:on    4:on    5:on    6:off
netfs           0:off   1:off   2:off   3:off   4:off   5:off   6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
random          0:off   1:off   2:on    3:on    4:on    5:on    6:off
rawdevices      0:off   1:off   2:off   3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:off   4:off   5:off   6:off
ipchains        0:off   1:off   2:off   3:off   4:off   5:off   6:off
<snip>

xinetd based services:
        rexec:  off
        rlogin: off
        rsh:    off
        chargen:        off
        chargen-udp:    off
        daytime:        off
        daytime-udp:    off
        echo:   off
        echo-udp:       off
        time:   off
        time-udp:       off
        finger: off
        ntalk:  off
        talk:   off
        telnet: off
        wu-ftpd:        on
        rsync:  off
        eklogin:        off
        gssftp: off
        klogin: off
```

Figure 4-8: Output from chkconfig –list.

4.7 SUMMARY

- All systemwide configuration files are located in /etc. So, if you want to change something across the system, look in /etc and its subdirectories first. If you're at a loss in terms of figuring out which configuration file you need to edit, try grepping for keywords in /etc.

- To change configuration variables for one or a few users, you can usually edit configuration files within the individual users' home directories. Most configuration files in home directories start with a . (period) so you need to look for them with the **ls -a** command.

- Be mindful of configuration file permissions to ensure that unauthorized parties cannot modify them. Flat out instant root access for unauthorized parties is one possible outcome of a modified configuration file. A more likely outcome is that a configuration file modification would make it easier for a system compromise to take place.

- You can either edit startup files by hand or by using one of the system administration tools such as chkconfig. You should at least know the format of the startup files and where they are, so that if automatic tools can't do the job for some reason, you can always change things yourself.

4.8 REVIEW QUESTIONS

(1) Explain the fstab file.
(2) Write a note on grub.conf.
(3) Write a short note on syslog.conf and cron files.
(4) Explain any 5 files in the /etc/sysconfig/ directory.
(5) List and explain the files that need to be changed when setting up a system.
(6) Explain how to manage init scripts.

4.9 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- Beginning Linux Programming 4th Edition by Neil Mathew, Richard Stone. Wiley Publishing
- Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill
- Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill
- Red Hat Linux Networking and System Administration 3rd Edition by Terry Collins and Kurt Wall.
- Sybex – RHCE Red Hat Certified Engineer Study Guide
- Red Hat Certified Technician & Engineer by Asghar Ghori.
- www.thegeekstuff.com
- www.tlpd.org
- www.linuxtopia.org

5

TCP/IP NETWORKING

Unit Structure :

5.0 OBJECTIVES

This chapter provides an overview of TCP/IP protocols and how it is applied to networking with Fedora Core and Red Hat Enterprise Linux. After giving a description of TCP/IP, this chapter explains how to configure such a network to the Red Hat environment.

5.1 INTRODUCTION

TCP/IP is an acronym for Transmission Control Protocol/Internet Protocol, and refers to a family of protocols used for computer communications. TCP and IP are just two of the separate protocols contained in the group of protocols developed by the Department of Defense, sometimes called the DoD Suite, but more commonly known as TCP/IP. In addition to Transmission Control Protocol and Internet Protocol, this family also includes Address Resolution Protocol (ARP); Domain Name System (DNS); Internet Control Message Protocol (ICMP); User Datagram Protocol (UDP); Routing Information Protocol (RIP); Simple Mail Transfer Protocol (SMTP); Telnet, and many others. These protocols provide the necessary services for basic network functionality. TCP/IP uses the client/server model of communication in which a client program on a source node requests a service and the server program on the destination responds. TCP/IP communication is primarily point-to-point, meaning each communication session is between two nodes. To be able to send and receive information on the network, each device connected to it must have an address. The address of any device on the network must be unique and have a standard, defined format by which it is known to any other device on the network. This device address consists of two parts:

- The address of the network to which the device is connected
- The address of the device itself — its node or host address

Devices that are physically connected to each other (not separated by routers) would have the same network number but different node, or host, numbers. This would be typical of an internal network at a company or university. These types of networks are now often referred to as intranets. The two unique addresses mentioned are called the network layer addresses and the Media Access Control (MAC) addresses. Network Layer addresses are IP addresses that have been assigned to the device. The MAC address is built into the card by the manufacturer and refers to only the lowest-level address by which all data is transferred between devices. Data is transferred across the network by breaking the information into small pieces of data called packets or datagrams. The data (the entire message) is broken into packets and not sent as one long stream of data for two reasons — sharing resources and error correction.

Hardware Address is a unique 48-bit address used to identify the correct destination node for data packets transmitted from a source node. A network protocol called Address Resolution Protocol (ARP), maps a hardware address to the destination node's IP address. Hardware address is also referred to as MAC or physical

address. IP stands for Internet Protocol and represents a unique 32-bit (8 bytes) software address that every single node in the network must have in order to communicate with other nodes. IP addresses can be assigned temporarily and permanently. Temporary addresses are known as dynamic addresses and are typically leased from a DHCP server for a specific period of time. Permanent addresses are referred to as static addresses and are not changed unless there is a requirement. Hardware and IP addresses work together to identify the correct network interface.

5.2 UNDERSTANDING NETWORK CLASSES

Each IP address is divided into two portions: a network portion and a node portion. The network portion identifies the correct destination network and the node portion identifies the correct destination node on that network. Based on how many bits are allocated to the network portion, there are 5 usable IP address classes: A, B, C, D, and E. Classes A, B, and C are widely used, while classes D and E are dedicated for multicast networks and scientific purposes only.

CLASS	FIRST BYTE
Class A	0–127
Class B	128–191
Class C	192–233

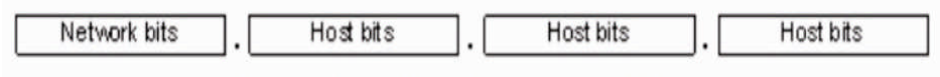
Table 5-1: Network Classses and their IP number range

There are a few ways to assign IP addresses to the devices, depending on the purpose of the network. If the network is internal, an intranet, not connected to an outside network, any class A, B, or C network number can be used. The only requirement is choosing a class that allows for the appropriate number of hosts to be connected. Although this is possible, in the real world this approach would not allow for connecting to the Internet. A more realistic approach would be to register with one of the domain registration services and request an officially assigned network number. An organization called the **InterNIC** maintains a database of all assigned network numbers to ensure that each assignment is unique. After obtaining a network number, the host numbers may be assigned as required. Nearly all IP devices require manual configuration.

The **MAC address**, as described earlier, is the address that is ultimately necessary for transmission of data. For transfer to happen, the **IP address** must be mapped to the Ethernet address

of the device. The mechanism that makes this possible is **Address Resolution Protocol, or ARP**. To determine the Ethernet address of a node on the same network, the sending device sends an ARP request to the Ethernet broadcast address. The Ethernet broadcast address is a special address to which all Ethernet cards are configured to “listen”. The ARP request, containing the sender’s IP and Ethernet addresses, as well as the IP address it is looking for, asks each device for the Ethernet address that corresponds to a particular IP address. The device whose address matches the request sends a reply to the sender’s Ethernet address. The sender is then able to send its data to the specific address it received in response to its ARP request. This process works for sending data between devices on the same network, but what about sending data to devices on different networks? For this you need a router. **Routers** enable networks not physically connected to each other to communicate. A router must be connected physically to each network that wants to communicate. The sending node sends its request to the router on its network. This router is typically called the **default gateway**, and its address must be configured in the sending node’s configuration files. The router receives the request from the sending node and determines the best route for it to use to transmit the data. The router has an internal program, called a **routing table**, which it uses to send the data, either to another router if the other network is not directly connected, or directly to the other network. If the destination network cannot be found in the routing table, then the packet is considered undeliverable and is dropped. Typically, if the packet is dropped, the router sends an ICMP Destination Unreachable message to the sender. **Dynamic acquisition** means that the router sends a message using the Routing Information Protocol (RIP) or Open Shortest Path First (OSPF) protocol. These dynamic protocols enable routers to share details with other routers concerning networks and their locations. Ultimately, the purpose of everything you have looked at so far — packets, IP addresses, and routing — is to give users access to services such as printing, file sharing, and email.

Transmission Control Protocol is encapsulated in IP packets and provides access to services on remote network devices. TCP is considered to be a stream-oriented reliable protocol. The transmission can be any size because it is broken down into small pieces, as you have already seen. Data that is lost is retransmitted, and out-of-order data is reordered. The sender is notified about any data that cannot be delivered. Typical TCP services are File Transfer Protocol (FTP), Telnet, and Simple Mail Transfer Protocol (SMTP).



Class A Address

Class A

Class A addresses are used for networks with an extremely large number of nodes. The first octet defines the network address and the rest are allocated to nodes. The total number of useable network addresses in class A can be up to 2^{8-1} – 2 (126) and the total number of useable node addresses up to 2^{24} – 2 (1,67,77,214). Two is subtracted from both calculations because addresses with all 0's in the first octet and all 1's in the last octet are reserved. Also, one network bit is subtracted from 8 to get 2^{7} network numbers since 0 is reserved. The network address range for class A networks is between 0 and 127 (**0**0000000 to **0**1111111). 0 and 255 in the decimal notation are network and broadcast addresses and are always reserved.

Class B Address

Class B

Class B addresses are used for mid-sized networks. The first two octets define the network address and the remaining are allocated to nodes. The total number of useable network addresses in class B can be up to 2^{16-2} (16,384) and the total number of useable node addresses up to 2^{16} – 2 (65,534). The first two bits in class B network addresses are reserved and therefore, not used in calculation. The network address range for class B networks is between 128 and 191 (**10**000000 to **10**111111).

Class C Address

Class C

Class C addresses are used for small networks not more than 254 nodes. The first three octets define the network address and the fourth is allocated to nodes. The total number of useable network addresses in class C can be up to 2^{24-3} (20,97,152) and the total

number of useable node addresses up to $2^8 - 2$ (254). The first three bits in class C network addresses are reserved and therefore, not used in calculation. The network address range for class C networks is between 192 and 223 (**110**00000 to **110**11111).

Class D ranges from 224 to 239 and Class E ranges from 240 to 255.

5.3 SETTING UP A NETWORK INTERFACE CARD (NIC)

Every Fedora Core and Red Hat Enterprise Linux distribution includes networking support and tools that can be used to configure your network. This section tells you how to configure a network interface card from the command line by modifying the configuration files directly. To configure the network interface card using a graphical based configuration utility is explained in the section titled, “Configuring the Network with the Network Configuration Tool”.

Even if the computer is not connected to outside networks, internal network functionality is required for some applications. This address is known as the loopback address, and its IP address is 127.0.0.1. You should check that this network interface is working before configuring your network cards. To do this, you can use the *ifconfig* utility to get some information. If you type *ifconfig* at a console prompt, you will be shown your current network interface configuration. Figure 5-1 illustrates the output of the ifconfig command.

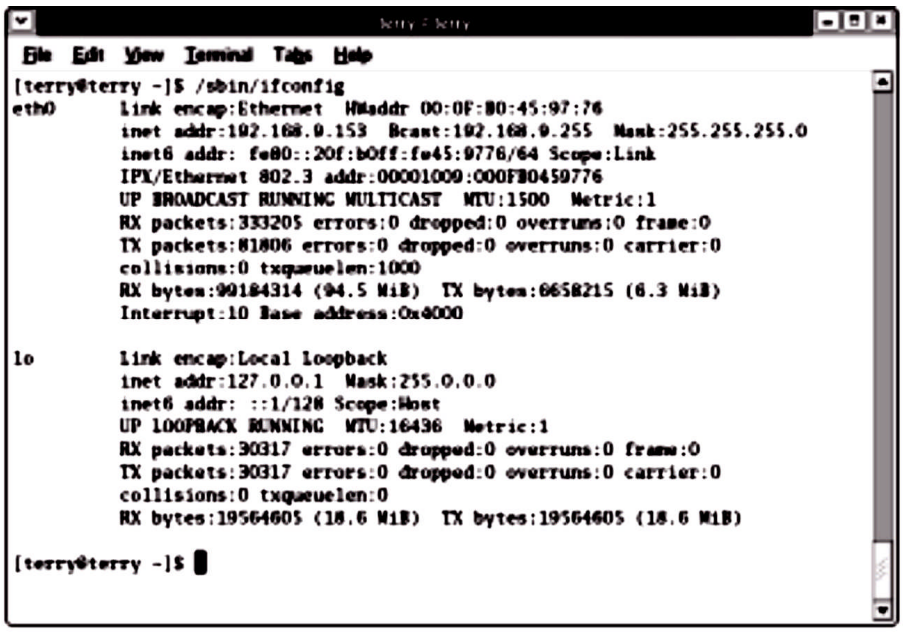


Figure 5-1: The ifconfig utility showing current network information

If your loopback is configured, the ifconfig shows a device called lo with the address 127.0.0.1. If this device and address are not shown, you can add the device by using the ifconfig command as follows:

ifconfig lo 127.0.0.1

You then need to use the route command to give the system a little more information about this interface. For this, type:

route add -net 127.0.0.0

You now have your loopback set up, and the ifconfig command shows the device lo in its listing.

5.3.1 Configuring the Network Card

The procedure for configuring a network card is the same as that for configuring the loopback interface. You use the same command, *ifconfig*, but this time use the name '**eth0**' for an Ethernet device. You also need to know the IP address, the net mask, and the broadcast addresses. These numbers vary, depending on the type of network being built. For an internal network that never connects to the outside world, any IP numbers can be used; however, there are IP numbers typically used with these networks. Table 11-2 shows the IP numbers that are usually used for such networks.

NETWORK CLASS	NET MASK	NETWORK ADDRESSES
A	255.0.0.0	10.0.0.0–10.255.255.255
B	255.255.0.0	172.16.0.0–172.31.255.255
C	255.255.255.0	192.168.0.0–192.168.255.255

Table 5-2: Reserved Network Numbers

If you are connecting to an existing network, you must have its IP address, net mask, and broadcast address. You also need to have the router and domain name server (DNS) addresses. In this example, you configure an Ethernet interface for an internal network. You need to issue the following command:

ifconfig eth0 192.168.2.5 netmask 255.255.255.0 broadcast 192.168.2.255

This results in the creation of device **eth0** with a network address of 192.168.2.5, a **net mask** of 255.255.255.0, and a **broadcast address** of 192.168.2.255. A file is created in */etc/sysconfig/network-scripts* called *ifcfg-eth0*. A listing of this file, shown in Figure 5-2, shows the information that you just entered. The line *onboot=yes* tells the kernel to configure this device at system startup. The line *bootproto=static* means that the IP address

was manually entered for the NIC. If you desire, you can use Dynamic Host Configuration Protocol, or DHCP, to obtain the required IP information for your NIC. Also note that gateway and the MAC address are mentioned in this file.

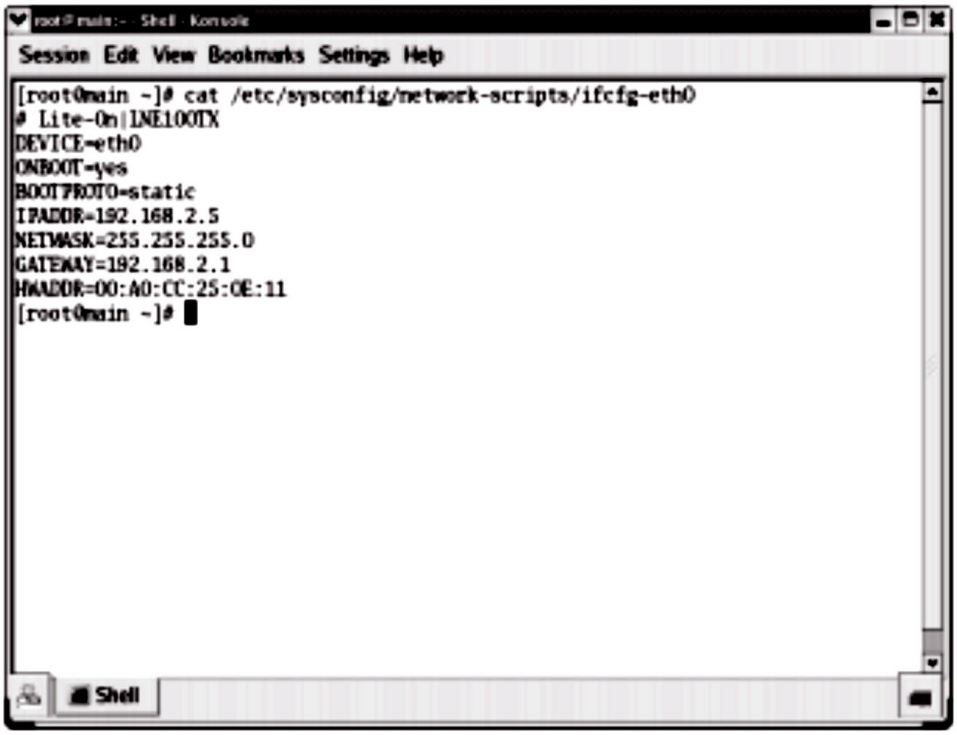


Figure 5-2: The configuration file for the network device eth0.

It is also possible to assign more than one IP address to a single NIC. This is accomplished by using the *ifconfig* and *route* commands. To add another IP address, 192.168.1.4, to **eth0** issue these commands:

ifconfig eth0:1 192.168.1.4
route add -host 192.168.1.4 dev eth0

The first command binds the IP address to the virtual interface **eth0:1**, and the second command adds a route for the address to the actual device eth0. Another method for adding a second IP address to a single NIC is to create an alias file. The configuration file for device eth0 is located in */etc/sysconfig/network-scripts/ifcfg-eth0*. Copy this file to another file called */ifcfg-eth0:1* in the same directory. Open the newly copied file and change the line that reads:

DEVICE=eth0

to read:

DEVICE=eth0:1

5.3.2 Configuring an Internal Network

Now you have a network device configured for one computer. To add additional computers to your network, you need to repeat this process on the other computers you want to add. The only change is that you need to assign a different IP address. For example, the second computer on your network could have the address 192.168.2.6, the third could have 192.168.2.7, and so on. In addition to configuring the network cards on each of the computers in the network, four files on each computer need to be modified. These files are all located in the */etc* directory:

- ***/etc/nsswitch.conf***
- ***/etc/hosts***
- ***/etc/resolv.conf***
- ***/etc/sysconfig/network***

The */etc/nsswitch.conf* file contains configuration information for the name resolver and should contain the following line:

hosts: files dns

This configuration tells the name resolver to check the */etc/hosts* file before attempting to query a name server and to return all valid addresses for a host found in the */etc/hosts* file instead of just the first.

The */etc/hosts* file could contain the names of all the computers on the local network, or an outside network. For a small network, maintaining this file is not difficult, but for a large network, like the Internet, keeping the file up to date is often impractical. Figure 5-3 shows my home network, containing several computers. The first address represents the current system, and the other addresses represent other computers on the network.

The */etc/resolv.conf* file provides information about name servers employed to resolve hostnames. Figure 5-4 shows a typical resolv.conf file listing.

The */etc/sysconfig/network* file contains two lines, as follows:

NETWORKING=yes
HOSTNAME=(host and domain name of your system)

The first line enables networking for your system. The second line displays the hostname of your system and the name of the domain to which it belongs.

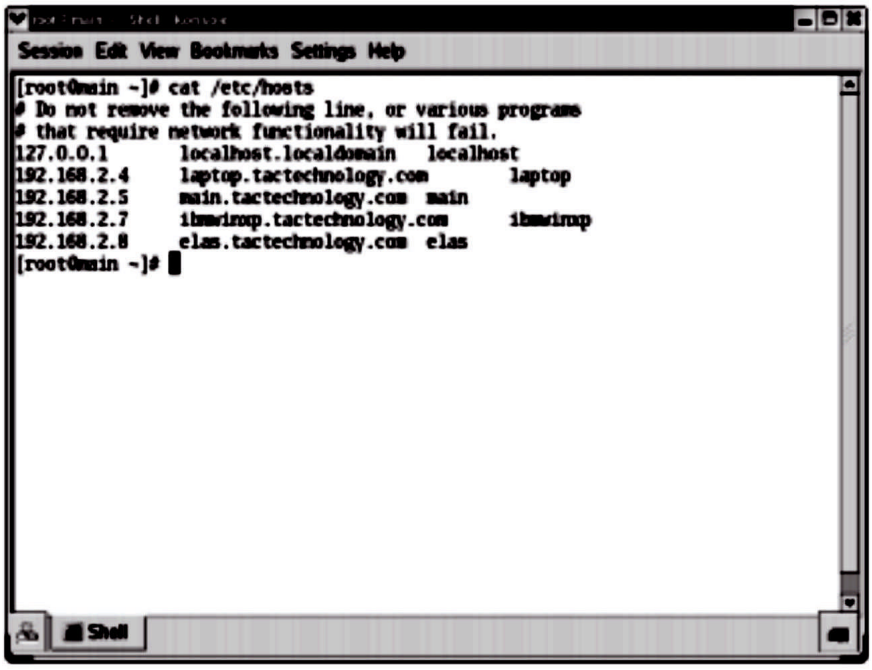


Figure 5-3: The /etc/hosts file contains a listing of the computers on a network.

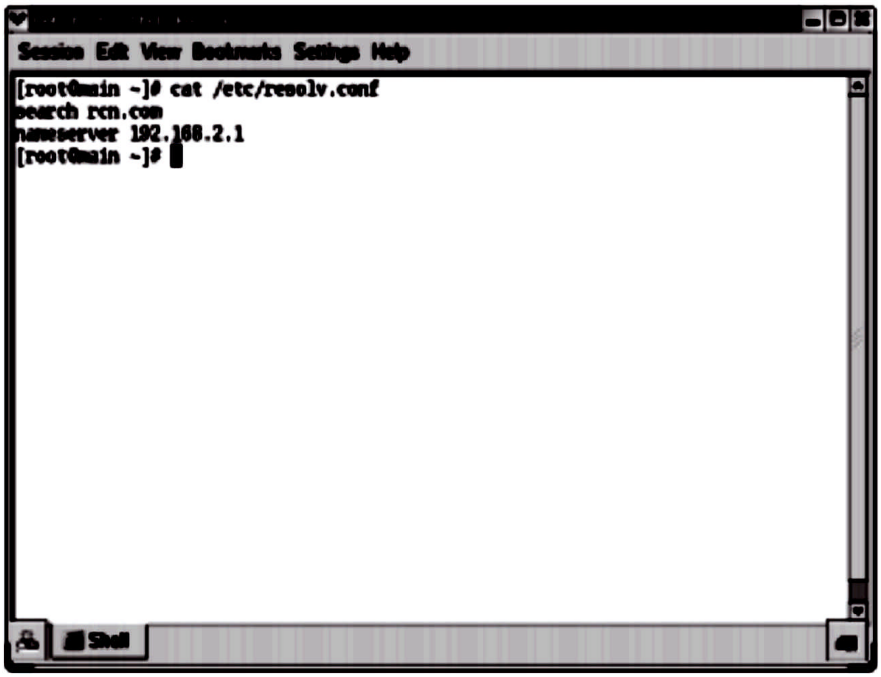


Figure 5-4: The /etc/resolv.conf file contains a listing of the domain and name servers on the network.

5.4 UNDERSTANDING SUBNETTING

Subnetting is a method by which a large network address space can be divided into several smaller and more manageable logical sub-networks, commonly referred to as subnets. Subnetting usually results in reduced network traffic, improved network performance and de-centralized and easier administration among other benefits. Subnetting does not touch the network bits, it makes use of the node bits only. The following should be kept in mind when working with subnetting:

- Subnetting does not increase the number of IP addresses in a network. In fact, it reduces the number of useable IP addresses.
- All nodes in a given subnet must have the same subnet mask.
- Each subnet acts as a separate network and requires a router to talk to other subnets.
- The first and the last IP address in a subnet (similar to a network) are reserved. The first address points to the subnet itself and the last is the broadcast address.

Subnetting employs using required number of node bits. For example, if you wish to divide a class C network address of192.168.12.0 with default netmask of 255.255.255.0 into 6 useable subnets each with 30 useable node addresses, you need 3 left-most node bits (highlighted) from the roght-most octet (node octet), as shown below:

```
    192   .   168   .   12    .   0
11000000.10101000.00001100.000 00000
```

Here is the formula to calculate useable subnets. 2 subnet bits give $2^2 - 2 = 2$ subnets, 3 subnet bits give $2^3 - 2 = 6$ subnets, 4 subnet bits give $2^4 - 2 = 14$ subnets, 5 subnet bits give $2^5 - 2 = 30$ subnets, 6 subnet bits give $2^6 - 2 = 62$ subnets and 7 subnet bits give $2^7 - 2 = 126$ subnets. This formula is applicable to determine number of useable subnets created out of a class A, B, or C network address.

Similarly, use the formula to determine number of useable node addresses. 2 node bits give $2^2 - 2 = 2$ addresses, 3 node bits give $2^3 - 2 = 6$ addresses, 4 node bits give $2^4 - 2 = 14$ addresses, 5

node bits give $2^5 - 2 = 30$ addresses, 6 node bits give $2^6 - 2 = 62$ addresses and 7 node bits give $2^7 - 2 = 126$ addresses. This formula is applicable to determine number of useable node addresses created out of a class A, B, or C network address.

5.4.1 Subnet Mask

After a network address is subnetted, you need to determine something called *subnet mask* or *netmask.* The subnet mask is the network portion plus the subnet bits. In other words, the subnet mask segregates the network bits from the node bits. It is used by routers to identify the start and end of the network/subnet portion and the start and end of the node portion of a given IP address. The subnet mask, like an IP address, can be represented in either decimal or binary notation. The 1's in the subnet mask identify the subnet bits and 0's identify the node bits. The default subnet masks for class A, B and C networks are 255.0.0.0, 255.255.0.0, and 255.255.255.0, respectively.

You should remember two important things about the network mask. The network mask affects only the interpretation of IP numbers on the same network segment, and the network mask is not an IP number; it is used to modify the way IP numbers are interpreted by the network.

A subnet enables you to use one IP address and split it up so that it can be used on several physically connected local networks. This is a tremendous advantage, as the number of IP numbers available is rapidly diminishing. You can have multiple subnetted networks connected to the outside world with just one IP address. By splitting the IP address, it can be used on sites that need multiple connections; splitting the address eliminates the problems of high traffic and difficult manageability. The other advantages to subnetting are that different network topologies can exist on different network segments within the same organization, and overall network traffic is reduced. Subnetting also enables increased security by separating traffic into local networks. There is a limit to the number of subnets that can be created based on the number of times a given number can be divided. Tables 5-3, 5-4, and 5-5 show the possible numbers of subnets and hosts that can exist. In class A networks, there are 22 valid netmasks. In class B networks, there are 14 valid netmasks. In class C networks, there are 6 valid netmasks.

Subnet Mask	# of Subnet Bits	Useable Subnets	Nodes per Subnet
255.128.0.0	1	1	8,388,606
255.192.0.0	2	2	4,194,302
255.224.0.0	3	6	2,097,150
255.240.0.0	4	14	1,048,574
255.248.0.0	5	30	524,286
255.252.0.0	6	62	262,142
255.254.0.0	7	126	131,070
255.255.0.0	8	254	65,534
255.255.128.0	9	510	32,766
255.255.192.0	10	1,022	16,382
255.255.224.0	11	2,046	8,190
255.255.240.0	12	4,094	4,094
255.255.248.0	13	8,190	2,046
255.255.252.0	14	16,382	1,022
255.255.254.0	15	32,766	510
255.255.255.0	16	65,534	254
255.255.255.128	17	131,070	126
255.255.255.192	18	262,142	62
255.255.255.224	19	524,286	30
255.255.255.240	20	1,048,574	14
255.255.255.248	21	2,097,150	6
255.255.255.252	22	4,194,302	2

Table 5-3: Class A subnets and subnet masks.

Subnet Mask	# of Subnet Bits	Useable Subnets	Nodes per Subnet
255.255.128.0	1	1	32,766
255.255.192.0	2	2	16,382
255.255.224.0	3	6	8,190
255.255.240.0	4	14	4,094
255.255.248.0	5	30	2,046
255.255.252.0	6	62	1,022
255.255.254.0	7	126	510
255.255.255.0	8	254	254
255.255.255.128	9	510	126
255.255.255.192	10	1,022	62
255.255.255.224	11	2,046	30
255.255.255.240	12	4,094	14
255.255.255.248	13	8,190	6
255.255.255.252	14	16,382	2

Table 5-4: Class B subnets and subnet masks.

Subnet Mask	# of Subnet Bits	Useable Subnets	Nodes per Subnet
255.255.255.128	1	1	126
255.255.255.192	2	2	62
255.255.255.224	3	6	30
255.255.255.240	4	14	14
255.255.255.248	5	30	6
255.255.255.252	6	62	2

Table 5-5: Class C subnets and subnet masks.

5.4.2 Classless Inter Domain Routing

Classless InterDomain Routing (CIDR) was invented several years ago to keep the Internet from running out of IP addresses. The class system of allocating IP addresses can be very wasteful. Anyone who could reasonably show a need for more than 254 host addresses was given a Class B address block of 65,533 host addresses. Even more wasteful was allocating companies and organizations Class A address blocks, which contain over 16 million host addresses! Only a tiny percentage of the allocated Class A and Class B address space has ever been actually assigned to a host computer on the Internet. People realized that addresses could be conserved if the class system was eliminated. By accurately allocating only the amount of address space that was actually needed, the address space crisis could be avoided for many years. This solution was first proposed in 1992 as a scheme called ***supernetting***. Under supernetting, the class subnet masks are extended so that a network address and subnet mask could, for example, specify multiple Class C subnets with one address.

Under *CIDR*, the subnet mask notation is reduced to simplified shorthand. Instead of spelling out the bits of the subnet mask, the number of 1 bits that start the mask are simply listed. In the example, instead of writing the address and subnet mask as

192.60.128.0, Subnet Mask 255.255.252.0

the network address is written simply as

192.60.128.0/22

This address indicates the starting address of the network, and number of 1 bits (22) in the network portion of the address. If you look at the subnet mask in binary, you can easily see how this notation works.

(11111111.11111111.11111100.00000000)

The use of a CIDR-notated address is the same as for a class address. Class addresses can easily be written in CIDR notation (Class A = /8, Class B = /16, and Class C = /24).

5.5 WORKING WITH GATEWAYS AND ROUTERS

Earlier in this chapter, you learned that a router is necessary for separate networks to communicate with each other. You also learned that each network must be connected to a router in order for this communication to take place. This router connected to each network is called its gateway. In Linux, you can use a computer with two network interfaces to route between two or more subnets. To be able to do this you need to make sure that you enable IP forwarding. All current Linux distributions have IP forwarding compiled as a module, so all you need to do is make sure the

module is loaded. You can check this by entering the following query at a command prompt:

cat /proc/sys/net/ipv4/ip_forward

If forwarding is enabled, the number 1 is displayed; if forwarding is not enabled, the number 0 is displayed. To enable IP forwarding if it is not already enabled, type the following command:

echo “1” > /proc/sys/net/ipv4/ip_forward

Assume that a computer running Linux is acting as a router for your network. It has two network interfaces to the local LANs using the lowest available IP address in each subnetwork on its interface to that network. The network interfaces would be configured as shown in Table 5-6. The network routing the system would use is shown in table 5-7.

INTERFACE	IP ADDRESS	NET MASK
Eth0	192.168.1.1	255.255.255.128
Eth1	192.168.1.129	255.255.255.128

Table 5-6: Network Interface Configuration.

DESTINATION	GATEWAY	MASK	INTERFACE
192.168.1.0	192.168.1.1	255.255.255.128	eth0
192.168.1.128	192.168.1.129	255.255.255.128	eth1

Table 5-7: Network Routing Configuration.

Each computer on the subnet has to show the IP address for the interface that is its gateway to the other network. The computers on the first subnet, the 192.168.1.0 network, would have the gateway 192.168.1.1. Remember that you used the first IP address on this network for the gateway computer. The computers on the second subnet, 192.168.1.128, would use 192.168.1.129 as the gateway address. You can add this information using the route command as follows:

route add -net 192.168.1.0

and then type

route add default gw 192.168.1.129

This command sets up the route for local (internal) routing and the external route for your first subnet. You need to repeat the previous commands, substituting the appropriate numbers for the second subnet and any additional subnets.

5.6 CONFIGURING DYNAMIC HOST CONTROL PROTOCOL (DHCP)

So far, you have learned to configure a network card and assign it an IP address, subnet mask, broadcast address, and gateway. Using Dynamic Host Configuration Protocol (DHCP), you can have an IP address and the other information automatically assigned to the hosts connected to your network. This method is quite efficient and convenient for large networks with many hosts, because the process of manually configuring each host is quite time consuming. By using DHCP, you can ensure that every host on your network has a valid IP address, subnet mask, broadcast address, and gateway, with minimum effort on your part.

5.6.1 Setting Up the Server

The program that runs on the server is *dhcpd* and is included as an RPM on the Fedora Core and Red Hat Enterprise Linux installation CDs. You can install it using the Package Management tool.

The configuration file for the DHCP server is */etc/dhcpd.conf*. This file does not contain any directives by default. You may copy the template *dhcpd.conf.sample* from */usr/share/doc/dhcp** directory into */etc*. A slightly customized version of the file is shown below (Figure 5-5).

A sample file is created when you install the dhcpd package that you can use as a guide. The sample file is in */usr/share/doc/dhcp** directory. You can modify it using a text editor. Be sure to use the proper addresses for your network. You need to restart the DHCP server whenever you make changes to the */etc/dhcpd.conf* file.

To start the server, run the command ***service dhcpd start***. To ensure that the **dhcpd** program runs whenever the system is booted, you should run the command ***chkconfig --level 35 dhcpd on***.

```
ddns-update-style none;
ignore client-updates;
subnet                          192.168.0.0
netmask                         255.255.255.0 {
= --- default gateway
     option routers             192.168.0.1;
     option subnet-mask         255.255.255.0;
     option nis-domain          "nis_domain";
     option domain-name         "testdom ";
     option domain-name-servers 192.168.0.201;
     option time-offset         -18000; = Eastern Standard Time
     option ntp-servers         192.168.0.201;
. . . . . .
      range dynamic-bootp       192.168.0.2  192.168.0.20;
      default-lease-time        21600;
      max-lease-time            43200;
      = we want the nameserver to appear at a fixed address
      host rhel01 {
          next-server           rhel02.getitcertify.com;
          hardware ethernet     00:0C:29:EC:5D:99;
          fixed-address         192.168.0.250;
     }
}
```

Figure 5-5: The dhcpd.conf file.

5.6.2 Configuring the DHCP Client

First, you need to be sure that you NIC is properly configured and recognized by your system. After that, it is easy to tell your system to use DHCP to obtain its IP information. Follow these steps:

(1) Using your favorite text editor, open the */etc/sysconfig/network-scripts/ifcfg-eth0* file.
(2) Find the line **bootproto=static**.
(3) Change static to **dhcp**.
(4) Save your changes.
(5) Restart the network by issuing the command **service network restart**, and your system will receive its IP information from the DHCP server.

5.7 CONFIGURING THE NETWORK USING THE NETWORK CONFIGURATION TOOL

Fedora Core and Red Hat Enterprise Linux provide a graphical network configuration tool that you can use to configure network interface devices installed in your system. With this tool, you can configure Crypto IP Encapsulation (CIPE), Ethernet, Integrated Services Digital Network (ISDN), modem, token ring,

wireless, and xDSL. The x refers to different versions of Digital Subscriber Loop (DSL) devices. You can access the Network Configuration tool by using the Applications menu from the GNOME desktop. To start the Network Configuration tool in Enterprise Linux choose Applications ➪ System Settings ➪ Network. In Fedora Core 4 choose Desktop ➪ System Settings ➪ Network. The Network Configuration window appears as shown in Figure 5-6.

The main Network Configuration tool window (shown in Figure 5-6) has five tabbed pages and opens to the Devices tab by default.

- **Devices** — This tab shows the network devices that are installed and configured on your PC. Network devices are associated with the actual physical hardware in the PC.

- **Hardware** — This tab shows the actual physical hardware installed in your PC.

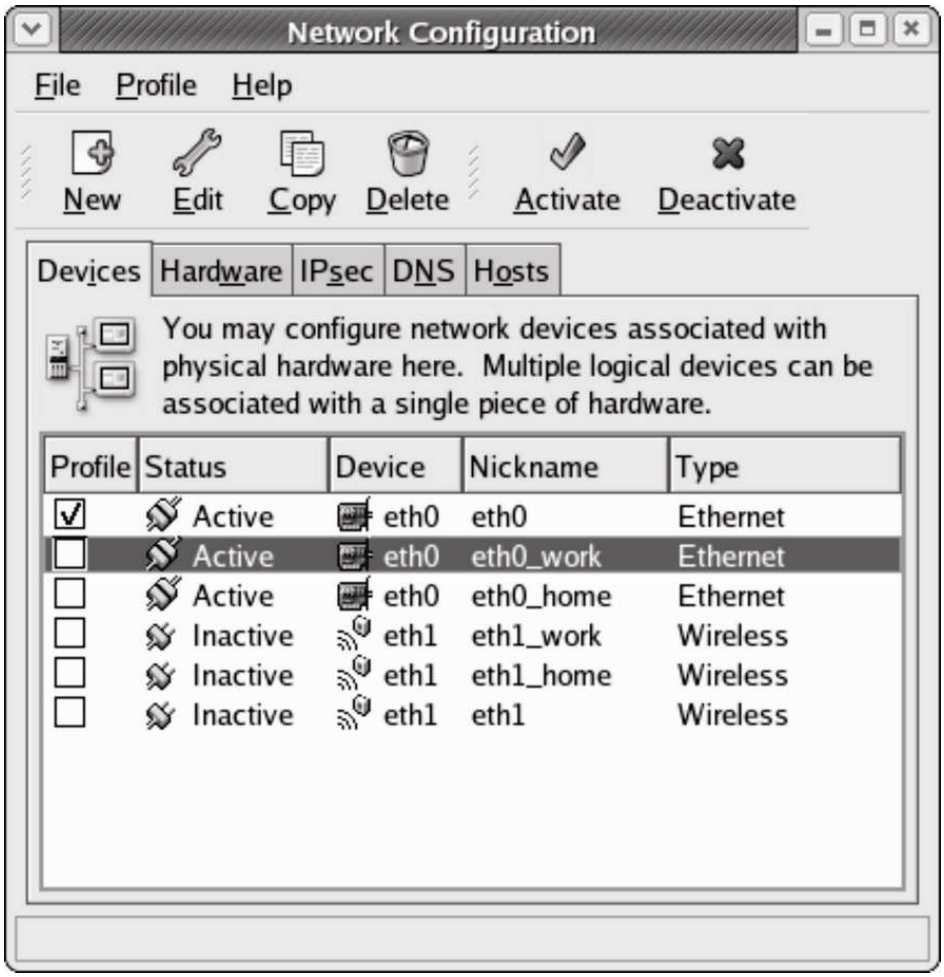


Figure 5-6: The Network Configuration Tool main window.

- **IPSec** — This tab is where you can configure IPSec tunnels used for secure communications.
- **DNS** — This tab shows the system hostname, domain, and name servers used for DNS lookups. You can configure this information here.
- **Hosts** — This tab shows the PC hostname to static IP address mapping.

If you have a supported NIC installed on your system during installation of Red Hat Enterprise Linux, your NIC should already be listed in the Network Configuration Tool. Click the Hardware tab to see information about the device. Figure 5-6 shows an Ethernet NIC with a wireless NIC already installed.

5.8 EDITING YOUR NETWORK CONFIGURATION

After you add and configure your network connection device, whether it is a wired NIC, wireless NIC, or modem, you usually don't need to change the configuration. You might need to modify the configuration, though, if you change to a different NIC.

5.8.1 Removing a NIC

Using the Network Configuration tool, you can easily make the necessary changes. Start the Network Configuration tool as follows:

- In Enterprise Linux choose Applications ➪ System Settings ➪ Network. In Fedora Core 4 choose Desktop ➪ System Settings ➪ Network.
- Click the Hardware tab.
- Highlight the device that you want to remove, and then click Delete.
- When finished, choose File ➪ Save to save your changes.

5.8.2 Changing the NIC Configuration

Using the Network Configuration tool, you can easily make the necessary changes. Start the Network Configuration tool as follows:

- In Enterprise Linux choose Applications ➪ System Settings ➪ Network. In Fedora Core 4 choose Desktop ➪ System Settings ➪ Network.
- Highlight the device that you want to modify, and then click Edit (on the toolbar). The Ethernet Device properties dialog box for the device you selected is shown in Figure 5-7.
- The three tabs available from this dialog box are used for the following:

- General — Here you can enter a nickname for the device and choose whether the device is activated when the system starts. You can also choose to allow other users to be able to enable and disable the device. You can choose to obtain IP information automatically by using DHCP, or you can manually enter the IP information for the device.
- Route — Here you can enter static routes to other networks. You need to enter the network IP number as well as the gateway IP number. In most cases, you don't need to enter any information here if you are using DHCP.
- Hardware Device — This tab contains information about the hardware associated with the Ethernet device. You can assign device aliases here if you desire. Device aliases are virtual devices associated with the same physical hardware, and are useful if you want to have more than one IP address for a system but the system has only one network card. If you have configured a device, such as eth0:
  - Click the Add button in the Network Administration tool to create an alias for the device.
  - Select the network device and configure the network settings.
- After you make the changes you desire, click OK to return to the Network Configuration dialog box.
- Choose File ➪ Save to write your configuration changes to a file.

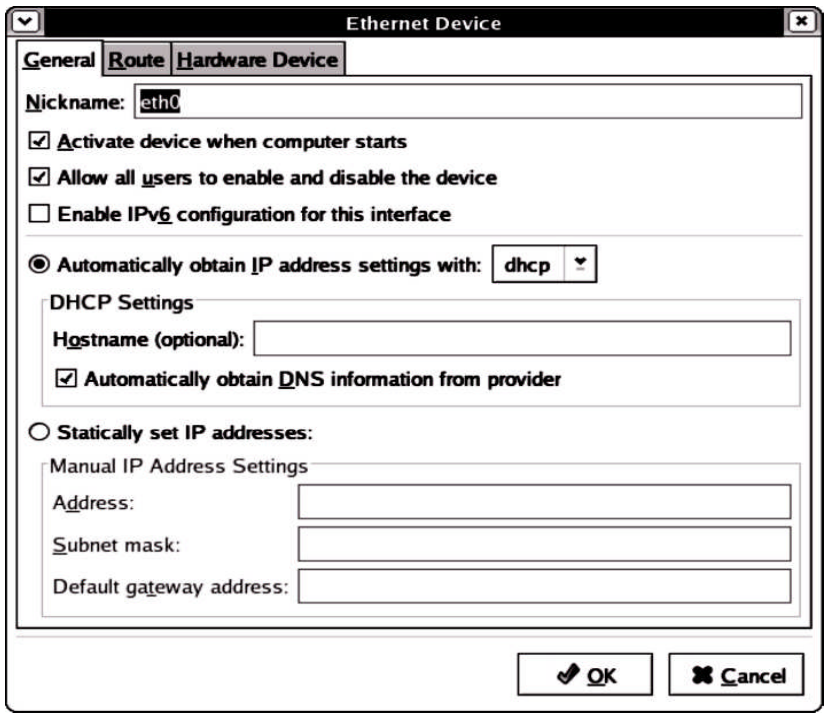


Figure 5-7: The Ethernet Device properties dialog box.

5.8.3 Managing DNS Settings

The DNS tab of the Network Configuration tool is where you configure the system's hostname, domain, name servers, and search domain. *Name servers* are used to look up other hosts on the network. To enter or change these settings, do the following:

- In Enterprise Linux choose Applications ➪ System Settings ➪ Network. In Fedora Core 4 choose Desktop ➪ System Settings ➪ Network.
- Click the DNS tab from the Network Configuration dialog box.
- On the DNS tab, enter the appropriate information for your system.
- After you finish, choose File ➪ Save to save your changes.

5.8.4 Managing Hosts

On the Hosts tab of the Network Configuration tool, you can add, edit, or remove hosts to or from the */etc/hosts* file. This file contains IP addresses and their corresponding hostnames. When your system tries to resolve a hostname to an IP address or determine the hostname for an IP address, it refers to the */etc/hosts* file before using the name servers (if you are using the default Fedora Core or Red Hat Enterprise Linux configuration). If the IP address is listed in the */etc/hosts* file, the name servers are not used. If your network contains computers whose IP addresses are not listed in DNS, it is recommended that you add them to the */etc/hosts* file.

- In Enterprise Linux choose Applications ➪ System Settings ➪ Network. In Fedora Core 4 choose Desktop ➪ System Settings ➪ Network.
- Click the Hosts tab from the Network Configuration dialog box. The Hosts tab that appears shows the hostname to static IP address mappings, if any.
- Click New from the toolbar to open the Add/Edit Hosts Entry dialog box.
- Enter the hostname and its IP address. If there is an alias for the hostname, enter it as well.
- Click OK to add the entry to the list.
- Choose File ➪ Save to save your changes.

5.9 SUMMARY

- TCP/IP is an acronym for Transmission Control Protocol/Internet Protocol, and refers to a family of protocols used for computer communications.

- The device address consists of two parts:
  - The address of the network to which the device is connected
  - The address of the device itself — its node or host address
- Based on how many bits are allocated to the network portion, there are 5 usable IP address classes: A, B, C, D, and E. Classes A, B, and C are widely used, while classes D and E are dedicated for multicast networks and scientific purposes only.
- Class A addresses are used for networks with an extremely large number of nodes.
- Class B addresses are used for mid-sized networks.
- Class C addresses are used for small networks not more than 254 nodes.
- The ifconfig and route commands are used to configure internal and external networks.
- In addition to configuring the network cards on each of the computers in the network, four files on each computer need to be modified. These files are all located in the /etc directory:
  - /etc/nsswitch.conf
  - /etc/hosts
  - /etc/resolv.conf
  - /etc/sysconfig/network
- Subnetting is a method by which a large network address space can be divided into several smaller and more manageable logical sub-networks, commonly referred to as subnets.
- The subnet mask segregates the network bits from the node bits. It is used by routers to identify the start and end of the network/subnet portion and the start and end of the node portion of a given IP address.
- Using Dynamic Host Configuration Protocol (DHCP), you can have an IP address and the other information automatically assigned to the hosts connected to your network.
- By using DHCP, you can ensure that every host on your network has a valid IP address, subnet mask, broadcast address, and gateway, with minimum effort on your part.
- The configuration file for the DHCP server is /etc/dhcpd.conf.
- Fedora Core and Red Hat Enterprise Linux provide a graphical network configuration tool that you can use to configure network interface devices installed in your system.

- After you add and configure your network connection device, whether it is a wired NIC, wireless NIC, or modem, you can also edit and modify the configuration.

5.10 REVIEW QUESTIONS

(1) Write a short note on network classes.
(2) How to work with Gateways and Routers?
(3) How to configure the Dynamic Host Configuration Protocol?
(4) Explain the files used for network configuration?
(5) Write a note on subnetting.
(6) Explain subnet masks and subnetting?
(7) How to configure the network using the network configuration tool?
(8) How to change the NIC configuration using the network configuration tool?

5.11 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

Beginning Linux Programming 4th Edition by Neil Mathew, Richard Stone. Wiley Publishing

Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill

Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill

Red Hat Linux Networking and System Administration 3rd Edition by Terry Collins and Kurt Wall.

Sybex – RHCE Red Hat Certified Engineer Study Guide

Red Hat Certified Technician & Engineer by Asghar Ghori.

www.thegeekstuff.com

www.tlpd.org

www.linuxtopia.org

6

THE NETWORK FILE SYSTEM

Unit Structure :

6.0 OBJECTIVES

Linux Servers are often installed to provide centralized file and print services for networks. This chapter explains how to use the Network File System (NFS) to create a file server. After a short

overview of NFS, you learn how to plan an NFS installation, how to configure an NFS server, and how to set up an NFS client. You'll learn how to mount remote file systems automatically, eliminating the need to mount remote file systems manually before you can access it. The final section of the chapter highlights NFS-related security issues.

6.1 INTRODUCTION

Network File System (NFS) is the UNIX/Linux way of sharing files and applications across the network. The NFS concept is somewhat similar to that of Microsoft Windows disk sharing, in that it allows you to attach to a disk and work with it as if it were a local drive — a handy tool for sharing files and large storage space among users.

6.2 NFS OVERVIEW

The *Network File System (NFS)* service is based on the client/server architecture whereby users on one system accesses files, directories and file systems (let us collectively call them as resources) residing on a remote system as if they exist locally on their system. The remote system that makes its resources available to be accessed over the network is called an *NFS server*, and the process of making them accessible is referred to as *exporting*. The resources exported by the NFS server can be accessed by one or more systems. These systems are called *NFS clients*, and the process of making the resources accessible on clients is referred to as *mounting*. Resources may be kept mounted until either they are unmounted manually or system is reebooted. The other method unmounts them automatically after a pre-determined time is elapsed.

6.2.1 Understanding NFS :

A system can function as both an NFS server and an NFS client at the same time. When a directory or file system resource is exported, the entire directory structure beneath it becomes available for mounting on the client. A sub-directory or parent directory of an exported resource cannot be re-exported if it exists in the same file system. Similarly, a resource mounted by an NFS client cannot be exported further by the client. NFS is built on top of *Remote Procedure Call (RPC)* and *eXternal Data Representation (XDR)* to allow a server and client to communicate. They provide a common language that both the server and client understand. This is standardized based on the facts that the NFS server and client may be running two completely different operating systems on different hardware platforms. RPC uses program numbers defined in the */etc/rpc* file.

6.2.2 Uses of NFS :

The possible uses of NFS are quite varied. NFS is often used to provide diskless clients, such as X terminals or the slave nodes in a cluster, with their entire file system, including the kernel image and other boot files. Another common scheme is to export shared data or project-specific directories from an NFS server and to enable clients to mount these remote file systems anywhere they see fit on the local system. Perhaps the most common use of NFS is to provide centralized storage for users' home directories. Many sites store users' home directories on a central server and use NFS to mount the home directory when users log in or boot their systems. Usually, the exported directories are mounted as /home/username on the local (client) systems, but the export itself can be stored anywhere on the NFS server, for example, /exports/users/username. Figure 6-1 illustrates both of these NFS uses.

The network shown in Figure 6-1 shows a server (havin the name diskbeast) with two set of NFS exports, user home directories on the file system /exports/homes and a project directory stored on a separate file system named /proj. Figure 6-1 also illustrates a number of client systems (pear, apple, mango, and so forth). Each client system mounts /home locally from diskbeast. On diskbeast, the exported file systems are stored in the /exports/homes directory. When a user logs in to a given system, that user's home directory is automatically mounted on /home/username on that system. Figure 6-1 also shows that three users, u5, u6, and u7, have mounted the project-specific file system, /proj, in various locations on their local file systems. Specifically, user u5 has mounted it as /work/proj on kiwi (that is, kiwi:/work/proj in host:/mount/dir form) u6 as lime:/projects, and u7 as peach:/home/work.

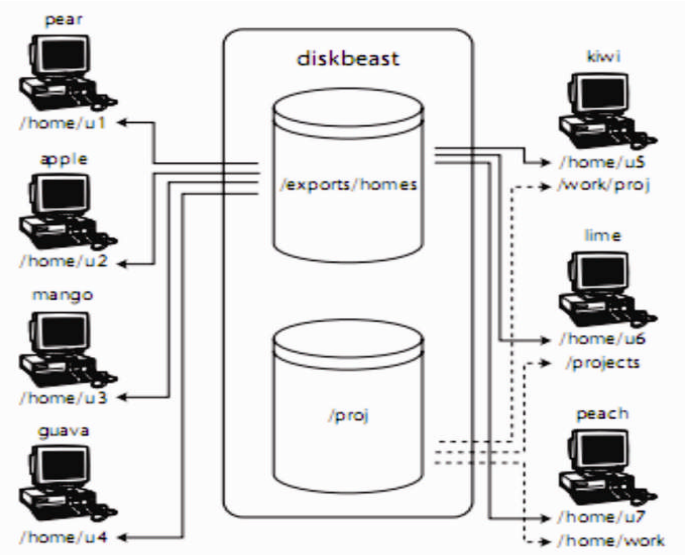


Figure 6-1: Exporting home directories and project-specific file systems.

NFS can be used in almost any situation requiring transparent local access to remote file systems. In fact, you can use NFS and NIS together to create a highly centralized network environment that makes it easier to administer the network, add and delete user accounts, protect and back up key data and file systems, and give users a uniform, consistent view of the network regardless of where they log in. As you will see in the next sections, NFS is easy to set up and maintain and pleasantly flexible. Exports can be mounted read-only or in read-write mode. Permission to mount exported file systems can be limited to a single host or to a group of hosts using either hostnames with the wildcards * and ? or using IP address ranges, or even using NIS groups, which are similar to, but not the same as, standard UNIX user groups. Other options enable strengthening or weakening of certain security options as the situation demands.

6.2.3 NFS versions

RHEL 5 comes with version 4 of NFS protocol (NFS v4), which is an Internet Engineering Task Force (IETF) standard protocol that provides enhanced security, scalability, encrypted transfers, better cross-platform interoperability, works better through firewalls and on the Internet, and is more efficient than NFS v3. NFS v4 uses usernames and groupnames rather than UIDs and GIDs when sharing files. NFS v3 is still the default protocol for NFS in RHEL 5; however, NFS v4 can be used.

6.2.4 NFS Advantages and Disadvantages

Clearly, the biggest advantage NFS provides is centralized control, maintenance, and administration. It is much easier, for example, to back up a file system stored on a single server than it is to back up directories scattered across a network, on systems that are geographically dispersed, and that might or might not be accessible when the backup is made. Similarly, NFS makes it trivial to provide access to shared disk space, or limit access to sensitive data. When NFS and NIS are used together, changes to systemwide configuration files, such as authentication files or network configuration information, can be quickly and automatically propagated across the network without requiring system administrators to physically visit each machine or requiring users to take any special action.

NFS can also conserve disk space and prevent duplication of resources. Read-only file systems and file systems that change infrequently, such as /usr, can be exported as read-only NFS mounts. Likewise, upgrading applications employed by users throughout a network simply becomes a matter of installing the new application and changing the exported file system to point at the new application. End users also benefit from NFS. When NFS is combined with NIS, users can log in from any system, even

remotely, and still have access to their home directories and see a uniform view of shared data. Users can protect important or sensitive data or information that would be impossible or time-consuming to re-create by storing it on an NFS mounted file system that is regularly backed up.

NFS has its shortcomings, of course, primarily in terms of performance and security. As a distributed, network-based file system, NFS is sensitive to network congestion. Heavy network traffic slows down NFS performance. Similarly, heavy disk activity on the NFS server adversely affects NFS's performance. In the face of network congestion or extreme disk activity, NFS clients run more slowly because file I/O takes longer. If an exported file system is not available when a client attempts to mount it, the client system can hang. Another shortcoming of NFS is that an exported file system represents a single point of failure. If the disk or system exporting vital data or application becomes unavailable for any reason, such as a disk crash or server failure, no one can access that resource. NFS suffers from potential security problems because its design assumes a trusted network, not a hostile environment in which systems are constantly being probed and attacked. The primary weakness of most NFS implementations based on protocol versions 1, 2, and 3 is that they are based on standard (unencrypted) remote procedure calls (RPC). RPC is one of the most common targets of exploit attempts. As a result, sensitive information should never be exported from or mounted on systems directly exposed to the Internet, that is, one that is on or outside a firewall. While RPCSEC_GSS makes NFSv4 more secure and perhaps safer to use on Internet-facing systems, evaluate such usage carefully and perform testing before deploying even a version 4 – based NFS system across the Internet. Never use NFS versions 3 and earlier on systems that front the Internet; clear-text protocols are trivial for anyone with a packet sniffer to intercept and interpret. Quite aside from encryption and even inside a firewall, providing all users access to all files might pose greater risks than user convenience and administrative simplicity justify. Care must be taken when configuring NFS exports to limit access to the appropriate users and also to limit what those users are permitted to do with the data.

6.2.5 How NFS works?

The following outlines the process of exporting and mounting a resource:

- The contents of */etc/exports* file are evaluated for any syntax problems and access issues.
- Each resource listed in this file is exported and an entry is added to the */var/lib/nfs/etab* file on the server. The *showmount*

command looks into this file to display exported resource information.

- The client issues the *mount* command on the NFS client to request the NFS server to provide file handle for the requested resource.
- The request goes to the *rpc.mountd* daemon on the NFS server through the *portmap* daemon that runs on both the server and the client.
- The *rpc.mountd* daemon consults TCP Wrappers and performs an access check to validate if the client is authorized to mount the resource.
- The *rpc.mountd* daemon sends a file handle for the requested resource to the client.
- The client mounts the resource if the correct *mount* command syntax is used. To automate the mount process, an entry for the resource can be added to the */etc/fstab* file, which ensures that the resource will get automatically mounted when the client reboots.
- The *mount* command tells the *rpc.mountd* daemon on the NFS server that the resource has been mounted successfully. Upon receiving a confirmation, the daemon adds an entry to the */var/lib/nfs/rmtab* file. The *showmount* command uses this file to display remotely mounted NFS resources. When the resource is unmounted on the client, the *umount* command sends a request to the *rpc.mountd* daemon to remove the entry from this file.
- The *mount* command also adds an entry to the */etc/mtab* file for the mounted resource on the client. The *mount* and *umount* commands update this file whenever they are executed successfully.
- Any file access request by the client on the mounted resource is now going to be handled by the server's *rpc.nfsd* daemon.
- The *rpc.lockd* and *rpc.statd* daemons are involved when the client requests the server to place a lock on a file.

6.3 PLANNING AN NFS INSTALLATION

Planning an NFS installation is a phrase that boils down to *thoughtful design* followed by careful *implementation*. Of these two steps, design is the more important because it ensures that the implementation is transparent to end users and trivial to the administrator. The implementation is remarkably straightforward. “*Thoughtful design*” consists of deciding what file systems to export to which users and selecting a naming convention and mounting scheme that maintains network transparency. When you are

designing your NFS installation, you need to:

- Select the file systems to export.
- Establish which users (or hosts) are permitted to mount the exported file systems.
- Identify the automounting or manual mounting scheme that clients will use to access exported file systems.
- Choose a naming convention and mounting scheme that maintains network transparency and ease of use.

With the design in place, implementation is a matter of configuring the exports and starting the appropriate daemons. Testing ensures that the naming convention and mounting scheme works as designed and identifies potential performance bottlenecks. Monitoring is an ongoing process to ensure that exported file systems continue to be available, network security and the network security policy remain uncompromised, and that heavy usage does not adversely affect overall performance. A few general rules exist to guide the design process. The following tips and suggestions for designing an NFS server and its exports will simplify administrative tasks and reduce user confusion:

- Good candidates for NFS exports include any file system that is shared among a large number of users, such as /home, workgroup project directories, shared data directories, such as /usr/share, the system mail spool (/var/spool/mail), and file systems that contain shared application binaries and data.
- Use /home/username to mount home directories. This is one of the most fundamental directory idioms in the Linux world, so disregarding it not only antagonizes users but also breaks a lot of software that presumes user home directories live in /home. On the server, you have more leeway about where to situate the exports. Recall from Figure 6-1, for example, that diskbeast stored user home directories in /exports/home.
- Few networks are static, particularly network file systems, so design NFS servers with growth in mind. For example, avoid the temptation to drop all third-party software onto a single exported file system. Over time, file systems usually grow to the point that they need to be subdivided, leading to administrative headaches when client mounts must be updated to reflect a new set of exports. Spread third-party applications across multiple NFS exports and export each application and its associated data separately.
- If the previous tip will result in a large number of NFS mounts for clients, it might be wiser to create logical volume sets on the NFS server. By using logical volumes underneath the exported file systems, you can increase disk space on the exported file

systems as it is needed without having to take the server down or take needed exports offline.

- At large sites, distribute multiple NFS exports across multiple disks so that a single disk failure will limit the impact to the affected application. Better still, to minimize downtime on singleton servers, use RAID for redundancy and logical volumes for flexibility. If you have the capacity, use NFSv4's replication facilities to ensure that exported file systems remain available even if the primary NFS server goes up in smoke.
- Similarly, overall disk and network performance improves if you distribute exported file systems across multiple servers rather than concentrate them on a single server. If it is not possible to use multiple servers, at least try to situate NFS exports on separate physical disks and/or on separate disk controllers. Doing so reduces disk I/O contention.
- When identifying the file systems to export, keep in mind a key restriction on which file systems can be exported and how they can be exported. You can export only local file systems and their subdirectories. To express this restriction in another way, you cannot export a file system that is itself already an NFS mount.

6.4 CONFIGURING AN NFS SERVER

This section shows you how to configure an NFS server, identifies the key files and commands you use to implement, maintain, and monitor the NFS server, and illustrates the server configuration process using a typical NFS setup. On Fedora Core and Red Hat Enterprise Linux systems, the /etc/exports file is the main NFS configuration file. It lists the file systems the server exports, the systems permitted to mount the exported file systems, and the mount options for each export. NFS also maintains status information about existing exports and the client systems that have mounted those exports in /var/lib/nfs/rmtab and /var/lib/nfs/xtab.

When working with NFS, several daemons, commands, configuration files and scripts are involved. The tables given below list and explain them.

NFS Server Configuration and Status Files

File	Description
/etc/exports	Server-side file that contains a list of resources to be exported.
/var/lib/nfs/etab	Server-side file that contains a list of exported resources whether or not they are

	remotely mounted. This file is updated when a resource is exported or unexported, and is maintained by the rpc.mountd daemon.
/var/lib/nfs/rmtab	Server-side file that contains a list of exported resources, which have been mounted by clients. This file is updated when a resource is remotely mounted or unmounted, and is maintained by the rpc.mountd daemon.
/etc/fstab	Client-side file that contains a list of resources to be mounted at system reboots or manually with the mount command.
/etc/mtab	Client-side file that contains a list of mounted resources. The mount and umount commands update this file.
/etc/sysconfig/nfs	Server- and client-side configuration file used by NFS startup scripts.
/etc/idmapd.conf	Server- and client-side configuration file used to translate NFSv4 IDs to user and group IDs and vice versa.
/etc/gssapi_mech.conf	Controls GSS daemon (rpc.svcgssd). Lists the specific function call used to initialize a given GSS library.

Table 6-1: NFS Configuration and Functional Files

The server configuration file is */etc/exports*, which contains a list of file systems to export, the clients permitted to mount them, and the export options that apply to client mounts. Each line in */etc/exports* has the following format:

dir [host](options) […]

dir specifies a directory or file system to export, *host* specifies one or more hosts permitted to mount *dir*, and *options* specifies one or more mount options. If you omit *host*, the listed options apply to every possible client system, likely not something you want to do.
Do not insert a space between the hostname and the opening parenthesis that contains the export options; a space between the hostname and the opening parenthesis of the option list has four (probably unintended) consequences:

(1) Any NFS client can mount the export.

(2) You'll see an abundance of error messages in /var/log/messages.

(3) The list options will be applied to all clients, not just the client(s) identified by the host specification.

(4) The client(s) identified by the host specification will have the default mount options applied, not the mount options specified by options.

host can be specified as a single name, an NIS netgroup, a subnet using address/net mask form, or a group of hostnames using the wildcard characters ? and *. Multiple *host(options)* entries, separated by whitespace, are also accepted, enabling you to specify different export options for a single dir depending on the client. Consider the following sample */etc/exports* file:

```
/usr/local          *.example.com(ro)
/usr/devtools             192.168.1.0/24(ro)
/home               192.168.0.0/255.255.255.0(rw)
/projects           @dev(rw)
/var/spool/mail     192.168.0.1(rw)
/opt/kde
gss/krb5(ro)
```

The first line permits all hosts with a name of the format *somehost.example.com* to mount */usr/local* as a read-only directory. The second line uses the *address/net mask* form in which the net mask is specified in Classless Inter-Domain Routing (CIDR) format. In the CIDR format, the net mask is given as the number of bits (*/24*, in this example) used to determine the network address. The third line permits any host with an IP address in the range 192.168.0.1 to 192.168.0.254 to mount */home* in read-write mode. This entry uses the *address/net* mask form in which the net mask is specified in dotted quad format. The fourth line permits any member of the NIS netgroup named *dev* to mount */projects* (again, in read-write mode). The fifth line permits only the host whose IP address is 192.168.0.1 to mount */var/mail*. The final line allows any host using RPCSEC_GSS security to mount */opt/kde* in read-only mode.

The export options, listed in parentheses after the host specification, determine the characteristics of the exported file system. Table 6-2 lists valid values for options.

Option	Description
*	Represents all possible matches. In the examples above, * means that any client can mount the exported resource and *.getitcertify.com (or .getitcertify.com) means that any client system on the getitcertify.com domain can mount the resource. You can specify one or more hostnames, IP addresses, domain names or network addresses, or a combination.
all_squash / no_all_squash (default=no_all_squash)	all_squash treats users on client systems as anonymous users and no_all_squash does not.
anongid=GID (default=65534)	Assigns the GID to anonymous groups.
anonuid=UID (default=65534)	Assigns the UID to anonymous users.
mp	Exports only if the specified directory is a file system.
root_squash / no_root_squash (default=root_squash)	root_squash prevents *root* users on an NFS client from gaining *root* access on a mounted NFS resource by mapping *root* to a special, unprivileged user called *nfsnobody* with UID 65534. no_root_squash allows *root* access. It is recommended to use the default to prevent unauthorized *root* access on clients.
rw / ro (default=ro)	rw (read/write) allows file modifications and ro (read-only) prevents doing it.
secure / insecure (default=secure)	secure allows access on ports lower than 1024 and insecure allows access on ports beyond 1024.
secure_locks / insecure_locks (default=secure_locks)	secure_locks checks file permissions on older NFS clients and insecure_locks does not.
subtree_check / no_subtree_check (default=no_subtree_check)	Enables / disables permission checking on higher level directories of an exported resource.
sync / async (default=sync)	Changes are written to disk before / after a command is complete.
wdelay / no_wdelay (default=wdelay)	wdelay delays data writes to the resource and no_wdelay writes data right away.

Table 6-2: export Options.

If you intend to use NFSv4-specific features, you need to be familiar with the RPCSEC_GSS configuration files, /etc/gssapi_mech.conf and /etc/idmapd.conf. idmapd.conf is the configuration file for NFSv4's idmapd daemon. idmapd works on the behalf of both NFS servers and clients to translate NFSv4 IDs to user and group IDs and vice versa; idmapd.conf controls idmapd's runtime behavior. The default configuration (with comments and blank lines removed) should resemble Figure 6-2.

```
 [General]
Verbosity = 0
Pipefs-Directory = /var/lib/nfs/rpc_pipefs
Domain = localdomain
[Mapping]
Nobody-User = nobody
Nobody-Group = nobody
[Translation]
Method = nsswitch
```

Figure 6-2: Default idmapd configuration.

In the *[General]* section, the *Verbosity* option controls the amount of log information that *idmapd* generates; *Pipefs-directory* tell *idmapd* where to find the RPC pipe file system it should use (*idmapd* communicates with the kernel using the pipefs virtual file system); *Domain* identifies the default domain. If *Domain* isn't specified, it defaults to the server's fully qualified domain name (FQDN) less the hostname. The *[Mapping]* section identifies the user and group names that correspond to the nobody user and group that NFS server should use. The option *Method* = *nsswitch*, finally, tells *idmapd* how to perform the name resolution. In this case, names are resolved using the name service switch (NSS) features of glibc.

The */etc/gssapi_mech.conf* file controls the GSS daemon (rpc.svcgssd). You won't need to modify this file. As provided in Fedora Core and RHEL, *gssapi_mech.conf* lists the specific function call to use to initialize a given GSS library. Programs (in this case, NFS) need this information if they intend to use secure RPC.

Two additional files store status information about NFS exports, */var/lib/nfs/rmtab* and */var/lib/nfs/etab*. */var/lib/nfs/rmtab* is the table that lists each NFS export that is mounted by an NFS client. The daemon *rpc.mountd* is responsible for servicing requests to mount NFS exports. Each time the *rpc.mountd* daemon receives a mount request, it adds an entry to */var/lib/nfs/rmtab*. Conversely, when mountd receives a request to unmount an exported file system, it removes the corresponding entry from */var/lib/nfs/rmtab*. The following short listing shows the contents of */var/lib/nfs/rmtab* on an NFS server that exports */home* in read-write mode and */usr/local* in read-only mode. In this case, the host with IP address 192.168.0.4 has mounted both exports:

```
$ cat /var/lib/nfs/rmtab
192.168.0.4:/home:0x00000001
192.168.0.4:/usr/local:0x00000001
```

Fields in *rmtab* are colon-delimited, so it has three fields: the host, the exported file system, and the mount options specified in */etc/exports*. Rather than try to decipher the hexadecimal options field, though, you can read the mount options directly from */var/lib/nfs/etab*. The *exportfs* command, maintains */var/lib/nfs/etab*. *etab* contains the table of currently exported file systems. The following listing shows the contents of */var/lib/nfs/etab* for the server exporting the */usr/local* and */home* file systems shown in the previous listing (the output wraps because of page width constraints).

```
$ cat /var/lib/nfs/etab
/usr/local
192.168.0.4(ro,sync,wdelay,hide,secure,root_squash,no_all_sq
```

uash,subtree_check,secure_locks,mapping=identity,anonuid=-2,anongid=-2)
/home
192.168.0.2(rw,sync,wdelay,hide,secure,root_squash,no_all_squash,subtree_check,secure_locks,mapping=identity,anonuid=-2,anongid=-2)

As you can see in the listing, the format of the etab file resembles that of */etc/exports*. Notice, however, that *etab* lists the default values for options not specified in */etc/exports* in addition to the options specifically listed.

The last two configuration files to discuss, */etc/hosts.allow* and */etc/hosts.deny*, are not, strictly speaking, part of the NFS server. Rather, */etc/hosts.allow* and */etc/hosts.deny* are access control files used by the TCP Wrappers system; you can configure an NFS server without them and the server will function perfectly. However, using TCP Wrappers' access control features helps enhance both the overall security of the server and the security of the NFS subsystem.

6.4.1 NFS Server Daemons

Providing NFS services requires the services of six daemons: /sbin/portmap, /usr/sbin/rpc.mountd, /usr/sbin/rpc.nfsd, /sbin/rpc.statd, /sbin/rpc.lockd, and, if necessary, /usr/sbin/rpc.rquotad. They are generally referred to as portmap, mountd, nfssd, statd, lockd, and rquotad, respectively. If you intend to take advantage of NFSv4's enhancements, you'll also need to know about rpc.gssd, rpc.idmapd, and rpc.svcgssd. Table 6-3 briefly describes each daemon's purpose.

Daemon	Description
portmap	Server-and-client side daemon responsible for forwarding incoming RPC requests to appropriate RPC daemons. Access to this daemon can be controlled via TCP wrappers using /etc/hosts.allow and /etc/hosts.deny files.
rpc.idmapd	Server-and-client side daemon that controls mappings of UIDs and GIDs with their corresponding usernames and groupnames. Its configuration file is /etc/idmapd.conf.
rpc.lockd	Server-and-client side daemon that keeps an eye on the NFS client that has requested a lock on files to make sure the client is up and running. If the clent is rebooted unexpectedly,

	this daemon removes all locks placed on the files so that other NFS clients may use them.
rpc.mountd	Server side daemon that responds to client requests to mount a resource and provide status of exported and mounted resources. Access to this daemon can be controlled via TCP wrappers using /etc/hosts.allow and /etc/hosts.deny files.
rpc.nfsd	Server-side daemon that responds to client requests to access files.
rpc.rquotad	Server-side daemon that provides statistics on disk quota to clients.
rpc.statd	Server-and-client side daemon that works with rpc.lockd tp provide crash and recovery services.
rpc.gssd	Creates security contexts on RPC clients for exchanging RPC information using SecureRPC (RPCSEC) using GSS
rpc.svcgssd	Creates security contexts on RPC servers for exchanging RPC information using SecureRPC (RPCSEC) using GSS

Table 6-3: NFS Daemons

The NFS server daemons should be started in the following order to work properly:

(1) portmap
(2) nfsd
(3) mountd
(4) statd
(5) rquotad (if necessary)
(6) idmapd
(7) svcgssd

The start order is handled for you automatically at boot time if you have enabled NFS services using Service Configuration Tool (/usr/bin/system-config-services). Notice that the list omits lockd. nfsd starts it on an as-needed basis, so you should rarely, if ever, need to invoke it manually.

6.4.2 NFS Server Scripts and Commands

Three initialization scripts control the required NFS server daemons, */etc/rc.d/init.d/portmap*, */etc/rc.d/init.d/nfs*, and */etc/rc.d /init.d/nfslock*. The *exportfs* command enables you to manipulate the list of current exports on the fly without needing to edit */etc/exports*. The *showmount* command provides information about clients and the file systems they have mounted. The *nfsstat* command displays detailed information about the status of the NFS subsystem. The *portmap* script starts the *portmap* daemon, frequently referred to as the *portmapper*. All programs that use RPC, such as NIS and NFS, rely on the information the portmapper provides. The portmapper starts automatically at boot time, so you rarely need to worry about it, but it is good to know you can control it manually. The primary NFS startup script is */etc/rc.d/init.d/nfs*. It requires a single argument, *start*, *stop*, *status*, *restart*, or *reload*. start and stop start and stop the NFS server, respectively. NFS services also require the file-locking daemons *lockd* and *statd*. As explained earlier, *nfsd* starts *lockd* itself, but you still must start *statd* separately. You can use an initialization script for this purpose, */etc/rc.d/init.d/nfslock*. It accepts almost the same arguments as */etc/rc.d/init.d/nfs* does, with the exception of the reload argument (because *statd* does not require a configuration file).

You can also find out what NFS daemons are running using the *rpcinfo* command with the **-p** option. *rpcinfo* is a general-purpose program that displays information about programs that use the RPC protocol, of which NFS is one. The **-p** option queries the portmapper and displays a list of all registered RPC programs.

The various commands are listed below in Table 6-4.

Command	Description
exportfs	Server-side command that exports resources listed in the */etc/exports* file and displays exported resources listed in the */var/lib/nfs/etab* file.
showmount	Server- and client-side command that displays which resources are exported to which clients by consulting the */var/lib/nfs/etab* file and displays which clients have those resources mounted by consulting the */var/lib/nfs/rmtab* file.
mount	Client-side command that mounts a resource specified at the command line or listed in the */etc/fstab* file followed by adding an entry to client's */etc/mnttab* file, and server's */var/lib/nfs/rmtab* file via the *rpc.mountd* daemon. It also displays mounted resources listed in the */etc/mnttab* file.
umount	Client-side command that unmounts a single resource specified at the command line or listed in the */etc/mnttab* file followed by removing corresponding entry from this file and server's */var/lib/nfs/rmtab* file via the *rpc.mountd* daemon.
rpcinfo	Server-side command that checks whether NFS server daemons are registered with RPC.
nfsstat	Server- and client-side command that displays NFS and RPC statistics.

Table 6-4: NFS Commands

6.4.3 Using Secure NFS

Although NFSv4 is installed, the default installation does not use NFSv4's security enhancements by default. You need to set

this up manually. To do so, use the following procedure:

(1) Enable secure NFS by adding the following line to */etc/sysconfig/nfs*:
SECURE_NFS=no

(2) Edit */etc/idmapd.conf* and set the *Domain* option to your domain and change the *Nobody-User* and *Nobody-Group* options to *nobody*:
Domain = example.com
[Mapping]
Nobody-User = nobody
Nobody-Group = nobody

(3) Restart the *portmapper* and NFS using the service utility:
 # service portmap restart
 # service nfs condrestart

You do not need to start the GSS client and server daemons, rpcgssd and rpcsvcgssd, respectively, unless you wish to use Kerberos 5 or another strong encryption mechanism (in which case there is additional setup to perform that this chapter does not address).

6.4.4 Using the NFS Server Configuration Tool

If you prefer to use graphical tools for system administration, Red Hat Enterprise Linux includes the NFS Server Configuration tool. It edits the */etc/exports* file directly, so you can use the graphical tool and edit the configuration file directly using a text editor interchangeably. To start the NFS Server Configuration tool, select Red Hat ➪ System Settings ➪ Server Settings ➪ NFS on Fedora Core or Applications ➪ System Settings ➪ Server Settings ➪ NFS on RHEL. You can also start the tool by executing the command system-config-nfs (as root) in a terminal window. Figure 6-3 shows the NFS Server Configuration tool.

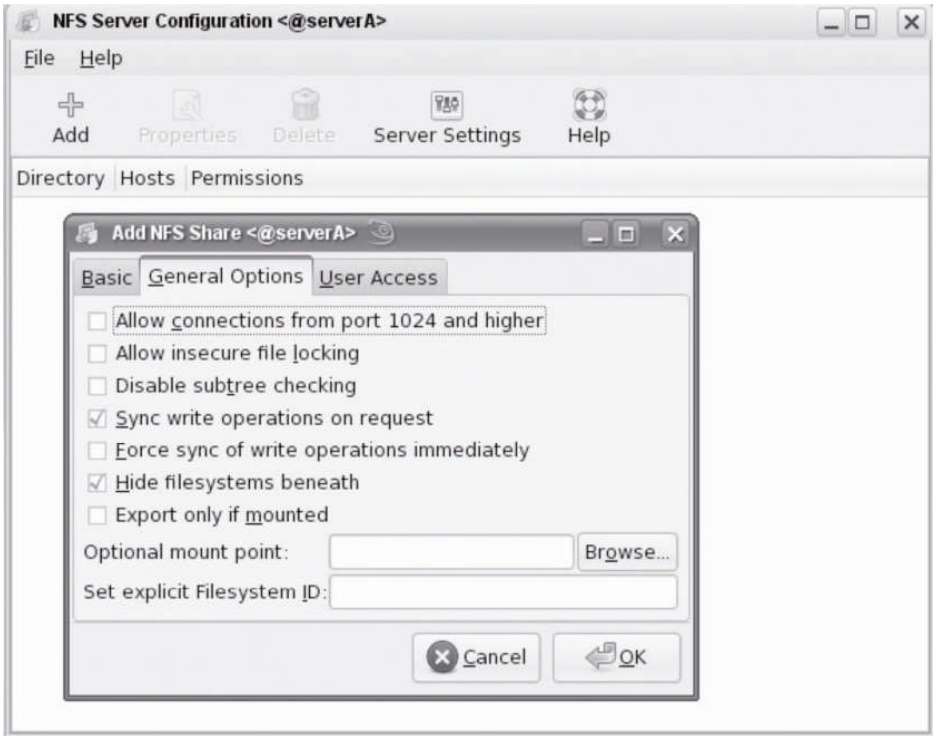


Figure 6-3: NFS Server Configuration Tool Dialog Box.

There are three tabs – Basic, General Options and User Access – on the main screen. The Basic tab is where you input a resource name to be exported, hostname or IP address of the server, domain or network to be exported to (or an * for all hosts) and permissions. The General Options and User Access tabs allow you to modify some of the options listed in table 6-2.

6.5 Configuring an NFS Client

Configuring client systems to mount NFS exports is simpler than configuring the NFS server itself. Configuring a client system to use NFS involves making sure that the *portmapper* and the NFS file locking daemons *statd* and *lockd* are available, adding entries to the client's */etc/fstab* for the NFS exports, and mounting the exports using the *mount* command. As a networked file system, NFS is sensitive to network conditions, so the NFS client daemons accept a few options, passed via the *mount* command, address NFS's sensitivities and peculiarities. Table 6-5 lists the major NFS-specific options that mount accepts. For a complete list and discussion of all NFS-specific options, see the NFS manual page (***man nfs***).

Option	Description
async / sync (default=async)	Changes are written to disk after / before a command is complete.
fg / bg (default=fg)	Use "fg" (foreground) for resources that must be available to a client to boot successfully or operate properly. If a foreground mount fails, it is retried for "retry" minutes in the foreground until it either succeeds or is interrupted if "intr" option is also used. With "bg" (background), mount attempts are tried and retried for "retry" minutes in the background without hampering the system boot process or hanging the client.
hard / soft (default=hard)	With "hard" option, the client tries and retries to mount a resource until either it succeeds or is interrupted if "intr" option is also used. If the server goes down, processes using the mounted resource hang until the server comes back up. Use "soft" to avoid this situation. With this option, if a client attempts to mount a resource for "retrans" times unsuccessfully, an error message is displayed.
intr / nointr (default=nointr)	Use "intr" (interrupt) if you want to be able to manually interrupt a request. Use "nointr" for the opposite.
nfsvers=*n* (default=3)	NFS version to be used.
retrans=*n* (default=3)	This many times a client retransmits a read or write request after the first transmission times out. If the request does not succeed after *n* retransmissions, a "soft" mount displays an error message and a "hard" mount continues to retry.
retry=*n* (default=2 minutes and 10000 minutes)	For this duration (2 minutes with "fg" and 10,000 minutes with "bg") a client tries to mount a resource after the first try fails. With "intr", mount attempts can be interrupted before this many retries. If "nointr" is used, mount attempts must wait until this many retries have been made, mount succeeds or the client is rebooted.
rsize=*n* (default=32k)	Size of each read request from client to server.
rw / ro (default=rw)	"rw" (read/write) allows file modifications and "ro" (read-only) prevents from doing it.
suid / nosuid (default=suid)	"suid" enables users on a client to execute a setuid-enabled program located on the NFS mounted resource with the same privileges as the owner of the program has on it. If the program has *root* ownership, it is executed with *root* privileges regardless of who runs it. "nosuid" prevents users from running setuid programs.
timeo=*n* (default=7)	Sets timeout, in tenths of a second, for NFS read and write requests. If a request times out, this value is doubled and the request is attempted again for "retrans" times. When number of "retrans" attempts are made, a "soft" mount displays an error message while a "hard" mount continues to retry.
wsize=*n* (default=32k)	Size of each write request from client to server.

Table 6-5: mount command options for NFS.

The options you are most likely to use are *rsize*, *wsize*, *hard*, *intr*, and *nolock*. Increasing the default size of the NFS read and write buffers improves NFS's performance. The suggested value is 8192 bytes, that is, *rsize=8192* and *wsize=8192*, but you might find that you get better performance with larger or smaller values. The *nolock* option can also improve performance because it eliminates the overhead of file locking calls, but not all servers support file locking over NFS. If an NFS file operation fails, you can use a keyboard interrupt, usually **Ctrl+C**, to interrupt the operation if the exported file system was mounted with both the *intr* and *hard* options. This prevents NFS clients from hanging.

Like an NFS server, an NFS client needs the *portmapper* running in order to process and route RPC calls and returns from the server to the appropriate port and programs. Accordingly, make sure that the portmapper is running on the client system using the portmap initialization script:

 # service portmap status

To use NFS file locking, both an NFS server and any NFS clients need to run *statd* and *lockd.* As explained in the section on configuring an NFS server, the simplest way to accomplish this is to use the initialization script, */etc/rc.d/init.d/nfslock*. Presumably, you have already started nfslock on the server, so all that remains is to start it on the client system:

 # service nfslock start

Once you have configured the mount table and started the requisite daemons, all you need to do is mount the file systems using the mount command with the requisite options.

6.5.1 Configuring an NFSv4 Client

The introduction of NFSv4 into the kernel added some NFSv4-specific behaviour of which you need to be aware and changed some of the mount options. Table 6-6 lists the new or changed mount options. The two new options listed in Table 6-6 are *clientaddr* and *proto*. Version 3 of NFS introduced NFS over TCP, which improved NFS's reliability over the older UDP-based implementation. Under NFSv3, you would use the mount option *tcp* or *udp* to specify to the client whether you wanted it to use TCP or UDP to communicate with the server. NFSv4 replaces *tcp* and *udp* with a single option, ***proto=*** that accepts two arguments: *tcp* or *udp*. In case it isn't clear, the NFSv3 option *tcp* is equivalent to the NFSv4 option *proto=tcp*. Figuring out the udp option is left as an exercise for the reader.

OPTION	DESCRIPTION
`clientaddr=n`	Causes a client on a multi-homed system to use the IP address specified by `n` to communicate with an NFSv4 server.
`proto=type`	Tells the client to use the network protocol specified by `type`, which can be `tcp` or `udp` (the default is `udp`); this option replaces the `tcp` and `udp` options from earlier versions of NFS.
`rsize=n`	Sets the read buffer size to `n` bytes (the default for NFSv4 is `8192`); the maximum value is `32678`.
`sec=mode`	Set the security model to `mode`, which can be `sys`, `krb5`, `krb5i`, or `krb5p`.
`wsize=n`	Sets the write buffer size to `n` bytes (the default for NFSv4 is `8192`); the maximum value is `32678`.

Table 6-6: NFSv4 specific mount options.

6.6 USING AUTOMOUNT SERVICES

The easiest way for client systems to mount NFS exports is to use *autofs*, which automatically mounts file systems not already mounted when the file system is first accessed. *autofs* is the NFS client-side service, which automatically mounts an NFS resource on an as-needed basis. When an activity occurs in the mount point, the associated NFS resource gets mounted. When the resource is no longer accessed for a pre-defined period of time, it automatically gets mounted. *autofs* uses the *automount* daemon to mount and unmount file systems that automount has been configured to control. Although slightly more involved to configure than the other methods for mounting NFS file systems, *autofs* setup has to be done only once.

6.6.1 How autofs works?

autofs service consists of a daemon called *automount* that mounts configured resources automatically when accessed. This daemon is invoked at system boot up. It reads the autofs master map and creates initial mount point entries in the */etc/mtab* file; however, the resources are not actually mounted at this time. When a user activity occurs under one of the initial mount points, the daemon contacts the *rpc.mountd* daemon on the NFS server and actually mounts the requested resource. If the resource remains idle for a certain time period, *automount* unmounts it by itself. autofs uses RPC and its daemon is stateless and multi-threaded.

6.6.2 Configuring automount services

autofs uses a set of map files to control automounting. A master map file, */etc/auto.master*, associates mount points with secondary map files. The secondary map files, in turn, control the file systems mounted under the corresponding mount points. For example, consider the following */etc/auto.master autofs* configuration file:

```
/home          /etc/auto.home
/var      /etc/auto.var          --timeout 600
```

This file associates the secondary map file */etc/auto.home* with the mount point */home* and the map file */etc/auto.var* with the */var* mount point. Thus, */etc/auto.home* defines the file systems mounted under */home*, and */etc/auto.var* defines the file systems mounted under */var*. Each entry in */etc/auto.master*, what we'll refer to as the master map file, consists of at least two and possibly three fields. The first field is the mount point. The second field identifies the full path to the secondary map file that controls the map point. The third field, which is optional, consists of options that control the behavior of the *automount* daemon. In the example master map file, the *automount* option for the */var* mount point is **--timeout 600**,

which means that after 600 seconds (10 minutes) of inactivity, the */var* mount point will be unmounted automatically. If a timeout value is not specified, it defaults to 300 seconds (5 minutes).

The secondary map file defines the mount options that apply to file systems mounted under the corresponding directory. Each line in a secondary map file has the general form:

localdir [-[options]] remotefs

localdir refers to the directory beneath the mount point where the NFS mount will be mounted. *remotefs* specifies the host and pathname of the NFS mount. *remotefs* is specified using the *host:/path/name* format described in the previous section. *options*, if specified, is a comma-separated list of mount options. These options are the same options you would use with the *mount* command.

Finally, as the root user, start the autofs service:

 # /sbin/service autofs start

After starting the *autofs* service, you can use the status option to verify that the *automount* daemon is working:

 # /sbin/service autofs status

You can also see the *automount* daemon at work by using the *mount* command:

 # mount -t autofs

One of the handiest features of the *autofs* service is that changes made to the secondary map files go into effect almost immediately. The next time that a directory or file system managed by *autofs* is accessed, the automounter rereads the secondary map files. So, changes to the secondary map files do not require any special treatment. However, if you modify the master map file, you have to reload the configuration file using the following command:

/sbin/service autofs reload

6.7 EXAMINING NFS SECURITY

As explained at the beginning of the chapter, NFS protocol versions 3 and older have some inherent security problems that make it unsuitable for use across the Internet and potentially unsafe for use even in a trusted network. This section identifies key security issues of NFS in general and the security risks specific to an NFS server and to NFS clients and suggests remedies that minimize your network's exposure to these security risks. Be forewarned, however, that no list of security tips, however comprehensive, makes your site completely secure. Nor will plugging possible NFS security holes address other potential exploits.

6.7.1 General NFS Security Issues

One NFS weakness, in general terms, is the */etc/exports* file. If a cracker is able to spoof or take over a trusted address, an address listed in */etc/exports*, your exported NFS mounts are accessible. Another NFS weak spot is normal Linux file system access controls that take over once a client has mounted an NFS export: Once an NFS export has been mounted, normal user and group permissions on the files take over access control.

The first line of defense against these two weaknesses is to use host access control to limit access to services on your system, particularly the portmapper, which has long been a target of exploit attempts. Similarly, you should add entries in */etc/hosts.deny*, *lockd*, *statd*, *mountd,* and *rquotad*. More generally, judicious use of IP packet firewalls, using *netfilter*, dramatically increases NFS server security. *netfilter* is stronger than NFS daemon-level security or even TCP Wrappers because it restricts access to your server at the packet level. First, you need to know the ports and services NFS uses so that you know where to apply the packet filters. Table 6-7 lists the ports and protocols each NFS daemon (on both the client and server side) use.

Service	Port	Protocol
portmap	111	TCP, UDP
nfsd	2049	TCP, UDP
mountd	variable	TCP, UDP
lockd	variable	TCP, UDP
statd	variable	TCP, UDP
rquotad	variable	UDP

Table 6-7: NFS Ports and Network Protocols.

Note that *mountd*, *lockd*, *statd*, and *rquotad* do not bind to any specific port; that is, they use a port number assigned randomly by the portmapper (which is one of portmapper's purposes in the first place). The best way to address this variability is to assign each daemon a specific port using the portmapper's **-p** option and then to apply the packet filter to that port.

Regardless of how you configure your firewall, you must have the following rule:

iptables -A INPUT -f -j ACCEPT

This rule accepts all packet fragments except the first one (which is treated as a normal packet) because NFS does not work correctly unless you let fragmented packets through the firewall.

6.7.2 Server Security Considerations

On the server, always use the **root_squash** option in */etc/exports*. NFS helps you in this regard because root squashing is the default, so you should not disable it (with **no_root_squash**) unless you have an extremely compelling reason to do so, such as needing to provide boot files to diskless clients. With root squashing in place, the server substitutes the UID of the anonymous user for root's UID/GID (0), meaning that a *client's* root account cannot change files that only the *server's* root account can change. A user with root access on a client can usually ***su*** to any user, and that UID will be used over NFS. Without **all_squash**, a compromised client can at least view and, if the file system is mounted read-write, update files owned by any user besides root if *root_squash* is enabled. This security hole is closed if the **all_squash** option is used.

NFS also helps you maintain a secure server through the *secure* mount option; because this mount option is one of the default options *mountd* applies to all exports unless explicitly disabled using the *insecure* option. Ports 1–1024 are reserved for root's use; merely mortal user accounts cannot bind these ports. Thus, ports 1–1024 are sometimes referred to as privileged or secure ports. The *secure* option prevents a malevolent nonroot user from initiating a spoofed NFS dialog on an *unprivileged* port and using it as a launch point for exploit attempts.

6.7.3 Client Security Considerations

On the client, disable SUID (set UID) root programs on NFS mounts using the *nosuid* option. The *nosuid* mount option prevents a server's root account from creating an SUID root program on an exported file system, logging in to the client as a normal user, and then using the UID root program to become root on the client. In some cases, you might also disable binaries on mounted file systems using the *noexec* option, but this effort almost always proves to be impractical or even counterproductive because one of the benefits of NFS is sharing file systems, such as */usr* or */usr/local*, that contain scripts or programs that need to be executed.

NFS versions 3 and 4 support NFS file locking. Accordingly, NFS clients must run *statd* and *lockd* in order for NFS file locks to function correctly. *statd* and lockd, in turn, depend on the *portmapper*, so consider applying the same precautions for *portmap*, *statd*, and *lockd* on NFS clients that were suggested for the NFS server.

In summary, using TCP wrappers, the secure, *root_squash*, and *nosuid* options, and sturdy packet filters can increase the overall security of your NFS setup. However, NFS is a complex, nontrivial subsystem, so it is entirely conceivable that new bugs and exploits will be discovered.

6.8 Summary

- Network File System (NFS) is the UNIX/Linux way of sharing files and applications across the network.
- The Network File System (NFS) service is based on the client/server architecture whereby users on one system accesses resources residing on a remote system as if they exist locally on their system.
- The remote system that makes its resources available to be accessed over the network is called an NFS server, and the process of making them accessible is referred to as exporting.
- The resources exported by the NFS server can be accessed by one or more systems, called NFS clients, and the process of making the resources accessible on clients is referred to as mounting.
- The possible uses of NFS are quite varied, like:
    - to provide diskless clients, such as X terminals or the slave nodes in a cluster, with their entire file system
    - to export shared data or project-specific directories from an NFS server
    - to provide centralized storage for users' home directories
- Planning an NFS installation requires thoughtful design followed by careful implementation.
- "Thoughtful design" consists of deciding what file systems to export to which users and selecting a naming convention and mounting scheme that maintains network transparency.
- When working with NFS, several daemons, commands, configuration files and scripts are involved. They are as follows:
    - Daemons
        - rpc.gssd (new in NFSv4)
        - rpc.idmapd (new in NFSv4)
        - rpc.lockd
        - rpc.mountd

    - rpc.nfsd
    - rpc.portmap
    - rpc.statd
    - rpc.rquotad
    - rpc.svcgssd (new in NFSv4)
    - Configuration files (in /etc)
    - gssapi_mech.conf (new in NFSv4)
    - exports
    - idmapd.conf (new in NFSv4)
    - /var/lib/nfs/etab
    - /var/lib/nfs/rmtab
    - Initialization scripts (in /etc/rc.d/init.d)
    - rpcgssd (new in NFSv4)
    - rpcidmapd (new in NFSv4)
    - nfs
    - rpcsvcgssd (new in NFSv4)
    - Commands
    - exportfs
    - nfsstat
    - showmount
    - rpcinfo
- The server configuration file is /etc/exports, which contains a list of file systems to export, the clients permitted to mount them, and the export options that apply to client mounts.
- NFSv4 is not used by default. you need to manually configure it and edit the file /etc/idmapd.conf.
- If you prefer to use graphical tools for system administration, Red Hat Enterprise Linux includes the NFS Server Configuration tool. It edits the /etc/exports file directly, so you can use the graphical tool and edit the configuration file directly.
- Configuring a client system to use NFS involves making sure that the portmapper and the NFS file locking daemons statd and lockd are available, adding entries to the client's /etc/fstab for the NFS exports, and mounting the exports using the mount command.
- autofs is the NFS client-side service, which automatically

mounts an NFS resource on an as-needed basis.

- NFS protocol versions 3 and older have some inherent security problems that make it unsuitable for use across the Internet and potentially unsafe for use even in a trusted network.
- Using TCP wrappers, the secure, root_squash, and nosuid options, and sturdy packet filters can increase the overall security of your NFS setup.

6.9 REVIEW QUESTIONS

(1) Explain the uses of NFS?
(2) What are the advantages and disadvantages of NFS?
(3) Explain how NFS works?
(4) What are the steps involved in planning a NFS installation?
(5) Explain the files involved in configuring a NFS server?
(6) Write a short note on NFS daemons?
(7) Write a short note on /etc/exports.
(8) How will you configure a server using server configuration tool?
(9) Write a short note on mount command.

6.10 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

Beginning Linux Programming 4th Edition by Neil Mathew, Richard Stone. Wiley Publishing

Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill

Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill

Red Hat Linux Networking and System Administration 3rd Edition by Terry Collins and Kurt Wall.

Sybex – RHCE Red Hat Certified Engineer Study Guide

Red Hat Certified Technician & Engineer by Asghar Ghori.

www.thegeekstuff.com

www.tlpd.org

www.linuxtopia.org

7

CONNECTING TO MICROSOFT NETWORKS

Unit Structure

7.1 INTRODUCTION

Samba is a software package or bundle that provides file and print sharing services to SMB clients. Samba is freely available tool in Linux and UNIX and it well supports Linux/Unix and Windows clients. Samba allows other computer platforms, such as Mac OS, UNIX, Linux, IBM System 390 with Windows computers on the same network. Samba includes sharing files and sharing devices such as printer connected to other computers within the local network.

In this example we will configure a **samba** server and will transfer files from client side .For this example we are using two systems one linux server one window clients per quest of samba server

- A linux server with ip address 192.168.0.254 and hostname Server
- A window client with ip address 192.168.0.2 and hostname Client2
- Updated /etc/hosts file on linux system
- Running portmap and xinetd services
- Firewall should be off on server

7.1.1 Installing Samba.

Computers running Windows 95 or higher version of Windows use a protocol called Server Message Block (SMB) to communicate with each other and to share services such as file and print sharing.By using Samba,the Linux PC icon appears in the Windows Network Places window, and the files on the Linux PC can be browsed using Windows Explorer. The Windows file system can be mounted on your Linux system, and you can browse the Windows files from your Linux PC.

All recent distribution of Linux which includes four samba packages i.e. samba, SambaClient ,Samba Common and Samba-Swat. To check whether Samba Sever is installed in your System or not. Type the following command on the terminal windows if the Samba is not installed in your System, then the following command will display the output that the Samba is not installed in your System.

samba rpm is required to configure samba server.

```
[root@Server ~]# rpm -qa samba*
samba-3.0.25b-0.el5.4
samba-common-3.0.25b-0.el5.4
samba-client-3.0.25b-0.el5.4
[root@Server ~]# _
```

Here you can download the latest version of the samba from the samba's website: www.samba.org. After downloading the samba from the site, you have to followed the instruction to how to install the samba server in your system. After downloading the Samba RPM file, install it as follows ("name of file" is the version number downloaded):

rpm -i samba(name of file)

Please ensure that yo have to install the Samba-common RPM, and if you want to use the Sambaclient, you can install the Samba-client RPM. If you are unable to download the RPM version, or if you want to compile the program by yourself, download the file

samba-latest.tar.gz.

Extract the file using the following command:

tar -xfvz samba-latest.tar.gz

Change to the directory containing the extracted files (usually /usr/src) and type ./configure.

Press Enter and wait for the command prompt to return. From the command prompt, type make. Press Enter and wait for the command prompt to return. Finally, type make install from the command prompt. If all goes well, Samba is installed when the command prompt returns. Now you need to configure it.

Configuring the Samba Server

You it must be configured the Samba before, to connect with your Windows Machine. There are many graphical-based tool are available for configuring the Samba, these programs are just front ends tools that will used to do some changes to the Samba configuration file behind the scenes. It is much quicker and easier to edit the Samba configuration file itself.

One of the tool is Samba-swat web based system is used to configure the samba. For using the SWAT Web based tools.

Type the following command on the terminal window:

Yum –y install samba samba-client samba-swat system-config-samba

Press the return keyword and wait for the command prompt to return, if no error has been return from the command prompt then the samba is been properly installed in your system. There are numerous graphical based tool are available for the configuration of the samba sever, but you can edit the samba sever configuration file very quickly and easily

There is one file is present known as smb.conf is the samba configuration file, which is located in the etc/samba directory. SWAT is the web based configuration program that provides you to configure your smb.conf file graphically.

A sample smb.conf file was created during the installation that can be used for reference and modification. The smb.conf file is divided into several sections, called shares, the names of section enclosed in the bracketed subsection titles in the following discussion.

This is the main Samba configuration file. You should read the
smb.confmanual page in order to understand the options listed
Here. Samba has a huge number of configurable options (perhaps too
many!) Most of which are not shown in this example
Any line which starts with a; (semi-colon) or a # (hash)
is a comment and is ignored. In this example we will use a
Forcommentary and a; for parts of the config file that you

```
# may wish to enable
# NOTE: Whenever you modify this file you should run the command
“testparm”
# To check that you have not made any basic syntactic errors.
#====================== Global Settings
==================================

[global]
log file = /var/log/samba/%m.log
smbpasswd file = /etc/samba/smbpasswd
load printers = yes
passwd chat = *New*password* %n\n *Retype*new*password* %n\n
*passwd:*all*authentication*tokens*updated*successfully*
socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
obey pam restrictions = yes
encrypt passwords = yes
passwd program = /usr/bin/passwd %u
dns proxy = no
netbios name = rhl
writeable = yes
server string = Samba Server
printing = lprng
path = /home
default = homes
unix password sync = Yes
workgroup = Tardis
printcap name = /etc/printcap
security = user
max log size = 50
pam password change = yes

[homes]
comment = Home Directories
browseable = yes
writeable = yes
create mode = 0664
directory mode = 0775
max connections = 1

[printers]
browseable = yes
printable = yes
path = /var/spool/samba
comment = All Printers
```

Before editing the smb.conf file, you need to run the following command to configure the samba

By using Chkconfig command, you can configure samba and Swat which will start at the boot time. The following is used to start the samba and swat respectively.

Chkconfigsmb on
Chkconfig swat on

You can start/stop/restart samba after boot time. The following command is used to start, stop and restart the samba.

Servicesmb start/stop/restart

After running the following command, you need to configure the smb.conf file.The file contains the different section, starting with the name of the section in the square bracket and continues up to the next section. Each section defines some parameters and attributes in the smb.conf file there are special section i.e. [global], [homes] and [printers]

Let's discuss the about the each section of smb.conf file

[global]

It is the initial section in the smb.conf file. It contains the specific and more general samba configuration parameter that can be applied to the whole server. There are long list of option under the global section. The [global] section contains a list of options and values in the following format:

option = value

Some of the parameters and attributes of the global section as follows

[global]

workgroup = DOCS
netbios name = DOCS_SRV
security = share
smbpasswd file = /etc/samba/smbpasswd
encrypt passwords = yes

For a complete listing of options, refer to the smb.conf man page. Some of the more significant options are:

- **workgroup = DOCS** — This is the name of the workgroup shown in the identification tab of the network properties box on the Windows computer.

- **netbios name = DOCS_SRV** — This is the name by which the Samba server is known to the Windows computer.
- smbpasswd file = /etc/samba/smbpasswd — This shows the path to the location of the Samba password file. Be sure that you include this option/value pair in your smb.conf file.

- **dns proxy = No** — This indicates that the NetBIOS name will not be treated like a DNS name and that there is no DNS lookup.— Beginning with Windows NT service pack 3 and later, passwords are encrypted. If you are connecting to any systems running these versions of Windows, you should choose encrypted passwords.

- **server string = Samba Server** — This is shown as a comment on the Windows PC in the network browser.

- **Security = user** — This is the level of security applied to server access. Other options are share, domain, and server. Share is used to make it easier to create anonymous shares that do not require authentication, and it is useful when the NetBIOS names of the Windows computers are different from other names on the Linux computer. Server is used to specify the server to use if the password file is on another server in the network. Domain is used if the clients are added to a Windows NT domain using smbpasswd, and login requests are executed by a Windows NT primary or backup domain controller

- **log file = /var/log/samba/log** This file is located in the /var/log directory thanks to the log file configuration option. However, we can use variable substitution to create log files specifically for individual users or clients, such as with the %m variable in the following line.

- **max log size = 1000**

The max log size option sets the maximum size, in kilobytes, of the debugging log file that Samba keeps. When the log file exceeds this size, the current log file is renamed to add an .old extension (erasing any previous file with that name) and a new debugging log file is started with the original name

- **socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192**

The main options are:

TCP_NODELAY

Have the server send as many packets as necessary to keep delay low. This is used on telnet connections to give good response time, and is used - somewhat counter-intuitively - to get good speed even when doing small requests or when acknowledgments are delayed (as seems to occur with Microsoft TCP/IP). This is worth a 30-50 percent speedup by itself. Incidentally, in Samba 2.0.4, socket options = TCP_NODELAY became the default value for that option.

IPTOS_LOWDELAY

This is another option that trades off throughput for lower delay, but which affects routers and other systems, not the server. All the IPTOS options are new; they're not supported by all operating systems and routers. If they are supported, set IPTOS_LOWDELAY whenever you set TCP_NODELAY.

SO_SNDBUF and SO_RCVBUF

The send and receive buffers can often be the reset to a value higher than that of the operating system. This yields a marginal increase of speed (until it reaches a point of diminishing returns).

SO_KEEPALIVE

This initiates a periodic (four-hour) check to see if the client has disappeared. Expired connections are addressed somewhat better with Samba's keepalive and dead time options. All three eventually arrange to close dead connections, returning unused memory and process-table entries to the operating system.

- **dns proxy = No** — This indicates that the NetBIOS name will not be treated like a DNS name and that there is no DNS lookup.

[homes]

In the section, the user login is defined. It provides fast and simple services to the larger number of client to access the home directories. It provides the various option like browseable and writable by which you can restrict the user to access the home directories.

[homes]
Comment =Home Directories
Browseable=no
Writable=yes
Create mode=0664

Directory mode=0775
Max connection=1

- comment = Home Directories — A comment line.

- browseable = yes — Means that the directory will appear in the Windows file browser.

- writeable = yes — Means that users can write to their directories.

- create mode = 0664 — Sets the default file permissions for files created in the directory.

- directory mode = 0775 — Sets the default permissions for created directories.

- max connections = 1 — The maximum number of simultaneous connections allowed. Setting this number to 1 prevents a user from logging in to the server from more than one location. Setting this number to 2 allows a user to log in from two locations and so on. Setting this number to 0 allows an unlimited number of connections.

[printers]
In this section you are providing the configuration of printer. This option provides the user to set print option to the desired directory. Some of the moiré useful option of the printer section is

[printers]
Comment=All printers
Path=/var/pool/samba
Browseable=yes
Printable=yes

- path = /var/spool/samba — The location of the printer spool directory.
- printable = yes — Enables clients to send print jobs to the specifieddirectory. This option must be set, or printing does not work.

- browseable = yes — Means that the printer appears in the browse list

The smb.conf file shown in the examples allows users who already have system accounts to access their home directories and to use printers. After modifying and saving the /etc/samba/smb.conf file, check the syntax of the file.

To do this, you can use the testparm command as follows:

[root@]# testparm
Load smb config files from /etc/samba/smb.conf
Processing section “[printers]”
Processing section “[homes]”
Loaded services file OK.
Press enter to see a dump of y

Now check **smb, portmap, xinetd** service in system service it should be on

#setup Select **System service** from list
[*]portmap [*]xinetd
[*]smb

Now restart **xinetd** and **portmap** and **smb** service

```
[root@Server ~]# service portmap restart
Stopping portmap:                                          [  OK  ]
Starting portmap:                                          [  OK  ]
[root@Server ~]# service xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@Server ~]# _
```

To keep on these services after reboot on then via **chkconfig** command

```
[root@Server ~]# chkconfig portmap on
[root@Server ~]# chkconfig xinetd on
[root@Server ~]# _
```

After reboot verify their status. It must be in running condition

```
[root@Server ~]# service portmap status
portmap (pid 3430) is running...
[root@Server ~]# service xinetd status
xinetd (pid 3462) is running...
[root@Server ~]# _
```

7. 2 CREATING SAMBA USERS

We have assigned a Linux user account to each individual using the Linux file system and printer from windows. Then we have to provide SMB Password to each user. To create a new Samba user, you need to perform the following command

Type the following command in the user terminal

Create a normal user namevinita

```
[root@Server backup]# useradd vinita
[root@Server backup]# passwd vinita
Changing password for user vinita.
New UNIX password:
BAD PASSWORD: it is WAY too short
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@Server backup]#
```

Now create **/data** directory and grant it **full permission**

```
[root@Server ~]# mkdir /data
[root@Server ~]# chmod 777 /data
[root@Server ~]# _
```

Open **/etc/samba/smb.conf** main samba configuration files

```
[root@Server ~]# vi /etc/samba/smb.conf _
```

By default name of workgroup is **MYGROUP** in **smb.conf** file. You can change it with desire name

```
Hosts Allow/Hosts Deny lets you restrict who can
specifiy it as a per share option as well

        workgroup = MYGROUP
        server string = Samba Server Version %v
```

Our task is to share **data** folder for **vinita** user so go in the end of file and do editing as shown here in this image

```
# Add this line to share
    [data]
 comment = personal share
 path = /data
 public = no
 writable = yes
 printable = no
 browseable = yes
 write list = vinita
```

Save file with **:wq** and exit
Now add vinita user to **samba user**

```
[root@Server ~]# smbpasswd -a vinita
New SMB password:
Retype new SMB password:
[root@Server ~]# _
```

We have made necessary change now on **smb service** and check it status

```
[root@Server ~]# chkconfig smb on
[root@Server ~]# service smb start
Starting SMB services:
Starting NMB services:
[root@Server ~]# service smb status
smbd (pid 4332 4327) is running...
nmbd (pid 4330) is running...
[root@Server ~]# _
```

Starting the Samba Server

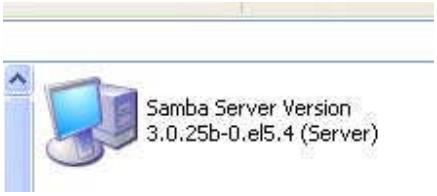
The last step is to start the Samba daemon. The command to start Samba is:
[root@ Server]# /sbin/service smb start
Starting SMB services: [OK]
Starting NMB services: [OK]

If you already have on this service then restart it with **service smb restart** commands.

7.3 CONNECTING WINDOWS PC TO THE SAMBA SERVER

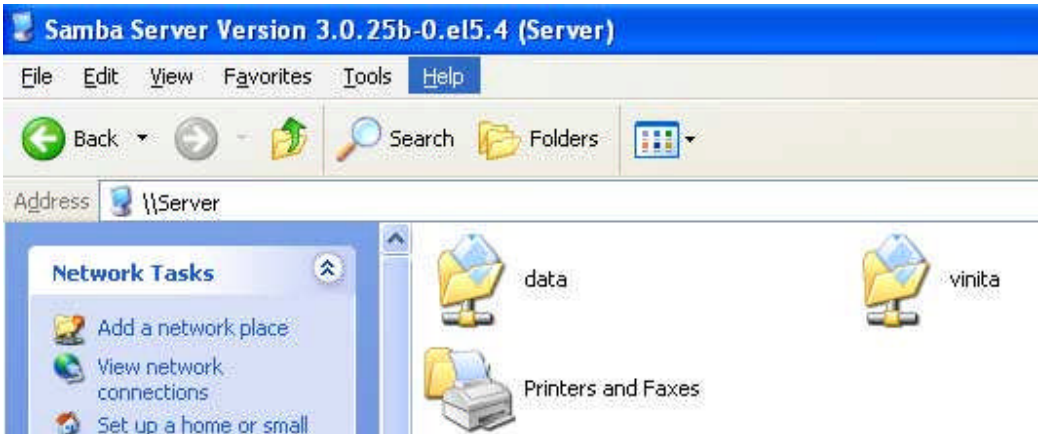
Go on windows system and **ping** samba server, change computer name to **client2** and workgroup name to **MYGROUP**

reboot system after changing workgroup nameAfter reboot open my network place here you can see **samba server** [if not see then click on view workgroup computer in right pane, if still not see then use search button from tool bar and search computer samba server form ip].

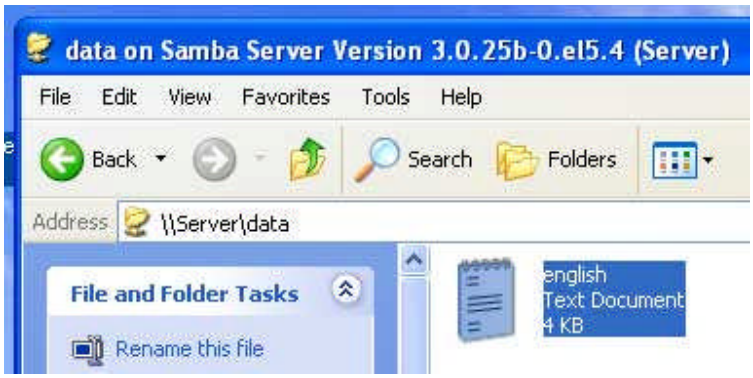


First try to login from user **nikita** she will not successes as nikita have not permission to login

As you can see in image user vinita gets the **/data** folder which we share from **samba server**



Copy some window files in **data** folder



7.4 CHECK STATUS ON SAMBA SERVER

On **samba server** you can check runtime status of samba server to check it run **smbstatus** command

```
[root@Server ~]# smbstatus

Samba version 3.0.25b-0.el5.4
PID     Username      Group         Machine
-------------------------------------------------------------------
 4720   vinita        vinita        client2      (192.168.0.49)

Service      pid     machine       Connected at
-------------------------------------------------------
data         4720   client2       Thu Feb 18 19:41:05 2010
IPC$         4720   client2       Thu Feb 18 19:40:56 2010

Locked files:
Pid          Uid        DenyMode   Access      R/W        Oplock           Share
Path   Name   Time
-------------------------------------------------------------------------------
------------------
4720         503        DENY_NONE  0x100001    RDONLY     NONE             /data
   .   Thu Feb 18 19:41:13 2010

[root@Server ~]# _
```

Let begin with an example

Requirement:-

- Package = samba
- Service = smb
- Port No. = 445
- Configuration File = /etc/samba/smb.conf

Per quest:-

- Configure IP = 192.168.1.1
- Hostname = server.rootuser.in
- Firewall should be off on server.

1] Install required packages
[root@server ~]# yum install samba* -y

2] Create one new user
[root@server ~]# useradd demo

[root@server~]# passwd demo
3] Create new directory that will host Samba share and grant it full permission.
[root@server ~]# mkdir /fulldata
[root@server ~]# chmod 777 fulldata

4] Create some files under fulldata directory
[root@server ~]# cd /fulldata
[root@serverfulldata]# touch a1 n2 u3 p4
[root@server fulldata]# cd

5] Now add demo user to Samba user
[root@server ~]# smbpasswd -a demo
New SMB password:
Retype new SMB password:

6] Use following command to confirm user was added to the samba database
[root@server ~]# pdbedit -w -L

7] Edit main configuration file
[root@server ~]# vim /etc/samba/smb.conf
#By default name of workgroup is MYGROUP you can change to desire name.

workgroup = rootuser
#Remove hash from follwing line and edit as follows
hosts allow 192.168.1.0/24
#Go to end of the file and type as follows:
[fulldata]
comment = mysamba
path = /fulldata #samba share directory name
public = no #Cannot be used by public users
browseable = yes
valid users = demo #Only demo anup can use samba share
:wq

8] Set Selinux security related options
[root@server ~]# chcon -t samba_shared_t /fulldata

9] Use following command to check smb.conf file parameter
[root@server ~]# testparm

10] Start the smb service and make it permanent
[root@server~]# service smb start
[root@server ~]# chkconfig smb on

7.5 CLIENT SIDE CONFIGURATION

1] Check communication with Samba server
[demo@server ~]$ ping 192.168.1.1

2] Create one directory to store downloaded files.
[demo@server~]$ mkdir /client

3] Now try to connect to Samba server
[demo@server ~]$ smbclient //192.168.1.1/fulldata -U demo
Password:
smb:\>ls
smb:\> get a1
smb:\> exit
[demo@server ~]$
OR
4] You can also mount samba share directory
[demo@server ~]# smbmount 192.168.1.1:/fulldata /client -o username=demo

7.6 WINDOWS CLIENT SIDE CONFIGURATION

1] Check communication with Samba server
cmd\> ping 192.168.1.1

2] Change computer name to **windowsclient** and change workgroup to **rootuser**
Right click on **My Computer** --> Select **Properties** --> Click the **Computer Name** Tab --> Select **Change button** --> Type the computer name in given filed and type the workgroup name in workgroup filed.

3] After reboot open Go to Start --> Select Search option --> then search Samba server with itsip

8

ADDITIONAL NETWORK SERVICES

Unit Structure

8.1 ADDITIONAL NETWORK SERVICES

Any System Administrator can handle the request for new serviced that are alert constantly. One of the common services is a timeserver. This service provides the authoritative clock against which all the clock in the system must be sync. In this section we learn the two non-essentials LAN based Services: an NTP based time server and a caching proxy Server

8.2. CONFIGURING A TIMESERVER

A timeserver is a daemon that is run on one physical machine and the other system will synchronize with their system clocks. Generally speaking, you can synchronize your timeserver's clock with one or more timeservers that are present outside your LAN.In some situation, the timeserver has to synchronize its time with a specially designed clock. The hardware clock is separate device specially meant for single use that will maintain the accurate clock time. The main aim of the separate clock is to keep the system time consistent through the LAN to facilitate the time sensitive operations. The irregularities in the system clocks between client system and the server can have adverse effects.

The Most important and widely used protocol for distributing and synchronizing time is the Network Time Protocol.NTP is a protocol is used for synchronizing the linux's system clock which

have the accurate time source. The ntpd, NTP Daemon is act as dual purpose. It can act as server for listening time synchronization request and providing the time in response and it can acts as client, it can adjust the local system time by communicating with the other timeservers

Following are the NTP utility programs

1) Ntpq: standard NTP Query program
2) Ntpdc:special NTP query program
3) Ntpdate: set the date and time via NTP
4) Sntp: Simple Network Time Protocol (SNTP) client
5) Ntptrace: trace a chain of NTP servers back to the primary source
6) Tickadj:set time related kernel variable
7) Ntptime:read and set kernel variable
8) Ntp-keygen: generate public and private keys
9) Ntpdism: network time protocol (NTP) Simulator

8.2.1 Install ntp

The ntp package contains utilities and daemons that will synchronize your computer's time to Coordinated Universal Time (UTC) via the NTP protocol and NTP servers. The ntppackageincludesntpdate (a program for retrieving the date and time from remote machines via a network) and ntpd (a daemon which continuously adjusts system time).
Installing the NTP software is simple. Use the rpmquery command to make sure that the ntp package is installed:

$ **rpmqueryntp**

ntp-4.2.0.a.20040617-4

The version number you see might be slightly different. If the ntp package isn’t installed, install it using the installation tool of your choice before proceeding.
Install the ntp package:
yum install ntp
This command installs ntp in the system.

8.2.2 Reference Clock

Timeserver gives you the accurate time after the time has been synchronizing with one or more clock and master clocks.NTP server are in distributed nature that is servers and clients are spread in worldwide, any given client can ask the time server for time check.

NTP uses a hierarchical system of levels of clock sources known as a stratum,to reduce load in any given server or set of servers.Stratum1 servers are referred to as primary servers,stratum 2 servers are called as secondary servers and so on. The secondary servers will synchronize with the primary servers and clients will synchronize to the secondary or tertiary servers.NTP also provide a large set of publically accessible secondary servers, pool servers to used in a large scale. The NTP pool time servers are organized in to the sub sets pool.ntp.org. The main aim of its to distributes the client load more or equally across the all servers that is participated in the pool and it will ensure that the client will synchronize with an distributed set of timeservers.

In the following section, you need to configure the NTO Servers that will use the pool servers

ntpd's configuration file, /etc/ntp.conf, stripped of most comments and white space.
restrict default nomodifynotrapnoquery

```
restrict 127.0.0.1
# --- OUR TIMESERVERS -----
server pool.ntp.org
server pool.ntp.org
server pool.ntp.org
# --- GENERAL CONFIGURATION ---
server 127.127.1.0  # local clock
fudge 127.127.1.0 stratum 10
driftfile /var/lib/ntp/drift
broadcastdelay 0.008
keys/etc/ntp/keys
```

Listing 1: The default NTP configuration file. The first two entries, beginning with the restrict directive, are, not surprisingly, restrictions on the listed IP addresses or hostnames. The first entry uses the keyword default, which means an IP address and mask of 0.0.0.0. The option flags, nomodify, notrap, and noquery, prevent the listed IP address from modifying, logging, or querying the NTP service on the server. The second rule, restrict 127 .0.0.1, permits all NTP activity over the loopback interface.

The next three entries, beginning with the server directive, identify the time servers you want to use as reference clocks.

Even though the names are the same, the NTP server pool is configured to use DNS round robin, so three hostname lookups on the same name will return three different IP addresses.

```
$ host pool.ntp.org
        pool.ntp.org has address 213.219.244.16
        pool.ntp.org has address 216.27.185.42
        pool.ntp.org has address 62.220.226.2
        pool.ntp.org has address 69.37.143.241
        pool.ntp.org has address 81.169.154.44
        pool.ntp.org has address 82.219.3.1
        pool.ntp.org has address 139.140.181.132
        pool.ntp.org has address 146.186.218.60
        pool.ntp.org has address 195.18.140.242
        pool.ntp.org has address 203.217.30.156
        pool.ntp.org has address 209.126.142.251
        pool.ntp.org has address 212.23.29.225
```

Normally, a hostname resolves to one and only one IP address, but when DNS round robin behaviour is enabled, a single hostname can resolve to multiple IP addresses, the purpose being to equalize the load on any single system. The line server 127.127.1.0 instructs the NTP daemon to use the local clock (referred to as an undisciplined local clock) if no external reference clocks are accessible. You can use any address in the range 127.127.1.0 to 127.127.1.255, although the convention is to use 127.127.1.0. The line fudge 127.127.1.0 stratum 10 limits the use of the local lock by assigning it a very low place in the time server hierarchy.

The directive driftfile/var/lib/ntp/drift specifies the name of the file that stores the oscillation frequency of the local clock. The broadcastdelay directive sets the number of seconds (0.008 in this case) used to calculate the network latency or delay between the local server and a remote reference server.

The last line, keys/etc/ntp/keys, tells NTP where to find the cryptographic keys used to encrypt exchanges between client and server machines.

NTP version 4 (NTPv4) supports asymmetric encryption, more commonly known as public key encryption, using a method or protocol referred to as autokey

To create the basic autokey setup and to configure an NTP Server follow the below step

1. Create a directory for the NTP Keys (for example, /etc/ntp)
2. . Add the following lines to ntp.conf:

   crypto pw server password

   keysdir /etc/ntp
3. Append autokey to the broadcast line in ntp.conf for the broadcast/multicast address that you want to authenticate with Autokey. Broadcast my.broadcast.or.multicast.addressautokey. The assigned NTP Multicast address is 224.0.1.1, but other valid multicast addresses may be used.
4. The server key and certificate will be generated by ntp-keygen if they are missing when a set of parameters are generated. The server certificate will be updated when existing parameters are updated or additional parameters are generated

Generate the key files and certificates with the following commands:

cd/etc/ntp
ntp-keygen -T -I -p serverpassword

If ntpd is running, restart it:
service ntpd restart
Shutting down ntpd: [OK]
Starting ntpd: [OK]

If ntpd is not running, start it:
service ntpd start
Starting ntpd: [OK]

4. Use the following chkconfig commands to make sure that ntpd starts in at boot time and in all multiuser run levels:
chkconfig --level 0123465 ntpd off
chkconfig --level 345 ntpd on

8.2.3 Configuring an NTP Client

The GUIaddicted →can use the Date/Time Properties tool. Either start it from the menu (Red Hat Date→System Settings & Time) or type system-config-date at a command prompt. Either way, you should see the screen shown in Figure 1. If NTP is already running on your system, the Date & Time tab will be disabled (grayed out). Click the Network Time Protocol tab to configure NTP.

Fig. 1 : The Date/Time Properties tool.

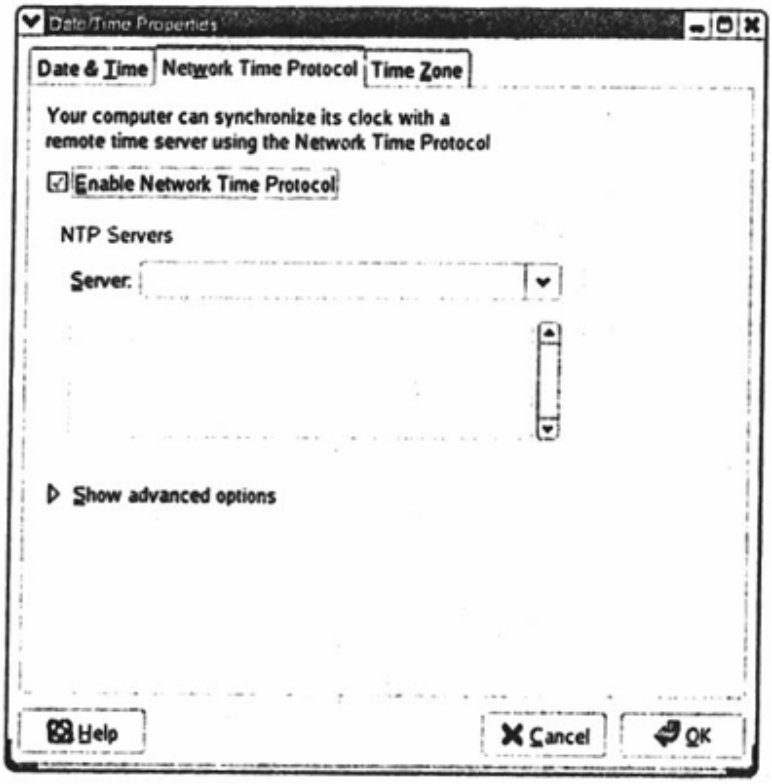


Fig. 2 : The Network Time Protocol tab.

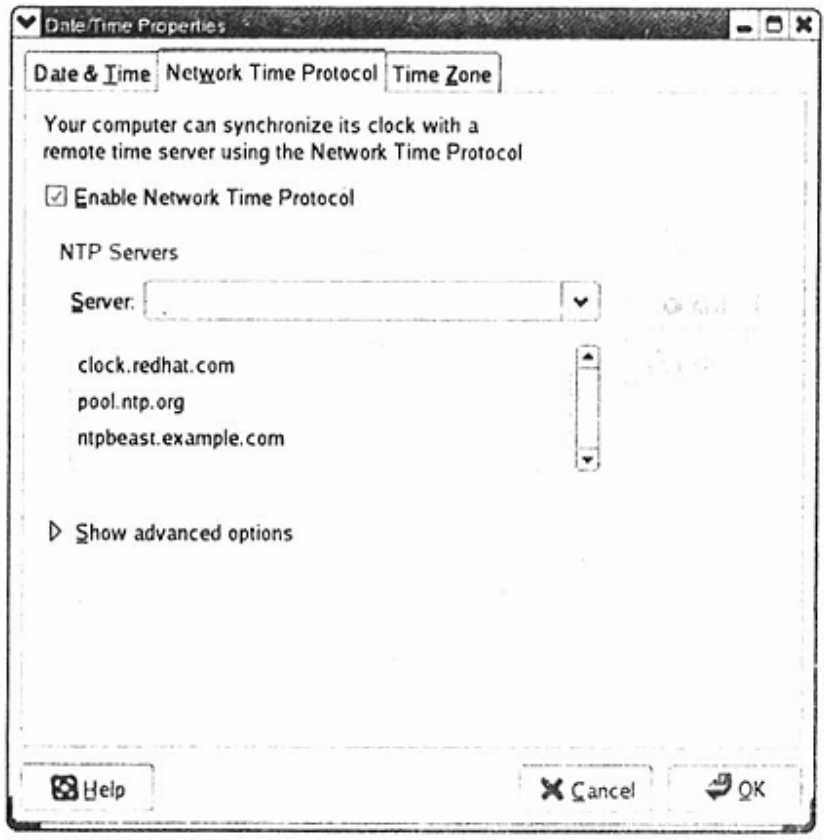


Fig. 3 : Adding NTP servers.

To configure advanced NTP options, click the Show advanced options arrow. These options allow you to use the system clock (the undisciplined local clock described in the previous section) as a reference clock (enabled by default) and to use NTP broadcast (disabled by default). NTP broadcast causes the NTP daemon to listen for remote servers rather than configuring clients to use a specific server.
After you have made your configuration changes, click the OK button to close the Date/Time Properties tool.

1. If you configured your NTP server to use autokey encryption, you will also need to configure any NTP clients you have to use autokey encryption. The following procedure walks you through doing so.
   Add the following lines to /etc/ntp.conf on the client:
   crypto pw clientpassword
   keysdir /etc/ntp
   server *timeserver* autokey
   Replace clientpassword with the password you want to use on the client. Replace timeserver with the name or IP address of the system

2. Generate the client keys and certificate:
cd /etc/ntp
ntp-keygen -H -p *clientpassword*

3. Import the key created in the section on server configuration using some sort of encrypted mechanism, such as sftp or scp. The following example uses scp:

cd /etc/ntp
scp*user*@*timeserver*:/etc/ntp/
ntpkey_IFFkey_*timeserver*.3318548048 .
ln -s ntpkey_IFFkey_*timeserver*.3318548048
ntpkey_iffkey_*timeserver*

Replace user with the name of a user that has scp privileges on the machine you are using as the time server. Under normal circumstances, this can be any user with a login account on the time server.

4. If ntpd is running, restart it:

 # servicentpd restart
Shutting down ntpd: [OK]
Starting ntpd: [OK]
If ntpd is *not* running, start it:
service ntpd start
Starting ntpd: [OK]

5. Execute the command ntpq -p to make sure that the NTP client can communicate with the designed server. The output should resemble the following:

 # ntpq -p ntpbeast.example.com

Replace ntpbeast.example.com in the command with the name (or IP address) of the time server you are using .As a final test, you can use the ntpstat command to query the time server from the host machine to ensure that you can retrieve the time. The output will show the server to which ntpd synchronizes, the clock's precision, and the interval on which ntpd resynchronizes:

ntpstat
synchronised to NTP server (192.168.0.1) at stratum 3
time correct to within 70 ms
polling server every 128 s

After ntpd has been running for a while, you can grep the system log for NTP-related log entries to see ntpd's activity. The following listing briefly illustrates what you might see:
grepntpd /var/log/messages

8.3 PROVIDING A CACHING PROXY SERVER

Caching a proxy server is a software that facilitates the storage of frequently requested Internet objects. Objects (Such as Web pages, java Scripts and other downloaded files) requested from the client are stored by proxy server.Main aim of caching proxy server is

1) To provide faster web browsing by reducing access time for frequently requested objects.
2) To minimize the bandwith consumption, by caching popular data locality(on server that eixts between the requesting client and the internet)

Squid provides the basic caching and proxy fuction such as DNS lookups, speed up, subsequent DNS queries and implements negative caching. Negative caching means that squid remembers when a request was made for an object that does not exists and does not try retrieve it or find the non-existent object in the cache. It has full SSL support. Also includes extensive access control and support rich logging system failure

An ordinary proxy requires Web clients to specify the hostname and port of the proxy to forward the requests to the requested server. But in squid, web clients think they are communicating with the requested server when in real they are communicating with the proxy. Therefore, you can say squid work as a transparent proxy.

The following query to check whether squid is installed in the system or not

$ rpmquery squid

The following command returns an output showing the version of quid installed in the system.

Squid-3.0.STABLE20.fc11.i586

Perform the foolowing steps to install the squid:

1) Verifying the kernel Configuration
2) Configuring Squid
3) Modifying the Netfilter Configuration
4) Starting Squid
5) Testing the Configuration

8.3.1 Verifying the kernel Configuration

You need to verify the kernel configuration to use its features as IP forwarding and NetFiliter(iptables) support. Netfilter support handles the actual proxying of browser requests.you need to enable Netfilter and the modules that support:

1) Connection tracking
2) IP tables
3) Full network Address Translation (NAT)
4) Support for The REDIRECT target

IP forwarding enables the kernel to send or forward packets that arrive on one network interface to other network interface . To enable IP forwarding on the system that run squid, run the command:

#sysctl –n net.ipv4.ip_forward

The above command returns the output either 0 (depicting disable) or 1 (depicting enable). To enable it use the following command:

#sysctl –w net.ipv4.ip_forward=1

To start squid automatically at the boot time , edit the /etc/sysctl.conf and change the line that reads:

net.ipv4.ip_forward=0

and Make changes as

net.ipv4.ip_forward=1

This enables IP Forwarding at boot times. After enabling IP Forwarding, you need to configure quid

8.3.1 Configuring Squid

The /etc/squid/squid.conf is the configuration file on fedora core.squid is controlled by the initialization scripts, /etc/rc.d/init.d/squid. It reads default values from /etc /sysconfig/squid.

Following are the some configuration settings required to configure squid, as shown Table 1

PARAMETER	DEFAULT VALUE	DESCRIPTION
cache_effective_group	squid	Identifies the group Squid runs as
cache_effective_user	squid	Identifies the group Squid runs as
httpd_accel_host	None	Defines the hostname of the real HTTP server (if using acceleration)
httpd_accel_with_proxy	off	Controls whether Squid runs as both an accelerator and a proxy
httpd_accel_port	80	Defines the port number of the real HTTP server (if using acceleration)
httpd_accel_uses_	off	Enables Squid to function as a host_header transparent proxy
httpd_access	deny all	Defines who can access the Squid server

cache_effective_user and cache_effective_ group identify the user ID (UID) and group ID (GID), respectively, under which Squid runs. httpd_accel_with_proxy, which defaults to off, controls whether

Squid runs as a cache (or accelerator) and proxy or just as proxy. Off value depicts that Squid functions only as a proxy. And on value depicts that, Squid works as both a cache *and* a proxy.

If you are using Squid's caching functionality, httpd_accel_port to 80 and use httpd_accel_host to define the name the host running Squid. If you want a transparent proxy server, set httpd_accel_uses_host_header to on. The default value, off, means that clients have to configure their Web clients to use a proxy server.

The final value to configure is httpd_access, which controls who can access the Squid server and, therefore, who can surf the Web through the proxy. The default configuration is deny all, which prevents *any* user from accessing the proxy. Modifying the Net Filter Configurationmodifying your netfilter firewall rules is an optional step. Youdo not require to configure it, if you are not using to maintain a firewall or to provide your LAN access to the Internet

Execute the following Command

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \ -j REDIRECT --to-port 3128

This command updates the NAT or *network address translation* table (-t nat), appending a rule to the prerouting chain (-A PREROUTING). The TCP protocol packets (-p tcp) arriving on the network interface eth0 (-i eth0) that are meant for port 80 (--dport 80) are redirected (-j REDIRECT) to port
3128 (--to-port 3128), which is the port on which Squid listens

Use the following command to verify that the rule is in effect:

sudo /sbin/iptables -L -t nat

The output shows the PREROUTING chain has been modified to redirect HTTP packets to port 3128. Now you're ready to start Squid

Starting Squid

Type the following command to start squid

service squid start

Starting squid: [OK]

You can use the chkconfig command to set Squid to start and stop automatically:

chkconfig --level 0123456 squid off
chkconfig --level 345 squid on

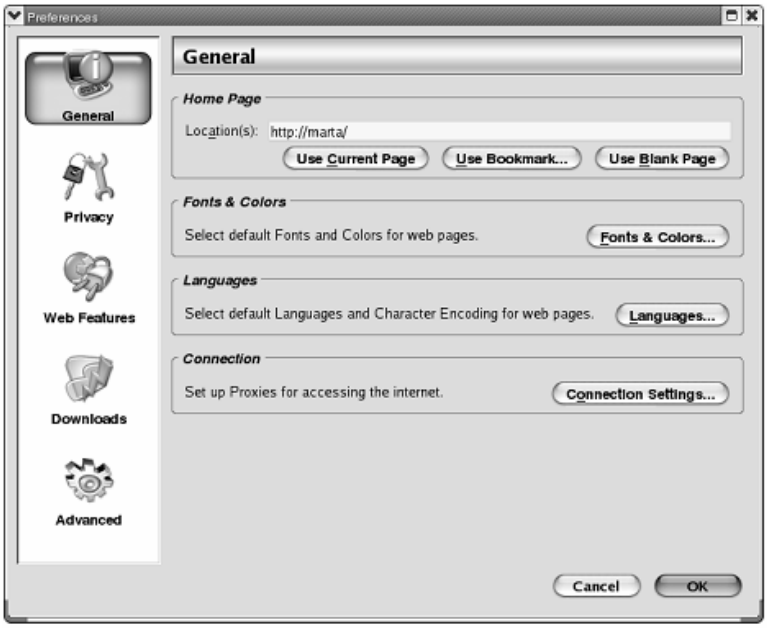
The first command disables Squid in all run levels. The second command causes Squid to start when the system enters run levels 3, 4, or 5.

8.3.2 Testing the Configuration

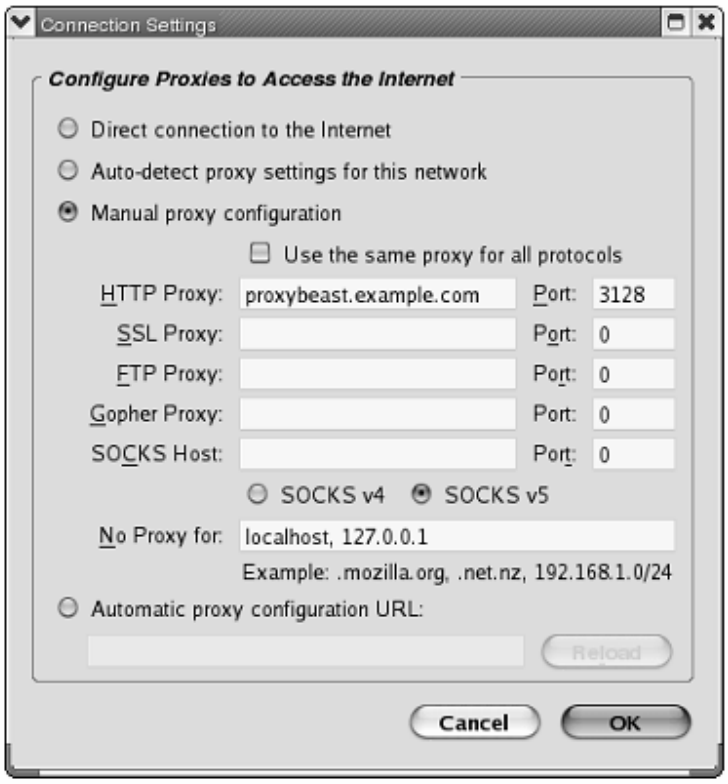
To test the configuration, you need to configure your browser to use proxy .In Mozilla Firefox, select Edit ➪ Preferences to open Mozilla's Preferences dialog box. On the General tab, click Connection Settings to open the Connection Settings dialog box. Click the Manual proxy configuration radio button and type the hostname or IP address of the proxy server in the HTTP Proxy text box.

Type **3128** in the accompanying Port text box. The completed settings might resemble shown in figure

Click OK to close the Connection Settings dialog box and OK again to save your changes and close the Preferences dialog box.



Firefox's Preferences dialog box.



Firefox's Connection Settings dialog box.

9

INTERNET SERVICES

Unit Structure :

9.1 INTERNET SERVICES

Any services that accessible through TCP/IP based Network are categorized as Internet Services. Internet services are provided through either secure or non-secure connection. Common Services of TCP/IP connection based are Telnet ,FTP, SMTP, HTTP, ICMP, ARP, DNS, SSH, SCP, SFTP and may more. TCP/IP provides platform-independent protocol to these services.

By Using TCP/IP protocol, any operating system can establish connection and transmit data with other computer on the network. Linux machines facilitate us with a wide range of internet services.

9.1.1 Secure Services

In the beginning stages of Internet services when everyone trusted everybody else on the internet. Data is used to send in the form of plain text format including the sensitive information . But with the growth of Internet. Exchanging information through these service is not a good idea. As its lead to be high risk of fraud. There are common services such as Telnet and FTP are less secures services are replaced with the some secures services that provide a stronger authentication control.

9.1.2SSH

SSH Stands for secure shell. It is secure protocol for accessing the remote computer system. It was created with

replacement of the Telnet services. IT uses a public / private encryption key exchange protocol to encrypt all the traffic including password.

This is what it looks like to SSH into a machine for the first time:

[vnavrat@buffy vnavrat$ ssh vnavrat@woolf.xena.edu

The authenticity of host 'woolf.xena.edu (123.456.789.65)'

can't be established.

RSA key fingerprint is

b2:60:c8:31:b7:6b:e3:58:3d:53:b9:af:bc:75:31:63.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added
'woolf.xena.edu,123.456.789.65'

(RSA) to the list of known hosts.

vnavrat@woolf.xena.edu's password:

Welcome to woolf

Unauthorized usage prohibited. Please check your quotas.

vnavrat:~>

You can see the ssh manpage by the following command in the terminal window

manssh

You can connect to a server that support ssh with the following command

ssh-p yourportyourname@your server

Replace yourport with the default port 22, yourname with the host name or an ip address and yourserver with the hostname or an ip address of the server you want to connect.

If the server support SSH connection, you can reach the default port 22. At the first time SSH ask to confirm whether you can to continue and ask to provide password. Aftersuccessfullyconnected to the server you can send and receive data.

In addition to providing terminal access, SSH tunnels almost any other protocol through it. So, it is possible to tunnel POP, RCP, and other protocols through SSH to turn them into encrypted, more

secure protocols. With enough imagination and practice, you can make almost anything more secure with SSH.

Following is an example of how to tunnel your mail through SSH to keep your password and mail encrypted and secure during transit. In this example, you use POP3 to retrieve your mail from the remote machine buffy.xena.edu. Normally you would tell your POP3 software to connect from your localhost to port 110 (the POP port) of buffy.xena.edu.

But in this example the first step is to configure your POP mailer to connect to port 16510 of your own machine, and put in the password for your account on buffy.xena.edu. The second step is to set up the SSH tunnel, which encrypts and forwards the traffic over the network to terry.muhlenberg.edu's POP port.

To set up the SSH tunnel, type the following at the command line:

ssh -N -L 16510:127.0.0.1:110 terry@terry.muhlenberg.edu

You are now sending and receiving your mail through an encrypted SSH tunnel.

9.1.3 Scp

Secure copy (scp) in the part of the SSH Package. The scp command allows you to copy files over ssh connection. It is replacement to RCP and FTP. The SCP Command can used in the three way:

1) To copy from a remote server to your computer, use the following syntax

   Scp file yourusername@youserver :/home/yourusername/

2) To copy from a remote server to a remote computer, use the following syntax

   Scp yourusername@yourserver :/home/yourusername/file

3) To copy from from a remote server to another remote server, use the foolowing syntax

   Scp yourname@yourserver: /home/yourusername /file yourusername2@yourserver2:/home/yourusername2/

In the above command, ‘file’ is the file to be transferred to the directory ‘home/yourname/’ at the server’your’ with the username ‘yourusername’ and ‘.’ The dot at the end means the current local directory

If you have KDE installed on your system, you can use the Konqueror browser and the fish protocol to establish an SSH connection to a remote PC Then you can drag and drop the files or directories you want to copy between the remote and local machine. Follow the steps here:

1. Open the Konqueror browser, enter fish://<name or IP address of remote PC> into the browser location field and press Enter. If this is the first time you are connecting to the remote host, you will be prompted about the connection, as shown in Figure 9.1. Click Enter toaccept the information.

2. When prompted, enter the username and password and click OK to connect. Figure 9.2 shows the contents of my home directory on my home PC.

3. To copy or move files or directories from the remote PC to the local PC, select the files or directories you want to copy or move, right-click the selected items and either copy to or move the selected items from the pop-up menu, and browse to where you want to place them

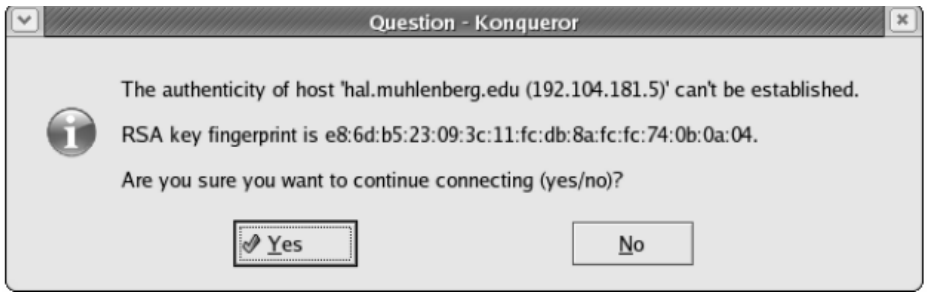


Fig. 9.1

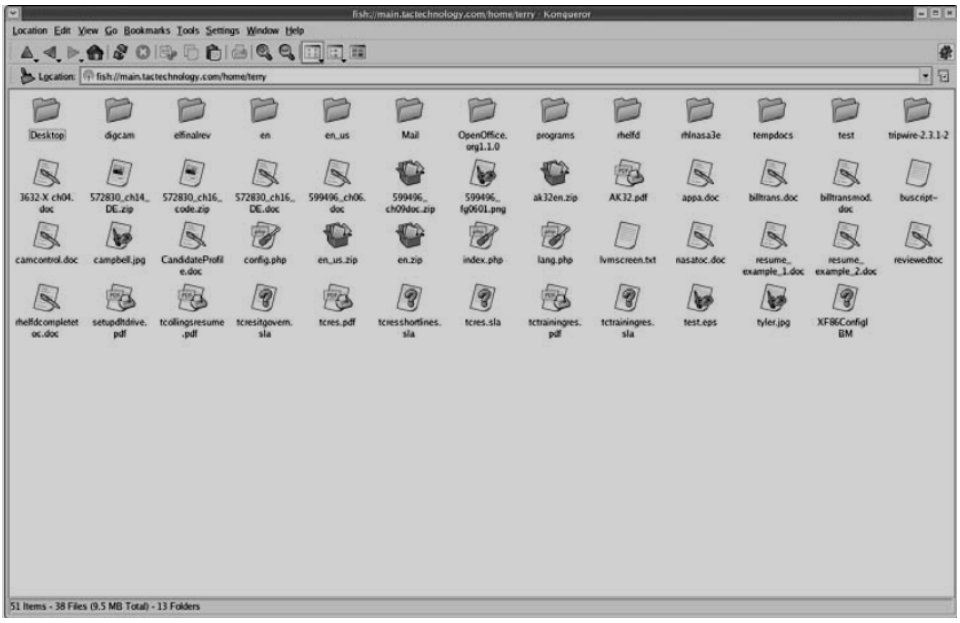


Fig. 9.2

9.1.4 Sftp

Secure File Transfer Program (sftp) is replacement to traditional FTP. It performs all the function over SSH. The Syntax for sftp is

sftpuser@host:file file

Here, user is the local machine, host is the remote machine and file is the filename

For example, shows copying the file tcres.pdf from the remote PC main to the localPC.

[terry@terry ~]$ sftp terry@main:tcres.pdf tcres.pdf

Connecting to main...

terry@main's password:

Fetching /home/terry/tcres.pdf to tcres.pdf

/home/terry/tcres.pdf 100% 222KB 73.9KB/s 00:03

The options are as follows:

Tag	Description
-**1**	Specify the use of protocol version 1.
-**B** *buffer_size*	Specify the size of the buffer that sftp uses when transferring files. Larger buffers require fewer round trips at the cost of higher memory consumption. The default is 32768 bytes.
-**b** *batchfile*	Batch mode reads a series of commands from an input batchfile instead of stdin. Since it lacks user interaction it should be used in conjunction with non-interactive authentication. A batchfile of '-' may be used to indicate standard input. sftp will abort if any of the following commands fail: get, put, rename, ln, rm, mkdir, chdir, ls, lchdir, chmod, chown, chgrp, lpwd and lmkdir. Termination on error can be suppressed on a command by command basis by prefixing the command with a '-' character (for example, -rm /tmp/blah*).
-**C**	Enables compression (via ssh's -**C** flag).
-**F** *ssh_config*	Specifies an alternative per-user configuration file for **ssh**. This option is directly passed to **ssh**.
-**o** *ssh_option*	Can be used to pass options to **ssh** in the format used in **ssh_config**. This is useful for specifying options for which there is no separate **sftp** command-line flag. For example, to specify an alternate port use: **sftp -oPort=24**. For full details of the options listed below, and their possible values, see **ssh_config**.

9.1.5 Less Secure Service

There are some non-secure service that should not use as all their traffic is sent over the network in plain text. In the following section there are some less secure services as follows

1) **Telnet:** It is the protocol that enables you to have access to a virtual terminal on a remote host.It is text-based console application. Telnet is an application that's available almost everywhere. Because of this distribution, most beginning UNIX users use Telnet exclusively to communicate with other UNIX and NT machines. Since all Telnet traffic, including passwords, is sent in plain text, the Secure Shell (ssh) command should be

used instead, if at all possible. SSH provides an equivalent interface to Telnet, with increased features, and most importantly, encrypted traffic and passwords.

This is what it looks like when you log into a machine with Telnet:
[terry@terry ~]$ telnet terry
Trying 127.0.0.1...
Connected to xena.
Escape character is '^]'.
Welcome to terry.muhlenberg.edu
login:

2) FTP: It is the standard application that run over port 20 and 21. You can type help command to list the available commands in FTP. Two important command are put and get which provides moving a file from your machine to the remote machine and to pull a file from the remote server to your machine respectively. Use mput and mget ,in case of multiple files.ls ordir gives you a listing of files available for downloading from the remote site.

The options are as follows:

TAG	DESCRIPTION
-A'	Use active mode for data transfers. This is useful fortransmissions to servers which do not support passive connections (for whatever reason.).
-p'	Use passive mode for data transfers. Allows use of ftp in environments where a firewall prevents connections from the outside world back to the client machine. Requires that the ftp server support the PASV command. This is the default now for all clients (ftp and pftp) due to security concerns using the PORT transfer mode. The flag is kept for compatibility only and has no effect anymore.
-i'	Turns off interactive prompting during multiple file transfers.
n'	Restrains ftp from attempting ''auto-login'' upon initial connection. If auto-login is enabled, ftp will check the. netrc file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, ftp will prompt for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login.
-e'	Disables command editing and history support, if it was compiled into the ftp executable. Otherwise, does nothing

-g'	Disables file name globing
-m'	The default requires that ftp explicitly binds to the same interface for the data channel as the control channel in passive mode. Useful on multi-homed clients. This option disables this behaviour. Files cannot be extracted from a thin ftpchive.
-v'	Verbose option forces ftp to show all responses from the remote server, as well as report on data transfer statistics
-d'	Enables debugging.

EXAMPLES

Example-1:

To see help of all available commands in ftp
$ ftp
ftp> help
output:
$ ftp
ftp> help
Commands may be abbreviated. Commands are:

!	dir	mdelete	qc	site
$	disconnect	mdir	sendport	size
account	exit	mget	put	status
append	form	mkdir	pwd	struct
ascii	get	mls	quit	system
bell	glob	mode	quote	sunique
binary	hash	modtime	recv	tenex
bye	help	mput	reget	tick
case	idle	newer	rstatus	trace
cd	image	nmap	rhelp	type
cdup	ipany	nlist	rename	user
chmod	ipv4	ntrans	reset	umask
close	ipv6	open	restart	verbose
cr	lcd	prompt	rmdir	?
delete	ls	passive	runique	
debug	macdef	proxy	send	

Example-2:

To Changing FTP Mode to binary or ascii
ftp> ascii
output:
200 Type set to A.

Example-3:

To download a file using ftp:

ftp> get README

output:
200 PORT command successful.
150 ASCII data connection for README
(128.138.242.10,3134) (2881 bytes).
226 ASCII Transfer complete.
local: README remote: README
2939 bytes received in 0.066 seconds (43 Kbytes/s)

3) Rysnc: This file transfer program is similar to RCP.It sends traffic unencrypted. The server listens on port 873

Some advantages and features of Rsync command

It efficiently copies and sync files to or from a remote system. Supports copying links, devices, owners, groups and permissions.

It's faster than scp (Secure Copy) because rsync uses remote-update protocol which allows to transfer just the differences between two sets of files.

First time, it copies the whole content of a file or a directory from source to destination but from next time, it copies only the changed blocks and bytes to the destination.

Rsync consumes less bandwidth as it uses compression and decompression method while sending and receiving data both ends.

Basic syntax of rsync command

rsync options source destination

Some common options used with rsync commands

-v : verbose

-r : copies data recursively (but don't preserve timestamps and permission while transferring data

-a : archive mode, archive mode allows copying files recursively and it also preserves symbolic links, file permissions, user & group ownerships and timestamps

-z : compress file data

-h : human-readable, output numbers in a human-readable format

4) Rsh: it is unencrypted process for executing command on remote hosts.You specify a command to be run on the remote host on rsh's command line, but if no command is given, you are logged into the remote host using rlogin.

rsh's syntax is
rsh remotehostname remotecommand

The options are as follows:

Tag	Description
-d	The -d option turns on socket debugging on the TCP sockets used for communication with the remote host.
-l	By default, the remote username is the same as the local username. The -l option allows the remote name to be specified.
-n	The -n option redirects input from the special device /dev/null (see the BUGS section of this manual page).

Shell metacharacters which are not quoted are interpreted on local machine, while quoted metacharacters are interpreted on the remote machine. For example, the command

rsh otherhost cat remotefile >> localfile

appends the remote file remotefile to the local file localfile, while

rsh otherhost cat remotefile >> other_remotefile

appends remotefile to other_remotefile.

5) rlogin: it is remote login program that connect your terminal to aremote machine's terminal. It is a non-secure protocol as it send all information in plain text.It also enables an implicit trust

relationship to exist between machines, so that you can use rlogn without a password.

rlogin [-8EKLdx] [-e char] [-l username] host

Rlogin starts a terminal session on a remote host host.

The standard Berkeley rhosts authorization mechanism is used. The options are as follows:

Tag	Description
-8	The -8 option allows an eight-bit input data path at all times; otherwise parity bits are stripped except when the remote side's stop and start characters are other than ^S/^Q .
-E	The -E option stops any character from being recognized as an escape character. When used with the -8 option, this provides a completely transparent connection.
-L	The -L option allows the rlogin session to be run in ''litout'' mode.
-d	The -d option turns on socket debugging on the TCP sockets used for communication with the remote host.
-e	The -e option allows user specification of the escape character, which is ''~'' by default. This specification may be as a literal character, or as an octal value in the form \nnn.

6) finger: It enables users on remote system to search information about users on another system. It display a user's login name, real name , terminal name, idle name , login time , office location and phone number . the Finger daemon listens on port 79

finger [-lmsp] [user ...] [user@host ...]

The finger displays information about the system users.

Options are:

Tag	Description
-s	Finger displays the user's login name, real name, terminal name and write status (as a ''*'' after the terminal name if write permission is denied), idle time, login time, office location and office phone number. Login time is displayed as month, day, hours and

	minutes, unless more than six months ago, in which case the year is displayed rather than the hours and minutes. Unknown devices as well as nonexistent idle and login times are displayed as single asterisks.
-l	Produces a multi-line format displaying all of the information described for the -s option as well as the user's home directory, home phone number, login shell, mail status, and the contents of the files ".plan", ".project", ".pgpkey" and ".forward" from the user's home directory. Phone numbers specified as eleven digits are printed as ''+N-NNN-NNN-NNNN''. Numbers specified as ten or seven digits are printed as the appropriate subset of that string. Numbers specified as five digits are printed as ''xN-NNNN''. Numbers specified as four digits are printed as ''xNNNN''. If write permission is denied to the device, the phrase ''(messages off)'' is appended to the line containing the device name. One entry per user is displayed with the -l option; if a user is logged on multiple times, terminal information is repeated once per login. Mail status is shown as ''No Mail.'' if there is no mail at all, ''Mail last read DDD MMM ## HH:MM YYYY (TZ)'' if the person has looked at their mailbox since new mail arriving, or ''New mail received ...'', '' Unread since ...'' if they have new mail.
-p	Prevents the -l option of finger from displaying the contents of the ".plan", ".project" and ".pgpkey" files.
-m	Prevent matching of *user* names. *User* is usually a login name; however, matching will also be done on the users' real names, unless the -m option is supplied. All name matching performed by finger is case insensitive.

If no options are specified, finger defaults to the -l style output if operands are provided, otherwise to the -s style. Note that some fields may be missing, in either format, if information is not available for them.

If no arguments are specified, finger will print an entry for each user currently logged into the system.

Finger may be used to look up users on a remote machine. The format is to specify a user as "user@host", or "@host", where the default output format for the former is the -l style, and the default output format for the latter is the -s style. The -l option is the only option that may be passed to a remote machine.

If standard output is a socket, finger will emit a carriage return (^M) before every linefeed (^J). This is for processing remote finger requests when invoked by fingerd.

FILES

Tag	Description
~/.nofinger	If finger finds this file in a user's home directory, it will, for finger requests originating outside the local host, firmly deny the existence of that user. For this to work, the finger program, as started by fingerd, must be able to see the .nofinger file. This generally means that the home directory containing the file must have the other-users-execute bit set (o+x). See chmod(1). If you use this feature for privacy, please test it with ''finger @localhost'' before relying on it, just in case.
~/.plan	These files are printed as part of a long-format request. The .project file is limited to one line; the .plan file may be arbitrarily long.

7) talk and ntalk: They are real time chat protocols that runs on port 517 and 518 repectively.Type talk or ntalkusername@hostname to send someone else to talk request.If that person's server is running a talk or ntalk daemon and the person is logged in, he or she will see an invitation to chat with you. talk and ntalk aren't as popular as they once were, sinceinstant messenger clients have become very popular.

talk person [ttyname]

Talk is a visual communication program which copies lines from your
terminal to that of another user.

Tag	Description
person	If you wish to talk to someone on your own machine, then person is just the person's login name. If you wish to talk to a user on another host, then person is of the form 'user@host'.
ttyname	If you wish to talk to a user who is logged in more than once, the *ttyname* argument may be used to indicate the appropriate terminal name, where *ttyname* is of the form 'ttyXX' or 'pts/X'.

When first called, **talk** contacts the talk daemon on the other user's machine, which sends the message

```
Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine
```

to that user. At this point, he then replies by typing

```
talk your_name@your_machine
```

It doesn't matter from which machine the recipient replies, as long as his login name is the same. Once communication is established, the two parties may type simultaneously; their output will appear in separate windows. Typing control-L (^L) will cause the screen to be reprinted. The erase, kill line, and word erase characters (normally ^H, ^U, and ^W respectively) will behave normally. To exit, just type the interrupt character (normally ^C); **talk** then moves the cursor to the bottom of the screen and restores the terminal to its previous state.

As of netkit-ntalk 0.15 **talk** supports scrollback; use esc-p and esc-n to scroll your window, and ctrl-p and ctrl-n to scroll the other window. These keys are now opposite from the way they were in 0.16; while this will probably be confusing at first, the rationale is that the key combinations with escape are harder to type and should therefore be used to scroll one's own screen, since one needs to do that much less often.

If you do not want to receive talk requests, you may block them using the **mesg**command. By default, talk requests are normally not blocked. Certain commands, in particular **nroff**, **pine**, and **pr**, may block messages temporarily in order to prevent messy output.

9.2 LINUX MACHINE AS SERVER

Following are the common server protocols available on Linux

1) HTTP: Apache is the most common Web Server used on Linux, which is started out of a system's rc scripts. You can easily configure Apache and its configuration file exist in /etc/httpd/conf/ . Apache mostly listens to port number 80 and can be set to listen to many different network ports.

2) sshd: the secure shell daemon (sshd) is started out of a system's rcscripts. Its global system configuration file exist in /etc/ssh and user's configuration files are placed in $HOME /.ssh/. It listen on port 22.

   Note:sshd can be configured to run on an alternative port. Running SSH on a port other than 22 comes in handy if port 22 is being blocked by a firewall. Running SSH on a different port also adds a small measure of security through obscurity. Automatic scanners used by hackers will miss that SSH is running on your machine if they don't find it running on the standard port they expect.

3) Ftpd: The Ftpd daemon uses port 20 and 21 to listen and start FTP requests. It configuration files ftpaccess , ftpconversions ,ftpgroups, ftphosts and ftpusers are located in /etc directory.

4) DNS: The Domain name Service (DNS) maps IP addresses to hostname. It uses port 53. It configuration file is named.conf in the /etc directory

10

CONFIGURING THE XINETD SERVER

Unit Structure :

10.1 CONFIGURING THE XINETD SERVER

Xinetd perform the same function as inetd with addition of more security and new features.It starts at system boot up and runs the programs that provide Internet services. It waits and listens for connections to come in the ports to which they are assigned in theirconf files. The Xinetd spawns a new server if required to listen to the connection request

It also provides access – control facilities.It doesnot limit its used system administrators only but also to those who are not root.Anyone can start the network services with xinetd .It also support encrypting plain-text services such as ftp command channel by wrapping them in stunnel.xinetd provides better logging facilities, including remote user ,ID, access time and server specific

information. It also kills servers that are no mapped in configuration file and those that violate the configuration’s access criteria.It is security step that prevents denial of services (DOS) attacks by limiting normal functions . For example,xientd can limit the number of incoming connection to prevent network overflow attacks. This in turn prevents the machine from being slowed down.

Type man xinetd.conf command in the terminal window, to see xinetd file in detail

```
#
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/
defaults
{
instances = 60
log_type = SYSLOG authpriv
log_on_success = HOST PID
log_on_failure = HOST
cps = 25 30
}
includedir /etc/xinetd.d
```

The last line of the xinetd.conf file say that all the files in the /etc/xinetd.d directory are read into the xinetd.conf file as well . You can easily enable or disable a service by setting the value of disable = to yes or no

These lines control the following aspects of xinetd:

instances — Sets the maximum number of requests xinetd can handle at once.

log_type — Configures xinetd to use the authpriv log facility, which writes log entries to the /var/log/secure file. Adding a directive such as FILE /var/log/xinetdlog would create a custom log file called xinetdlog in the /var/log/ directory.

log_on_success — Configures xinetd to log if the connection is successful. By default, the remote host's IP address and the process ID of server processing the request are recorded.

log_on_failure — Configures xinetd to log if there is a connection failure or if the connection is not allowed.

cps — Configures xinetd to allow no more than 25 connections per second to any given service. If this limit is reached, the service is retired for 30 seconds.

includedir /etc/xinetd.d/ — Includes options declared in the service-specific configuration files located in the /etc/xinetd.d/ directory. The /etc/xinetd.d/ Directory for more information about this directory.

10.1.2 The /etc/xinetd.d/ Directory

The files in the /etc/xinetd.d/ directory contains the configuration files for each service managed by xinetd and the names of the files correlate to the service. As with xinetd.conf, this file is read only when the xinetd service is started. For any changes to take effect, the administrator must restart the xinetd service.

The format of files in the /etc/xinetd.d/ directory use the same conventions as /etc/xinetd.conf. The primary reason the configuration for each service is stored in a separate file is to make customization easier and less likely to effect other services.

To gain an understanding of how these files are structured, consider the /etc/xinetd.d/telnet file:

```
service telnet
{
	flags		= REUSE
	socket_type	= stream
	wait		= no
	user		= root
	server		= /usr/sbin/in.telnetd
	log_on_failure	+= USERID
	disable		= yes
}
```

These lines control various aspects of the telnet service:

service — Defines the service name, usually one listed in the /etc/services file.

flags — Sets any of a number of attributes for the connection. REUSE instructs xinetd to reuse the socket for a Telnet connection.

socket_type — Sets the network socket type to stream.

wait — Defines whether the service is single-threaded (yes) or multi-threaded (no).

user — Defines what user ID the process process will run under.

server — Defines the binary executable to be launched.

log_on_failure — Defines logging parameters for log_on_failure in addition to those already defined in xinetd.conf.

disable — Defines whether or not the service is active.

10.1. 3. Altering xinetd Configuration Files

There are a large assortment of directives available for xinetd protected services. This section highlights some of the more commonly used options.

10.1.3.1. Logging Options

The following logging options are available for both /etc/xinetd.conf and the service-specific configuration files within the /etc/xinetd.d/ directory.

Below is a list of some of the more commonly used logging options:

ATTEMPT — Logs the fact that a failed attempt was made (log_on_failure).

DURATION — Logs the length of time the service is used by a remote system (log_on_success).

EXIT — Logs the exit status or termination signal of the service (log_on_success).

HOST — Logs the remote host's IP address (log_on_failure and log_on_success).

PID — Logs the process ID of the server receiving the request (log_on_success).

USERID — Logs the remote user using the method defined in RFC 1413 for all multi-threaded stream services (log_on_failure and log_on_success).

10.1.3.2 Access Control Options

Users of xinetd services can choose to use the TCP wrappers hosts access rules, provide access control via the xinetd configuration files, or a mixture of both. Information concerning the use of TCP wrappers hosts access control files can be found in TCP Wrappers Configuration Files.

This section discusses using xinetd to control access to services.

The xinetd hosts access control differs from the method used by TCP wrappers. While TCP wrappers places all of the access configuration within two files, /etc/hosts.allow and /etc/hosts.deny, xinetd's access control is found in each service's configuration file within the /etc/xinetd.d/ directory.

The following hosts access options are supported by xinetd:

only_from — Allows only the specified hosts to use the service.

no_access — Blocks listed hosts from using the service.

access_times — Specifies the time range when a particular service may be used. The time range must be stated in 24-hour format notation, HH:MM-HH:MM.

The only_from and no_access options can use a list of IP addresses or host names, or can specify an entire network. Like TCP wrappers, combining xinetd access control with the enhanced logging configuration can increase security by blocking requests from banned hosts while verbosely recording each connection attempt.

For example, the following /etc/xinetd.d/telnet file can be used to block Telnet access from a particular network group and restrict the overall time range that even allowed users can log in:

```
service telnet
{
	disable		= no
	flags		= REUSE
	socket_type	= stream
	wait		= no
	user		= root
	server		= /usr/sbin/in.telnetd
	log_on_failure	+= USERID
	no_access	= 10.0.1.0/24
	log_on_success	+= PID HOST EXIT
	access_times	= 09:45-16:15
}
```

In this example, when client system from the 10.0.1.0/24 network, such as 10.0.1.2, tries to access the Telnet service, it receives a message stating the following message:

Connection closed by foreign host.

In addition, their login attempts are logged in /var/log/secure as follows:

May 15 17:38:49 boo xinetd[16252]: START: telnet pid=16256 from=10.0.1.2

May 15 17:38:49 boo xinetd[16256]: FAIL: telnet address from=10.0.1.2

May 15 17:38:49 boo xinetd[16252]: EXIT: telnet status=0 pid=16256

When using TCP wrappers in conjunction with xinetd access controls, it is important to understand the relationship between the two access control mechanisms.

The following is the order of operations followed by xinetd when a client requests a connection:

The xinetd daemon accesses the TCP wrappers hosts access rules through a libwrap.a library call. If a deny rule matches the client host, the connection is dropped. If an allow rule matches the client host, the connection is passed on to xinetd.

The xinetd daemon checks its own access control rules both for the xinetd service and the requested service. If a deny rule matches the client host the connection is dropped. Otherwise, xinetd starts an instance of the requested service and passes control of the connection to it.

10.1.3.3 Binding and Redirection Options

The service configuration files for xinetd support binding the service to an IP address and redirecting incoming requests for that service to another IP address, hostname, or port.

Binding is controlled with the bind option in the service-specific configuration files and links the service to one IP address on the system. Once configured, the bind option only allows requests for the proper IP address to access the service. This way different services can be bound to different network interfaces based on need.

This is particularly useful for systems with multiple network adapters or with multiple IP addresses configured. On such a system, insecure services, like Telnet, can be configured to listen only on the interface connected to a private network and not to the interface connected with the Internet.

The redirect option accepts an IP address or hostname followed by a port number. It configures the service to redirect any requests for this service to the specified host and port number. This feature can be used to point to another port number on the same system, redirect the request to different IP address on the same machine, shift the request to a totally different system and port number, or any combination of these options. In this way, a user connecting to certain service on a system may be rerouted to another system with no disruption.

The xinetd daemon is able to accomplish this redirection by spawning a process that stays alive for the duration of the connection between the requesting client machine and the host actually providing the service, transferring data between the two systems.

But the advantages of the bind and redirect options are most clearly evident when they are used together. By binding a service to a particular IP address on a system and then redirecting requests for this service to a second machine that only the first machine can see, an internal system can be used to provide services for a totally different network. Alternatively, these options can be used to limit the exposure of a particular service on a multi-homed machine to a known IP address, as well as redirect any requests for that service to another machine specially configured for that purpose.

For example, consider a system that is used as a firewall with this setting for its Telnet service:

```
service telnet
{
        socket_type             = stream
        wait                    = no
        server                  = /usr/sbin/in.telnetd
        log_on_success                  += DURATION USERID
        log_on_failure          += USERID
        bind                = 123.123.123.123
        redirect             = 10.0.1.13 23
}
```

The bind and redirect options in this file ensures that the Telnet service on the machine is bound to the external IP address (123.123.123.123), the one facing the Internet. In addition, any requests for Telnet service sent to 123.123.123.123 are redirected via a second network adapter to an internal IP address (10.0.1.13) that only the firewall and internal systems can access. The firewall

then send the communication between the two systems, and the connecting system thinks it is connected to 123.123.123.123 when it is actually connected to a different machine.

This feature is particularly useful for users with broadband connections and only one fixed IP address. When using Network Address Translation (NAT), the systems behind the gateway machine, which are using internal-only IP addresses, are not available from outside the gateway system. However, when certain services controlled by xinetd are configured with the bind and redirect options, the gateway machine can act as a proxy between outside systems and a particular internal machine configured to provide the service. In addition, the various xinetd access control and logging options are also available for additional protection.

10.1.3.4. Resource Management Options

The xinetd daemon can add a basic level of protection from a Denial of Service (DoS) attacks. Below is a list of directives which can aid in limiting the effectiveness of such attacks:

per_source — defines the maximum number of instances for a service per source IP address. It accepts only integers as an argument and can be used in both xinetd.conf and in the service-specific configuration files in the xinetd.d/ directory.

cps — Defines the maximum of connections per second. This directive takes two integer arguments separated by white space. The first is the maximum number of connections allowed to the service per second. The second is the number of seconds xinetd must wait before re-enabling the service. It accepts only integers as an argument and can be used in both xinetd.conf and in the service-specific configuration files in the xinetd.d/ directory.

max_load — defines the CPU usage threshold for a service. It accepts a floating point number argument.

10.2 COMPARE XINETD AND STANDALONE

Services can better utilized if they are configured properly.To Control a service, you must know the origin. You must aware regarding the startup of services, that is, either they are spawned from super servers such as xinetd , or started on their own. The key difference between services that run standalone and those run by the xinetd server is the amount of configuration information

required. The other difference is the availability of services, means whether to make that service constantly available or whether to make them available only on incoming request

Let's discuss some services started from xinetd and standalone services in the following /etc/xinetd.d directories as each service has its own file located in this directory.

- **chargen** — The chargen service was intended for testing and measurement purposes and may listen on both TCP and UDP protocols. Upon opening a TCP connection, the server starts sending arbitrary characters to the connecting host and continues until the hosts closes the connection
- **daytime-udp** — A daytime server listens for client requests on port 13. When it receives a message from a client, a daytime server replies to that client with its current date and time over time.
- **finger** — finger programs that provide status reports on a particular computer system or a particular person at network sites.
- **kshell** — Restricts user access to the shell
- **rlogin** — is a UNIX command that allows an authorized user to login to other UNIX machines (hosts) on a network and to interact as if the user were physically at the host computer.
- **swat** — SWAT allows a Samba administrator to configure the complex smb.conf file via a Web browser. In addition, a swat configuration page has help links to all the configurable options in the smb.conf file allowing an administrator to easily look up the effects of any change.
- **time** — Gives you the time
- **echo** — Echoes back all characters sent to it over TCP
- **gssftp** — A kerberized xinetd-based FTP daemon which does not pass authentication information over the network.
- **rsh** — The remote shell (rsh) is a command line computer program that can execute shell commands as another user, and on another computer across a computer network.
- **talk** — A talk (real-time chat) server

- **time-udp** — Gives you the time over UDP
- **comsat** — Comcast, an Internet Service Provider (ISP) is s providing services of critical end-user notifications to web browsers .Such anotification system is being used to provide near-immediate notifications to customers, such as to warn them that their traffic exhibits patterns that are indicative of malware or virus infection.
- **echo-udp** — Echoes back all characters sent to
- **klogin** — This file is used by the Kerberos authentication system, it contains a list of trusted users who can login into your account and lives in your home directory.
- **ntalk** —ntalk is a chat application. It works in a client-server model and it's designed to work in all kinds of networks. ntalk was written to be much more easy-to-use, friendly and functional than Unix talk
- **rsync** — rsync is a utility for efficiently transferring and synchronizing files across computer systems, by checking the timestamp and size of files
- **telnet** — Telnet is a protocol used on the Internet or local area networks to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.
- **wu-ftpd** — WU-FTPD (more fully wuarchive-ftpd, also frequently spelled in lowercase as wu-ftpd) is a freeFTP server software (daemon) for Unix-like operating systems.
- **daytime** — Gives you the time over TCP
- **eklogin** — eklogin is the same as klogin but with encryption. There is no longer ekshell port because encrypted and normal connection use the same port (kshell).
- **krb5-telnet** — The telnet protocol has the ability to negotiate an authentication mechanism. We have configured the telnet client program on most of our Unix machines to use kerberos authentication if it is available.
- **rexec** — rexec stands for remote exec and like rsh, allows you to execute non-interactive programs on another system.

- **sgi_fam** — A Silicon Graphics daemon that is an RPC server that tracks changes to the filesystem under the IRIX operating system.

- **tftp** — Trivial File Transfer Protocol (TFTP) is a simple lockstep File Transfer Protocol which allows a client to get a file from or put a file onto a remote host.

10.2.1Standalone Services

The following service are standalone as they are started on their own from the rc scripts . you can enable and disable these services from rc directories.

- **apache** — Apache Web Server is designed to create web servers that have the ability to host one or more HTTP-based websites. **sshd** — SSH server

- **sendmail** — On the Internet, sendmail is the most popular UNIX-based implementation of the Simple Mail Transfer Protocol (SMTP) for transmitting e-mail .

- **qmail** — qmail is a mail transfer agent (MTA) that runs on Unix

- **postfix** — Postfix is a free and open-source mail transfer agent (MTA) that routes and delivers electronic mail.

- **thttpd** — Tt is a simple, small, fast, and secure HTTP server. It doesn't have a lot of special features, but it suffices for most uses of the web, it's about as fast as the best full-featured servers (Apache, NCSA, Netscape), and it has one extremely useful feature (URL-traffic-based throttling) that no other server currently has.

- **boa** — Boa is a tiny web server that also offers extremely high performance. It is specifically designed to run on UNIX-like systems, which includes Linux, as well as the *BSD systems.

- **named** — DNS server

- **xfs** — xfs is the X Window System font server. It supplies fonts to X Window System display servers.

- **xdm** — XDM is the default display manager for the X Window System. It is a bare-bones X display manager.

- **portmap** — **It** is a server that converts RPC program numbers into DARPA protocol port numbers.

- **rpc.quotad** — rquotad is an rpc server which returns quotas for a user of a local filesystem which is mounted by a remote machine over the NFS
- **knfsd** — This is a much-improved Linux NFS server with support for NFSv3 as well as NFSv2
- **rpc.mountd** — The rpc.mountd daemon implements the server side of the NFS MOUNT protocol, an NFS side protocol used by NFS version 2 and NFS version 3.
- **rpc.ypbind** — it finds the server for NIS domains and maintains the NIS binding information.
- **Squid**— itis a full-featured web proxy cache server application which provides proxy and cache services for Hyper Text Transport Protocol (HTTP), File Transfer Protocol (FTP), and other popular network protocols.
- **nessusd** — Nessus has been deployed for vulnerability, configuration and compliance assessments and prevents network attacks by identifying the vulnerabilities and configuration issues that hackers use to penetrate your network.
- **postgresql** — Itis an open-source, object-relational database management system (ORDBMS) that is not owned or controlled by one company or individual.
- **mysql** — It is an open source relational database management system (RDBMS) based on Structured Query Language
- **oracle** — An Oracle Database (RDBMS) is a collection of data organized by type with relationships being maintained between the different types

10.3. CONFIGURING LINUX FIREWALL PACKAGES

Firewall is a general term used to describe methods for permitting or denying access to a network or server. Firewalls perform a variety of services to protect your network. Firewalls monitor the traffic that comes between your network and the internet to prevent attack from unauthorized services. Linux provide system security through firewall package mechanism that is ,

iptables.It is the primary firewall package which enables you to run a personal firewall to protect your Linux Machine.

10.3.1 Iptables Config File

The default config files for RHEL / CentOS / Fedora Linux are:

/etc/sysconfig/iptables – The system scripts that activate the firewall by reading this file.

10.3.2 Display Default Rules

Type the following command:

iptables --line-numbers -n –L

Sample Output :

Chain INPUT (policy ACCEPT)

num target prot opt source destination

1 RH-Firewall-1-INPUT all -- 0.0.0.0/0 0.0.0.0/0

Chain FORWARD (policy ACCEPT)

num target prot opt source destination

1 RH-Firewall-1-INPUT all -- 0.0.0.0/0 0.0.0.0/0

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

Chain RH-Firewall-1-INPUT (2 references)

num target prot opt source destination

1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0

2 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 icmp type 255

3 ACCEPT udp -- 0.0.0.0/0 224.0.0.251 udp dpt:5353

4 ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:53

5 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED

6	ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0 state NEW tcp dpt:22
7	ACCEPT	tcp	--	0.0.0.0/0	0.0.0.0/0 state NEW tcp dpt:53
8	REJECT	all	--	0.0.0.0/0	0.0.0.0/0 reject-with icmp-host-prohibited

10.3.3Turn On Firewall

Type the following two commands to turn on firewall:

chkconfig iptables on

service iptables start

restart the firewall

service iptables restart

stop the firewall

service iptables stop

10.3.4 Understanding Firewall

There are total 4 chains:

1. **INPUT** – The default chain is used for packets addressed to the system. Use this to open or close incoming ports (such as 80, 25, and 110 etc.) and ip addresses / subnet (such as 202.54.1.20/29).

2. **OUTPUT** – The default chain is used when packets are generating from the system. Use this open or close outgoing ports and ip addresses / subnets.

3. **FORWARD** – The default chains is used when packets send through another interface. Usually used when you setup Linux as router. For example, eth0 connected to ADSL/Cable modem and eth1 is connected to local LAN. Use FORWARD chain to send and receive traffic from LAN to the Internet.

4. **RH-Firewall-1-INPUT** – This is a user-defined custom chain. It is used by the INPUT, OUTPUT and FORWARD chains.

10.3.5 Packet Matching Rules

1. Each packet starts at the first rule in the chain.
2. A packet proceeds until it matches a rule.
3. If a match found, then control will jump to the specified target (such as REJECT, ACCEPT, DROP).

10.3.6 Target Meanings

1. The target ACCEPT means allow packet.
2. The target REJECT means to drop the packet and send an error message to remote host.
3. The target DROP means drop the packet and do not send an error message to remote host or sending host.

/etc/sysconfig/iptables

Edit /etc/sysconfig/iptables, enter:

vi /etc/sysconfig/iptables

You will see default rules as follows:

```
filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 53 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
```

-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 53 -j ACCEPT

-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited

COMMIT

10.3.7 Drop All Traffic

Find lines:

*filter

:INPUT ACCEPT [0:0]

:FORWARD ACCEPT [0:0]

:OUTPUT ACCEPT [0:0]

Update as follows to change the default policy to DROP from ACCEPT for the INPUT and FORWARD built-in chains:

:INPUT DROP [0:0]

:FORWARD DROP [0:0]

11

THE DOMAIN NAME SYSTEM

Unit Structure

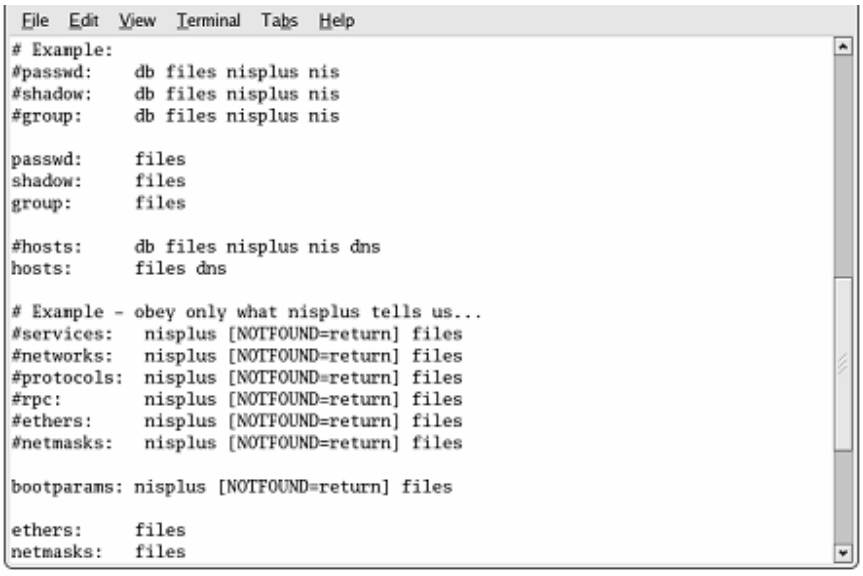
11.1 UNDERSTANDING DOMAIN NAME SYSTEM (DNS)

DNS Provides scalable and dispersed lookup mechanism that translate domain names into IP Addresses. The Internet works on the basis of Ip addresses. Whenever you use a domain name, a DNS service translates the name in to the corresponding IP address. For Example, the domain name www.example.com might translate to 198.123.124.7

Let's discuss the domain name and their organization using the domain name tactechnology.com for example: welfare.com the first part of this domain name is the name of the institution, company or organization. The part after the dot (.) is called the top-level domain (TLD). There are many top level domains Following Table showing the some of the commonly used top-level domains.

TOP-LEVEL DOMAIN	MEANING
com	A TLD typically used to register a business
edu	A TLD assigned to an educational institution
gov	A TLD given to a U.S. government agency
mil	A TLD used by a branch of the U.S. military
net	A TLD used by a network affiliated organization
org	A TLD used by a noncommercial organization
int	An TLD assigned to an international organization
us	The U.S. domain, with each listing as a lower level
biz	Another TLD used for businesses
info	A TLD that can be used for any type of organization
name	A TLD used to register sites for individuals
museum	A TLD used to register a museum
coop	A TLD assigned to a cooperative organization
aero	A TLD used by the air transport industry
pro	A TLD not yet active but assigned to professions
travel	A TLD that should be available in late 2005 used by travel related companies

When you type a host name you system uses its resources to resolve names in to IP addresses. It basically looks in/etc/nsswitch.conf file for the host information. Following is the /etc/nsswitch.conf. The line showing host tells the system to first look at the local files and then uses the DNS to resolve the name in to IP address



The nsswitch.conf file tells the resolver routines where to look for IP addresses.

One of the local files searched is the /etc/hosts file,wuhichconatins IP address and hostnames that you used on your network. Using this file is not beneficial in large networks as it is difficult to keep this file up to date. After this system look for the IP address in the /etc/ resolv.conf. Follow is the /etc/resolv.conffile. This file contains the IP address of computers as the name servers.

the /etc/resolv.conf

: generated by /sbin/dhcpclient-script

Searchloacldomain

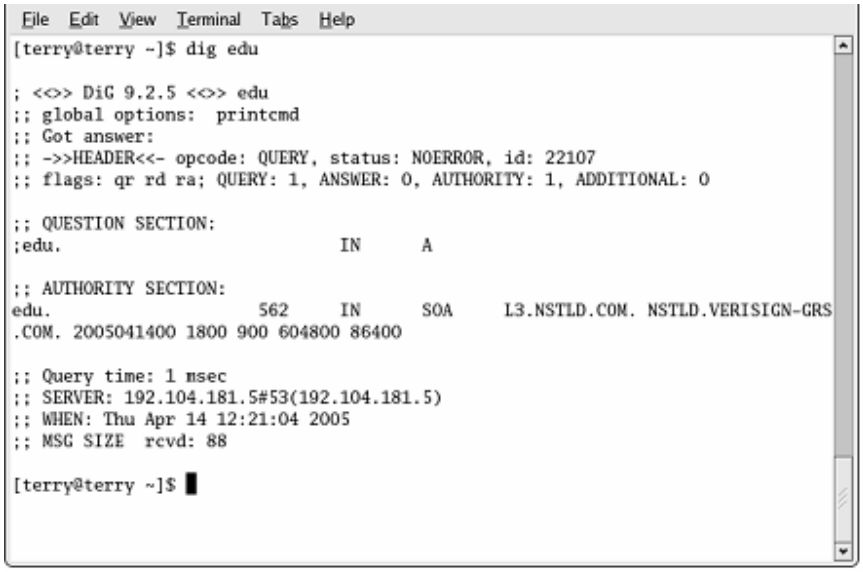
Nameserver 192.168.204.2

You can list up to three name servers, so that system can look for secons connection if first connection is not responding.

11.2 UNDERSTANDING TYPES OF DOMAIN SERVERS

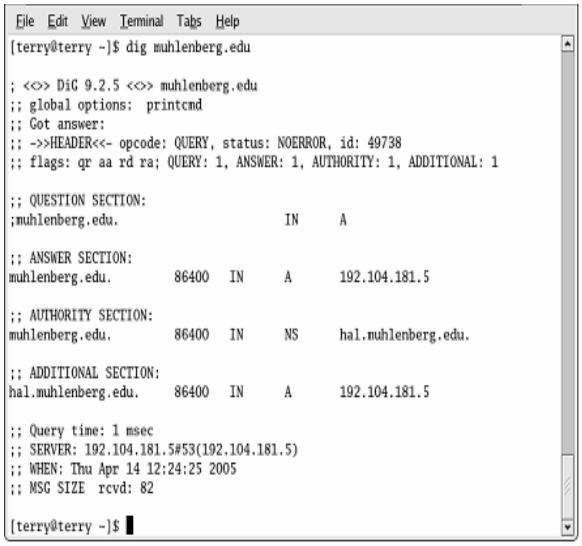
A top-level domain server is referred toas root name server as it provides information about the domains. For example if you want to search www.muhlenberg.edu, it looks to the root name server for .edu for information. Then the root name server directs the search to a lower-level domain name server until information is found

In Figure 1, you use dig edu command to search the information about .edu domain



A search for the top-level root name servers for .edu.

You can continue the search for the second by adding the name of the domain as shown in figure



A search for the second-level domain shows the authoritative name server.

After finding the domain, information about the domain is provided by its local domain servers. There are three types of local domain name servers.

Master: It is listed as an authoritative server as it contains all the information about the domain and provides this information when needed.

Slave: It maintain the backup of the master server (contains same information as in master server). It is used when master server goes down or is not available.

Caching: It is used to provide information to other servers and work stations on the local network. Caching server speeds up searches as the domain information is already stored in the memory

In the next section you will learn the source from where master server and slave servers gather information about domains. You have to configure server to provide this information when needed.

Let's learn to configure a server to provide domain name information.

11.3 EXAMINING SERVER CONFIGURATION FILES

Before you configure your server to provide domain name information. You will need to configure some files. This files depends on the BIND Server install in your system

Note: The most common DNS Server used in the Linux distribution is BIND or the Berkeley Internet Name Daemon. You can download the latest version of bind from www.isc.org. Type the following command in the terminal to install BIND

Yum –y install bind

You need to install BIND to configure DNS server

There are five file that are required to set up the named server .They are follows.

Named conf: this file is located in /etc directory and contain global properties and source configuration file.

Named.ca: This file is located in /var/named directory and contains the name and addresses of the root servers.

Named. local. This file is located in /var/named directory and provides information for resolving the loopback address for local host.

Zone: this filecontains the named and addresses of servers and workstation in the local domain and provide mapping of these names to IP address.

Reverse zone: This file provides information to map IP addresses to names.

The first three files are required irrespective of the configuration as a master, slave or caching. Only server and last two files are required for the master domain server.

To start the BIND Sever, /etc/rc.d/init.d/named script is used

Let's study the named.conf files in detail

Named.conf

Following is the content of /etc/named.conf

```
//
// named.conf for Red Hat caching-nameserver
//
options {
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
/

* If there is a firewall between you and name servers you want
* to talk to, you might need to uncomment the query-source
* directive below. Previous versions of BIND always asked
* questions using port 53, but BIND 8.1 uses an unprivileged
* port by default.
*/
// query-source address * port 53;
};
//
// a caching only nameserverconfig
//
controls {
inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};
zone "." IN {
type hint;
file "named.ca";
};
```

```
zone “localdomain” IN {
type master;
file “localdomain.zone”;
allow-update { none; };
};
zone “localhost” IN {
type master;
file “localhost.zone”;
allow-update { none; };
};
zone “0.0.127.in-addr.arpa” IN {
type master;
file “named.local”;
allow-update { none; };
};
include “/etc/rndc.key”;
```

Lines starting with // are the comment lines. Commands are passed to the file in the form of statements. Information about the statement is contained within curly brackets’ {}’ and terminated with a ‘;’ semicolon. Several command are shown above. You can have the following configuration statements in named.conf file

- **options** — Lists global configurations and defaults
- **include** — Gets information from another file and includes it
- **acl**— Specifies IP addresses used in an access control list
- **logging** — Specifies log file locations and contents
- **server** — Specifies properties of remote servers
- **zone** — Specifies information about zones
- **key** — Specifies security keys used for authentication

Options

The options statement is the first section of the named .conf and is used to specify the location of the named working directory. Only one options statement should define in a named.conf file. The options statements defines global server configuration options and sets defaults for other statements. You can have more than one value for the options statement

The option statement uses the following statement

Options {

Value “property”;

};

In the above statement, the value directives are replaced with the valid option. The list of values that can be defined in options statement is a long list; following are some commonly used options:

Allow –query: it specifies the host that are allowed to query the name server. By Default, all hosts are allowed to query.

Allow – recursion: it applies to recursive queries. By default, all host are allowed to perform recursive queries on the name server

Blackhole: it specifies lists of host that are not allowed to query the server.

Directory: it specifies the named working directory

Forward: it specifies the forwarding behavior of aforwarder directives. If set to first, the servers listed in the forwarder option are queried first and then the server tries to find the answer itself. If set to only, just the servers in the forwarders list are queried

Forwarders: it specifies a list of valid IP addresses for name servers where requests are forwarded

Listen –on: It specifies the port and interfaces on which server listen for queries (default port 53)

Notify: it Control whether named notifies the slave servers when a zone is updated. It accepts the following option: yes or no.

Pid-file: It specifies the name of the files holding the process ID

Statistics-file: It specifies an alternate location for statistics files. By default, named statistics are saved to the /var/named/ anmed.stats file

Include

The include statement allow to lists the path and name of any file that you want to be include in a named.conf. An include statement uses the following syntax:

Include “filename”

In this statement, filename is replaced with a path to a file.

ACL

Access Control List (ACL) defines one or more lists of IP addresses. The acl statement defines groups of host which are permitted or denies access to the name server.

An acl statement takes the following form

Ac<acl_name>{

<match-element>;

[<match-element>;…..]

}

In this statement, replace <acl-name> with the name of the access control list and replace <match-element> with a semi-colon separated list of IP address.

Logging

In logging statement you can specify your server’s logging option. It contains of channel and category .In the channel you can specify the location of the logged information. Logged information can be written to a file, sent to the syslog or thrown away by specifying an appropriate command. Once a customized channel has been defined, a category option is used to categorize the channel and is responsible to start logging when named is restarted.

Following is the syntax of logging statement

logging{
channel*channel_name*{
file*path to file*
versions*specify number or unlimited*
size specify size in bytes }
}

To discard the information, choose null as the destination.
logging {
channel*channel_name*{
null;}
}

The category is where you specify the type of information to log. This value follows the severity and print parameters and takes the following syntax:

category*category name* {
channel name; channel name;
};

You can choose from over 20 categories.

Cname : Information about CNAME references
config : Information about configuration files
db : Information about databases
default: The default if nothing is selected
eventlib : Information about event system debugging
insist: Details about failures from internal consistency checking
lame-servers: Information about LAME servers
load : Information about zone loading
maintenance: Information about maintenance
ncache: Information about negative caching
notify : Information about tracing the NOTIFY protocol
os: Information about operating system problems
packet: Dumps of all sent and received packets
panic : Information about faults that shut down the server
parser : Information about processing configuration commands
queries: Information about all received DNS queries
response-checks: Information about response-checking results
security : Information about security status of server
statistics: Information about server statistics
update: Information about dynamic updates
xfer-in : Information about inbound zone transfers
xfer-out: Information about outbound zone transfers

You can set the severity level for the information, syntax for setting severity level is as follow:

severitychoose from *critical,error,warning,notice,info,debug*\
level,dynamic
print-timechoose *yes* or *no*
print-severitychoose *yes* or no
print-categorychoose *yes* or *no*
};

Server

In server statement, you can define the properties or behaviors of remote server. You can specify how local server access or respond to a defined remote server, especially in regards

to notification and zone transfer. Server statement syntax is same as other statements following are the valid values for server statement:

Bogus: specify 'yes' for not querying remote server and 'no' to send queries to the remote server.

Transfer: specify the number of transfer allowed

Transfer –format: specify whether you want one-answer and many-answers

Key:specify key ID

Zone

The zone statements refer to the zone files. Each statement starts with the words zone followed by the domain name and data class. The data classes are in,hs, Hesiod and chaos. The data class in is default for internet. Next followed by the type option specifies the server as a master, slave/stub or hint file. A Stub server loads only the NS records. The hints file is used to initialize the root cache and contains a list of root servers. Further it is followed with the name of the zone file for the specified zone. The most common options for the zone statements are as follows

Allow – query: It specify the clients that are allowed to request information about this zone (default= allow all)

Allow-transfer: it specifies the slave servers that are allowed to request a transfer of the zone's information (default= allow all)

Allow –update: it specifies the host that are allowed to dynamically update information in their zones (default = deny)

File: It specifies the name of the file in the named working directory that contains the zone's configuration data.

Masters: it specifies the IP addresses from which to request authoritative zone information

Notify: It specifies whether or not named notifies the slave servers when a zone updated. This directive accepts yes or no option

A Zone statement takes the following form

Zone<zone-name><zone-class>{
<zone-options>;
[<zone-options>;…..]
};

In this statement,

<zone-name> is the name of the zone
<zone-class> is the optional class of the zone , and
<zone-options > is a list of option characterizing the zone

Zone files contained resource record (RR) about IP address . A zone file can contain the following types of RRs:

Start of Authority (SOA)

It is the first line in the zone file. It identifies the name server as the authoritative source of information about that domain. There is only one SOA in each zone file . SOA contains the following data :

```
@ IN SOA main.tactechnology.com.        mail.tactechnology.com.
(/
2000052101  ; Serial
8h                ;Refresh
2h                ;Retry
1w                ;Expire
1d)               ;Minimum TTL
```

The first character in the SOA line is a special symbol that means “to look at this domain.” IN means Internet. SOA means Start of authority. In this example, the authoritative server for this domain is main.tactechnology.com.And mail.tactechnology.com. Is the e-mail address of the administrator?

The information within the parenthesis is passed to other name servers, secondary masters that use this information to update their records. The line containing 2000052101; Serial is the serial number of the file. Secondary servers compare this number with their stored information. If the serial numbers are different, the file is downloaded to update the information in the secondary server. There is no need to download a file if the serial numbers are same. The semicolon indicates the comment.

In the above following are the comments used:

Refresh — the amount of time the server should wait before refreshing its data.

Retry — the amount of time the server should wait before attempting to contact the primary server if the previous attempt failed.

Expire — Means that if the secondary master is unable to contact a primary master during the specified period, the data expires and should be purged.

TTL — specifies the time to live for the data. This parameter is intended for caching name servers and tells them how long to hold the data in their cache.

The Reverse Zone File

The method of using a name to find an ip number is known as forward address resolution. You can also find a name from an Ip Number. This method is called reverse address resolution. It requires the use of a reverse zone file. Through this method, the server returns the domain name when you enter the IP address

Key

It defines shared keys uses to control and authenticate operation such a dynamics DNS and the remote control channel.

After understanding the server configuration files, let configuring a caching DNS server.

11.4. CONFIGURING A CACHING DNS SERVER

As you learn in the previous section three file: named.conf,named.ca and named.local make up the entire configuration in a caching –only server.

Installation of BIND. Also create these file by default. But it is always a good idea to verify the existence of these files.

Start with verifying the zone information in /etc/named.conf. There are two zone lines: one indicating ‘.’, reference to named.ca file and other referencing named.local.

Check the configuration of the /var/named/named.local file. This file contains the domain information for the local host. This file is

generally created when BIND is installed and therefore you do need to make any changes to this file
Now check the /etc/nsswitch file to ensure the following line.

Hosts : files dns

Check the /etc/resolv.conf to ensure the IP address of your local host is listed as a name server. Finally, you need to check /etc/host.confcontains the word bind

After checking the above files configurations now start the named daemon. Type the following command service named start and press enter, wait for the prompt to return and then type rndcstatus. You get the following output:

number of zones: 8
debug level: 0
xfers running: 0
xfers deferred: 0
SOA queries in progress: 0
query logging is off
server is up and running

11.5 CONFIGURING A SECONDARY MASTER DNS SERVER

To configure a secondary master DNS server you need to modify the /etc/named.conf file and add two files to complete the slave configuration. In the /etc/named.conf file add two zones, one for the forward lookup of your server and one for the reverse lookup.

In the following example, the master server is called main.tactechnology.com and the slave is p200.tactechnology.com

Add the following for the forward look up

zone “tactechnology.com” {
notify no;
type slave;
file “tactech.com”;
masters { 192.168.1.1; };
};
For the reverse lookup, you add this section:

```
zone “1.168.192.in-addr.arpa” {
notify no;
type slave;
file “tac.rev”;
masters { 192.168.1.1; };
};
```

After modifying the /etc/named.conf file, the configuration of the slave server is complete and you can move on to configuring the master server.

11.6 CONFIGURING A PRIMARY MASTER SERVER

The /etc/named.conf file on the master server needs to be modified .Add the following lines to the /etc/named.conf

```
zone “tactechnology.com” {
notify no;
type master;
file “tactech.com”;
};
```

For the reverse lookup you add this section:

```
zone “1.168.192.in-addr.arpa” {
notify no;
type master;
file “tac.rev”;
};
```

Always use the same names for the files on the master server as the slave server. As these files are downloaded by the slave in a zone file transfer and stored on the slave in the files shown by the file option. Now create the zones files that are referenced by the /etc/named.conf. Then add name server and mail exchange information. Finally add information about your local host, mail, FTP and web server. Now set up the reverse look up zone file, and add the information about name servers and their IP address.

After the following changes, restart the named daemon using following command

Service named restart
Your name server is ready to use after all the above steps

11.7 CHECKING CONFIGURATION

You can use several tools to check you configuration. Let's perform the following step

The host command enables you to find an ip address for the specified domain name. For this purpose, you require the domain name of the remote host. You get the following result from the host command.

[root@laptop root]# host tactechnology.com

tactechnology.com has address 12.129.206.112

By using the – t option, you can search for resource record types. For example, to search mail server for a domain, type the following command

[root@terry named]# host -t mx tactechnology.com
tactechnology.com mail is handled by 10 mail.tactechnology.com.

The dig command is used for debugging and obtaining other useful information

dig (@server) domain name (type)

dig can also be used to do reverse lookups by using –x switch and specifying the IP address

dig –x domain name

You can obtain the exact same servers from both the dig command

12

CONFIGURING MAIL SERVICES

Unit Structure

The main of the ELECTRONIC MAIL is send the messages to and receive the messages from other computer persons resides any part of the world. The Terms E-Mail is commonly used an Known word in day to day life in communication world. In this chapter we will see a how the email system is been configured and how the E-Mail system works.

12.1 E- MAIL EXPLAINED

The sending an E-mail message, just like sending the letter through regular mail. Starts with the sender and ends with the receiver, in the between the senders and receiver, the postal worker are come in the picture whose responsibility is send the letter from source place to destination place. The senders and receivers doesn't see the functionality of postal workers for moving the letter to source place to destination place. Similarly the Electronic Mail does the job and although there are not many people between the sender and receiver,the program perform the same functionality as the postal workers does , the program uses the network protocol to do the job of sending the message and ensure that the message is send from sender and rich to the destination properly .

In this chapter, we see how you configure the e-mailrun over the TCP/IP protocols. Before configuring the email server and e-mail client, you need to understand how the email systems works and how the programs uses the message and available the message to the different users. In email Communication , there are several components play the roles while sending the message and

receiving the message from the different users and how the email components uses to work the email system properly and as the system administration it is your responsibility to configure the email components. The following components are

- Program

1. A **Mail User Agent (MUA)** is component are used by the users that are able to read and write e-mail
2. A **Mail Transfer Agent (MTA)** is a component are able to deliver the e-mail messages between computers across a networks.
3. A **Local Delivery Agent (LDA)**is a component are able to deliver messages to users' mailbox files
4. A**mail notification** program to tell users that they have the new mail in the mail box.

   The TCP /IP protocol is used to store the e-mail message and transferring the email message between the MTAs.

- The other communication and mail storage components are as follows.

1. Ports

2. Mail queues

3. Mailbox files.

In This sections you will able to track the email message through the process of sending the message to delivering the message to the recipients and how all the component perform their jobs while delivering the message. After the learning the functionality of the components of the email systems. You needs to configure these components to build full-fledged email system for yours servers and clients.

12.1.1MAIL USER AGENT (MUA)

A Mail User Agents is a program that allow you and your users able to receive and sends e mail messages. The MUA is also called a mail client or e-mail agent that are enables users to reads and write the emails message. There are two types of MUA are one is in the form of Graphical User Interface (GUI) such as Microsoft Outlook , Netscape Messenger and another one is Command Line Interface (CLI) such as Pine , mutt.

Whether you are using the GUI Based or CLI Based MUA, after the message is composed, the MUA is send the composed

message to the Mail Transfer Agent (MTA), The MTA is a program is used to send the message out across the networks. The user does not aware the functionality of MTA, It only see the E-Mail client program.

12.1.2 Mail Transfer Agent (MTA)

MTA receives the message from MUA, then MTA perform its job,the default MTA is installed in the RedHat system is sendmail and postfix. The job of MTA is transfer the email message from one node on the network to another node. First of all, MTA reads the information fromTo section of email message and find out the ip address of the recipients mail servers .The MTA client tries to open a connection to the MTA recipient's server, on the port number 25. If the client MTA establishes the connection with MTA recipient's servers, then it able to send the message on the recipient's server using the Simple message transfer protocol.

The recipient's MTA adds the header information to the message. The header information is contains the information is used to track the message and it's ensure that the message correctly delivered to the recipient.

12.1.3 Local Delivery Agent (LDA)

After that LDA receive the message from the MTA and deliver the message to the user's mailboxesby identifying the username. In the Red Hatsystem there is program called procmail. The location of the procmail in the directory of Linux filesytems /usr/spool/mail/<user name>.

The lasts step in these process is when the intended receiver receive the message from his mailbox reads his message with the help of program called MUA on his computer.

An optional program in the mail is called mail notifier that use to check the mail periodically in the users mailbox for arrivals of new mails .If that program is install in the system. It's notifies the users for the arrivals of new mails.

12.2 INTRODUCING SMTP

The purpose of the Simple Mail Transfer Protocol is to manage the transfers of electronic mail from one computer to another. SMTP specifies the movement of message travel from one computer system to another computer system within the networks or across the networks and also specifies the movement of message between the MTA or MTAs on the another networks. The message may directly travels from sending MTA to the receiving

MTA or across the MTAs over the networks. These computer usually store the message in the queue before it transmit to the other MTA, if it is local to the MTA, or to a gateway that sends it to an MTA on another network.

The main properties of SMTP protocol transfer the message in ASCII text only. It does not handle the message in the fonts,colours,graphics,videos, audios format. If you to send the message in other format, then it will required another protocol to send the items.

The protocol support another message format is called as Multipurpose Internet Mail Extensions, or MIME. MIME enables you to send the message in colour sound and other form of data to the message enabling them to be delivered by SMTP. In order for MIME to work, you must have a MIME-compliant MUA.

12.2.1 Understanding POP3

The full of POP 3 is Post Office Protocol version 3. POP3 (Post Office Protocol 3) is the most recent version of a standard protocol for receiving e-mail. POP3 is a client/server protocol in which e-mail is received and held for you by your Internet server. Periodically, you (or your client e-mail receiver) check your mail-box on the server and download any mail, probably using POP3. This standard protocol is built into most popular e-mail products, such as Eudora and Outlook Express. It's also built into the Netscape and Microsoft Internet Explorer browsers.

The POP3 servers issued to store the message as it receive. Without the POP3 the message cannot be sent to the recipient if the recipient is offline. When you want to check the email , you have to connect the POP3 server for retrieving your message from the server which has stored in the POP3 Server to your local computer , then you can use MUA for reads the message on your PC .The message you retrieve to your PC are removed from the server.

When it comes to need when users does not want to remove the email from the server and want access yours email from another computer then it required another protocol to fulfilled your needs it is called IMAP4

12.2.2 Understanding IMAP4

The Internet Message Access Protocol Version 4 (IMAP4) provides the feature that enables you to store the message or email on network mail server. It provides most sophisticated functionality in server / client communication for handling the email. It provides

more features compares to POP3 features. With an IMAP the user emails re resides on a remote server permanently and you can retrieve the email whenever and wherever you want. You MUA must support or understand IMAP4 for retrieving the messages from the IMAP4 server.

12.3 CONFIGURING SENDMAIL

There are numerous mail transport agents are available in Linux system, in which Send mail is most widely used. Sendmail is Default MTA in Red Hat Linux. The main aim of the Sendmail is to safely transfer email among several hosts. Many systems administrators choose the sendmail as their MTA because of its power and Scalability. It is highly configurable.

12.3.1 Checking that Sendmail is installed and running

Before configuring Sendmail in your Linux system, It has to verify whether the sendmail is installed in you system or not, for checking the sendmail is installed in the system , type the following command on your Linux systems Terminal.

rpm –q sendmail

Sendmail -8.14.3-5.fc11.i586

Make sure that sendmail start when your systems boot. You have various way for checking the sendmail is running in your sytem or not. Use the one of the command is chkconfig is use to verify whether the sendmail start at the boot time

chkconfig --list sendmail

sendmail 0:off 1:off 2:on 3:on 4:on 5:on 6:off

The output shows whether sendmail is turned on or off for each of the run levels from 1 through 6

If sendmail is configured to run automatically, you should see it set to on for at least the run levels 3 and 5. If it's off for all run levels, type the following command to turn it on:

chkconfig --level 35 sendmail on

To check whether the sendmail is running or not. Type the following command

ps -auwx | grep sendmail

The above example usesps to look for Sendmail. Note that in the terminal field is “?” and it indicate that the sendmail is listening to port 25

root 8977 0.0 0.3 1488 472 ? S 12:16 0:00 sendmail:

Accepting connections on port 25

You can also use telnet to check whether sendmail is running. You telnet to yourself (localhost) and tell telnet specifically to use port 25.

To start sendmail after making changes to configuration files, type the following command:

service sendmail start

To can also use telnet to localhost 25

telnetlocalhost 25

Trying 127.0.0.1...

Connected to localhost.localdomain (127.0.0.1).

Escape character is '^]'.

220 fbreveal.com ESMTP Sendmail 8.13.8/8.13.8; Tue, 22 Oct 2013 05:05:59 -0400

quit

221 2.0.0 fbreveal.com closing connection

Connection closed by foreign host.

You can see the above output if sendmail is ruining. Type the quit command exit the session. This output is indicate that the sendmail is running and responding to incoming SMTP session

Once you have installed the sendmail in you system, you can configure it, you need to edit /etc/mail/sendmail.cf file (sendmail configuration file) for configuring sendmail. In that configuration file you have to make less changes. First of all find the Uppercase Letter DS in the configuration file as shown below

“Smart” relay host (may be null)

DS

And change the line to add the name of the mail relay host in our example specify the host name mailmessage.tactoexample.com as mail relay host. The mail relay host is the computer name that sends and receive mail on your net. Do not leave the space between DS and the hostname.

“Smart” relay host (may be null)

DS mailmessage.tactoexample.com

12.3.2 Understanding and managing the mail queue

Mail Queue

An Email message cannot delivered immediately due to various reason such as network connection is down. The recipient computer is unavailable .There can be many reason behind .The user can continue to compose email with their MUAs.When they send an message, the sendmail puts the message into the mail queue and keep trying to resend the message after some interval which is set by sendmail daemon. You can find the interval by checking the initialization script that start sendmail.

The following brief indication is from the file /etc/ rc.d/ rc2.d/ S80 sendmail

```
QUEUE=1h
fi
# Check that networking is up.
[ ${NETWORKING} = “no” ] && exit 0
[ -f /usr/sbin/sendmail ] || exit 0
RETVAL=0
start() {
# Start daemons.
echo -n “Starting sendmail: “
/usr/bin/newaliases> /dev/null 2>&1
for i in virtusertable access domaintablemailertable ; do
if [ -f /etc/mail/$i ] ; then
makemap hash /etc/mail/$i< /etc/mail/$i
fi
done
daemon /usr/sbin/sendmail $([ “$DAEMON” = yes ] && echo -bd) \
$([ -n “$QUEUE” ] && echo -q$QUEUE)
```

Here the line is begin with QUEUE define the retry interval as one hour. You can also set the interval in hours (h) , minutes (m) or seconds (s)

12.3.3 Configuring POP3

The steps involved in setting up POP3 include:

1. Installing the package that contains the POP3 daemon

2. Editing the file /etc/inetd.conf to make POP3 services available

3. Restarting the inetd daemon to make the changes in step 2 take effect

4. Checking that the POP3 daemon is accepting connections

Configuring IMAP4

To configure IMAP4, you follow the same basic steps as with POP3:

1. Installing the package that contains the IMAP4 daemon.

2. Editing the file /etc/inetd.conf to make IMAP4 services available. This

step is usually done when you install Linux.

3. Restarting the inetd daemon to make the changes in step 2 take effect.

4. Checking that the IMAP4 daemon is accepting connections. You can telnet to your own computer on port 143, as shown next, to see whether IMAP4 is accepting connections:

```
# telnetlocalhost 143
Trying 127.0.0.1...
telnet:Connected to localhost.
Esc character is '^'.
* OK localhost IMAP4rev1 v11.240 server ready
```

Setting up Mail aliases

The feature of Mail aliases is uses to create distribution list and making access to users more conveniently. For example, you can setup alias if you have trouble spelling someone's name; the mail still reaches the intended recipient if you misspell name. You can also alias a non-existent user to real user. The aliases file is usually /etc/aliases, as below.

```
# Basic system aliases -- these MUST be present.
mailer-daemon:                postmaster
postmaster:                   root
# General redirections for pseudo accounts.

daemon:                       root
lp:                           root
sync:                         root
shutdown:                     root
usenet:                       news
ftpadm:                       ftp
ftpadmin:                     ftp
ftp-adm:                      ftp
ftp-admin:                    ftp

# trap decode to catch security attacks
```

```
decode:                                 root

# Person who should get root's mail
root:                                    marry

#Users
bob:                                    alice
```

1. The following example contains entries for System aliases for mailer-daemon and postmaster, which are required.

2. Redirections for pseudo accounts such aslp, shutdown, and daemon. Most of these are all aliased to root by default, but you can change them.

3. User aliases, such as alice.

To create an entry in the aliases file, use your favorite editor. Each entry consists of the username, a colon, space(s) or tab(s), and the alias. After you save the file, you must run the newaliases command to make the changes take effect. This step is necessary because Sendmail looks at the binary file /etc/mail/aliases.db to read alias information. The newaliases command reads your aliases text file and updates the binary file.

Using other files and commands with Sendmail

Sendmail uses the following files and directories

/usr/sbin/sendmail : It issendmail daemon executable image

mailq or sendmail -bp — It Shows the contents of the mail queue:

/var/spool/mqueue — it is the directory that holds the mail queue

/var/spool/mail — it is the directory that holds a user's mail spool, for example:

ls /var/spool/mail/*

-rw-rw---- 1 alice mail 0 Jun 21 23:53 /var/spool/mail/alice

-rw-rw---- 1 bob mail 554 Mar 14 21:48 /var/spool/mail/bob

-rw------- 1 root root 6416 Jan 26 04:02 /var/spool/mail/root

/etc/mail/access —it list of usernames ,addresses , domain names and ip address that addresses not permitted to send mail to your system

/etc/mail/relay-domains — it list of hosts that are permitted to relay e-mail through your system

/etc/mail/local-host-names — it list other names for your system

/etc/mail/virtusertable — it Maps e-mail addresses to usernames on the system

Configuring the E-Mail Client

If you are system administrator, then you need to configure an e-mail client before you can sent and receive an e-mail. The configuration of MUAs depend upon the preferences and which user interface are available on your computer. If you are using Red hat Linux system there is no GUI Interface, only text based interface is available for email- configuration. In these section we will cover how to configure GUI MUA (Netscape Messenger) and three most popular text based MUAs (mail, elm and Pine).

Configuring Netscape Messenger

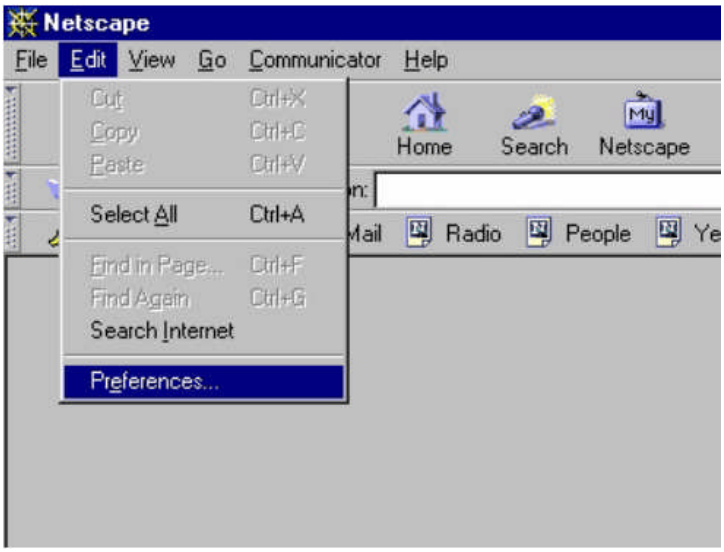
Most Linux user are familiar with the Netscape Navigator, as its default web browser comes along with Red Hat Linux system. Netscape Navigator has one of the program called as Netscape Communicator suite. In the Netscape Communicator suite there is one program called as Netscape Messenger e-mail client. If you want to check whether the Netscape package is installed in your system. Type the following command.

```
# rpm –qa | grepnetscape
netscape-communicator-4.75-2
netscape-common-4.75-2
netscape-navigator-4.75-2
```

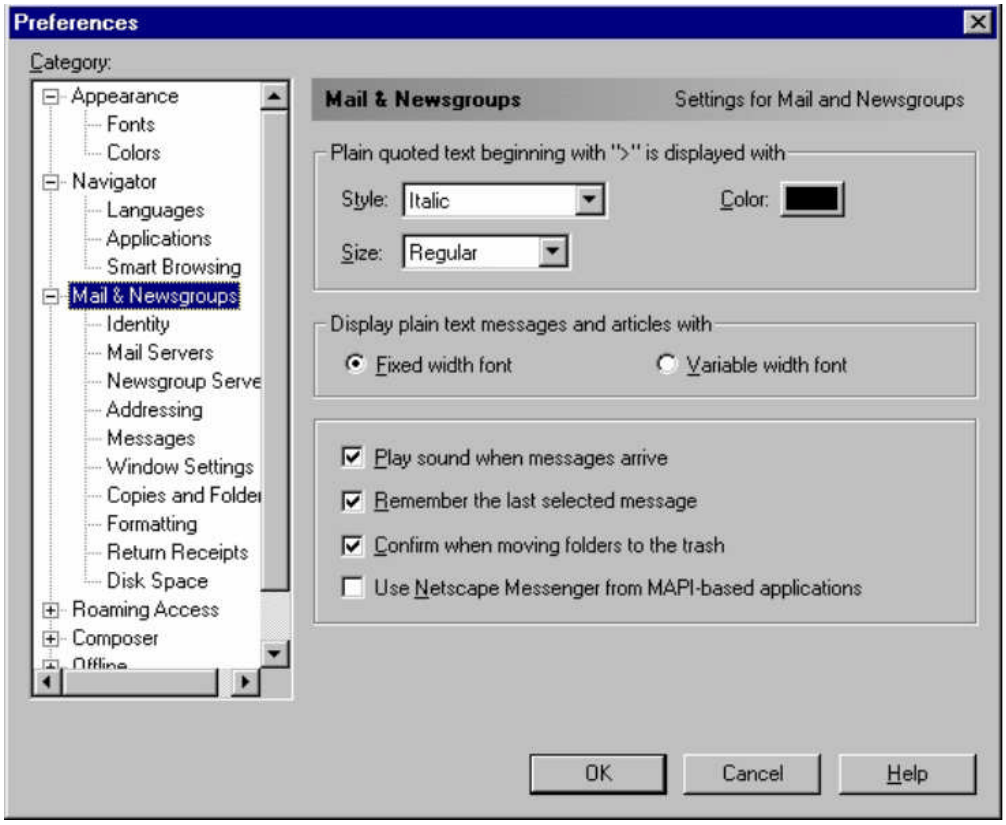
Netscape messenger has several advantage such as easy to use, east to configure and it is well integrate with Netscape navigator. The main disadvantage of the Netscape messenger is somewhat run slow.

Configuration Steps

1. Start Netscape Navigator, when the Netscape Navigator main screen appears, click onthe Edit menu item on the top menu bar (or press ALT-E), then choose/click on the suboption called Preferences.



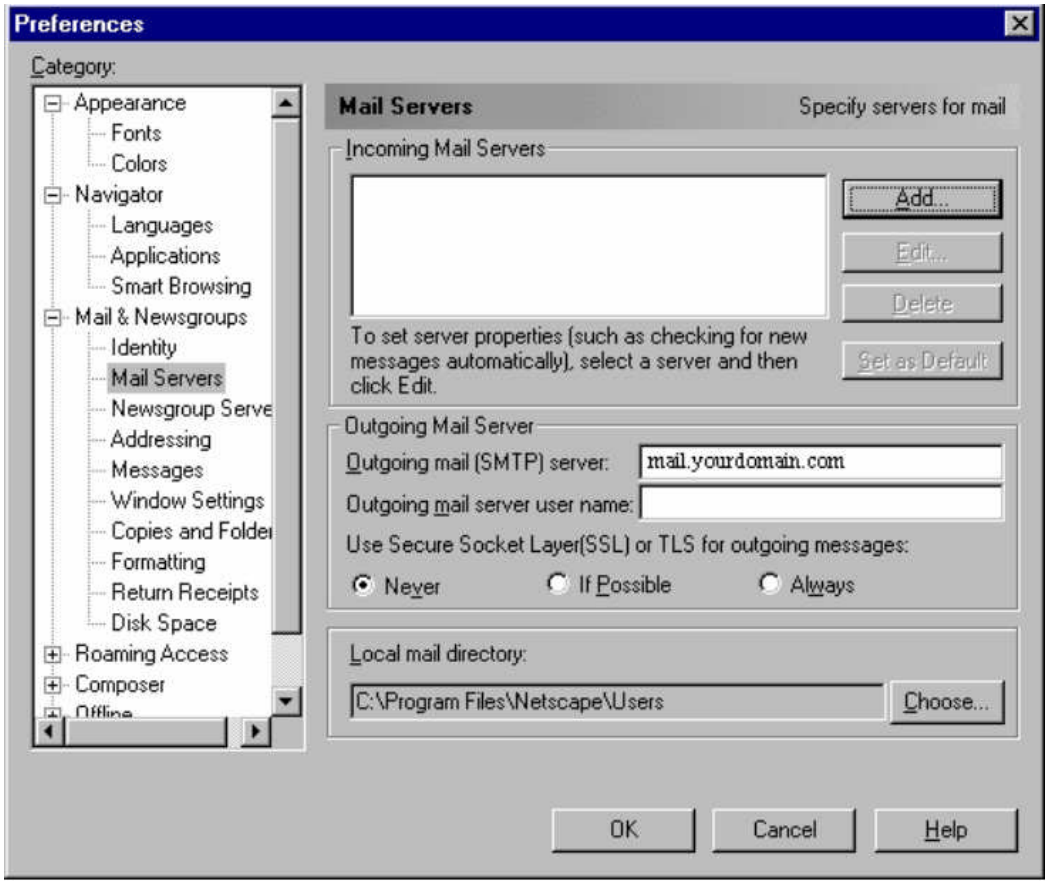
2. When the Preferences screen appears, click on the plus symbol next to the Mail & Newsgroups item listed on the left.



3. On the Mail Servers screen, enter the Outgoing mail (SMTP) server name:

mail.yourdomain.com.
Please DO NOT enter anything in the Outgoing mail server user

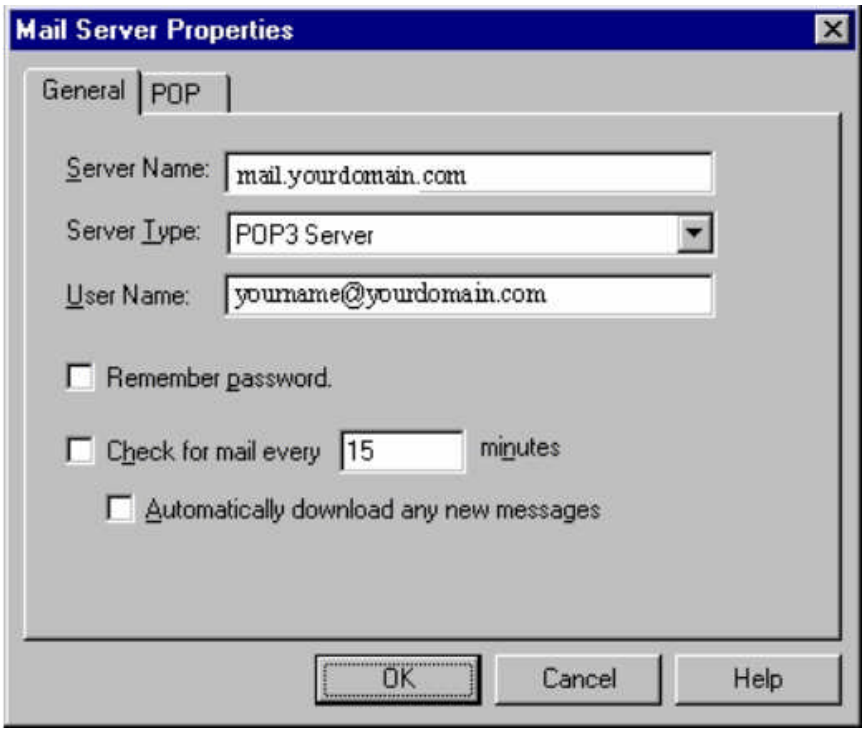
name box. Otherwise you will keep receiving "password incorrect" error messages.



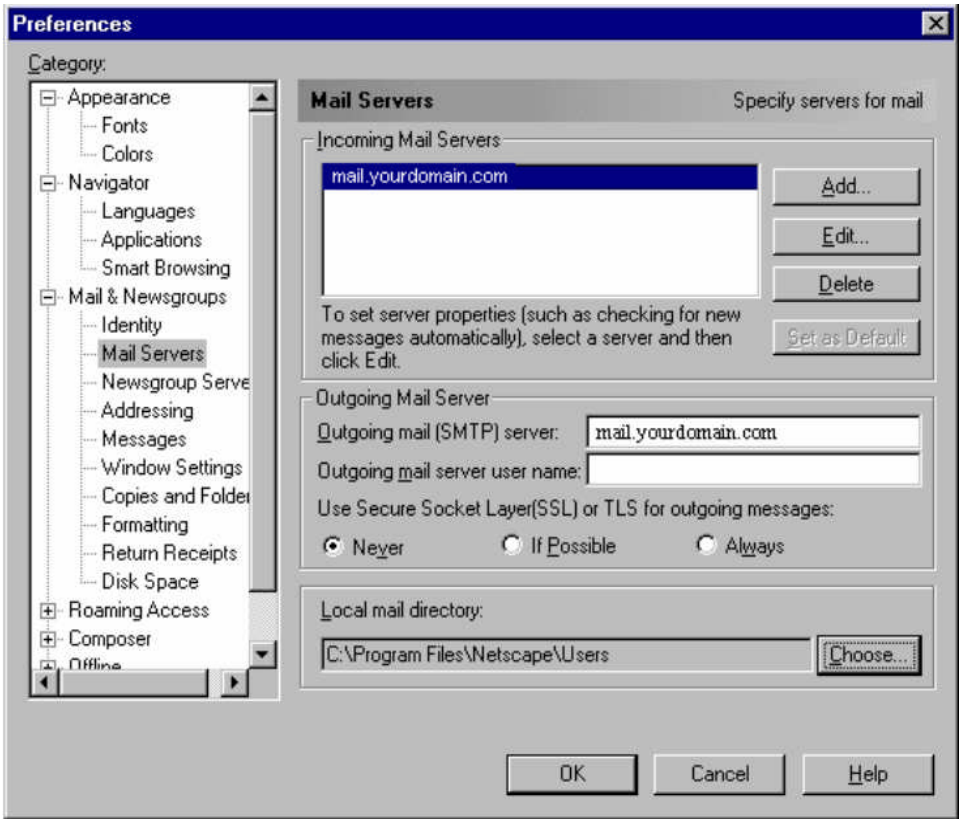
4. Next, still on the Mail Servers screen, click on the Add button to add your Incoming server name and parameters. This will bring up the Mail Server Properties screen for the account. Here you will enter in information regarding the server where you will get your mail from.

Make sure that the Server Type is set to POP3 on this screen, then enter the Server Name for your domain (this is your Incoming Server name). The example Server Name used is mail.yourdomain.com. Click the OK button when finished.

Enter the User Name as shown below, but if it doesn't work make your User Name like this: name%domain.com



5. Your settings should now look similar to the information below.



6. After verifying the above information is correct, click the OK button on all screens to return to the main Netscape Navigator screen. You may need to close Netscape Navigator and then start it again.

Recent security enhancements require authentication when sending mail. Under your outgoing mail server settings choose to authenticate when sending email. If asked, the method of authentication is MD5 Challenge-Response.

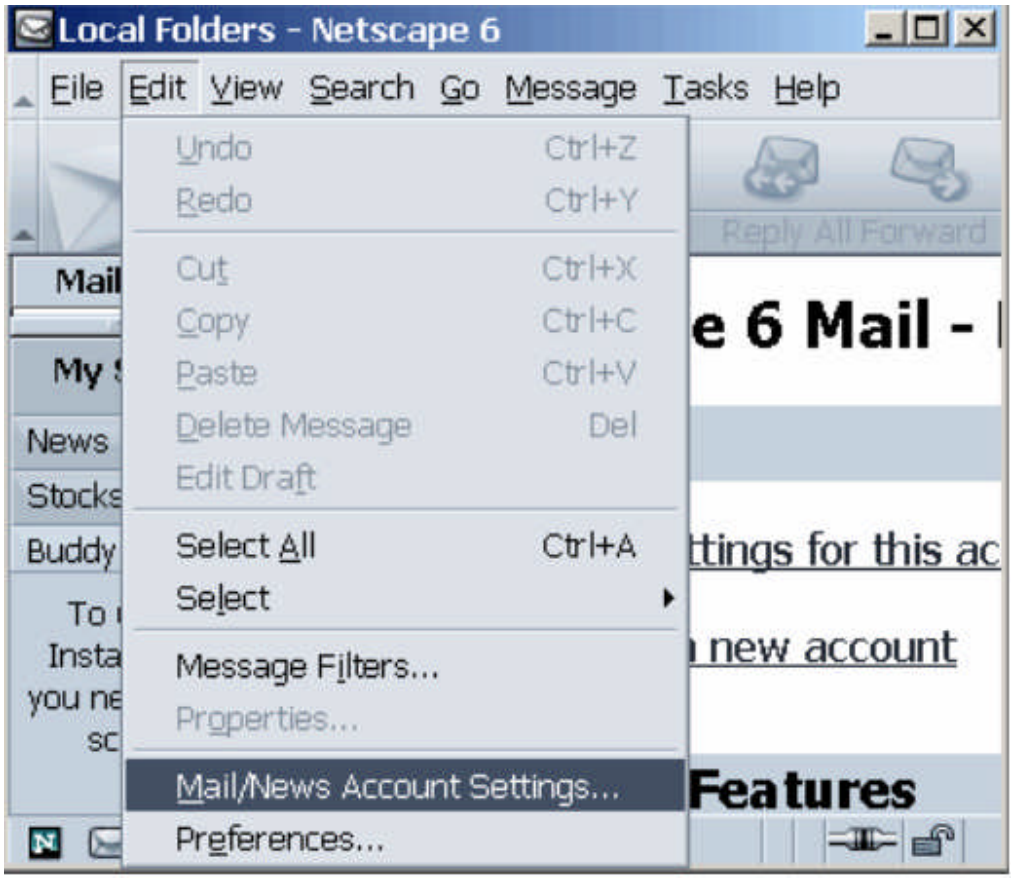
Configuration is complete!

B. Netscape Messenger 6.1

Configuration Steps

Open Netscape Mail from Start/Programs/Netscape 6/Mail. If you already have NetscapeNavigator main screen open, click on the Tasks menu item on the top menu bar (or pressALT-T), then choose/click on the sub-options called Mail.

1. In the Mail Window, open the “Edit" menu and choose “Mail/News Account Settings.”

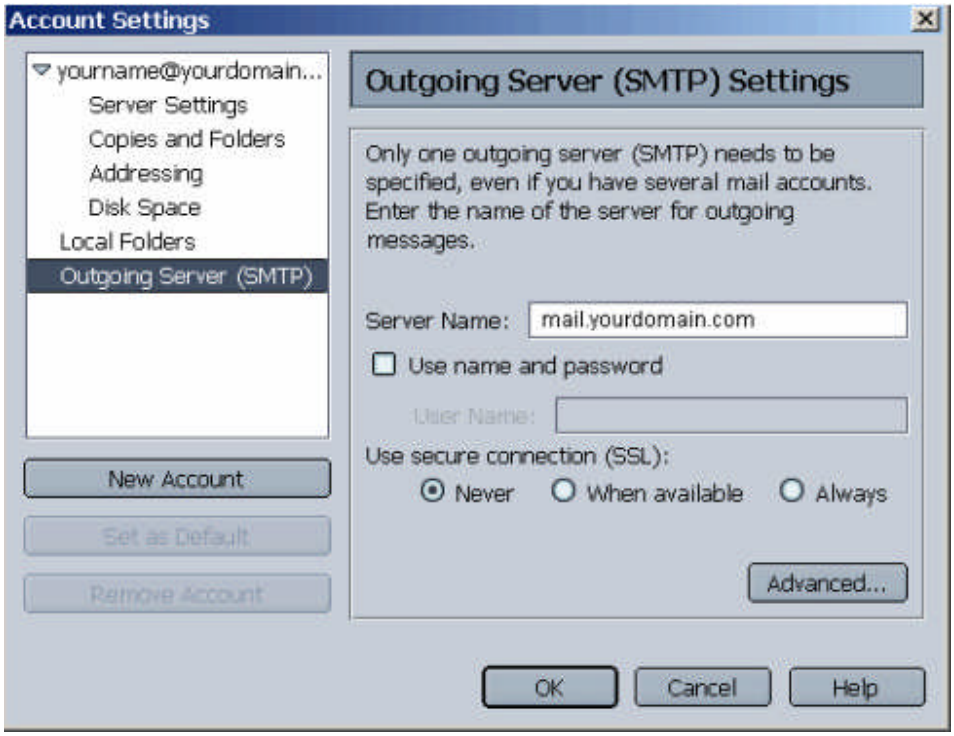


2. Then on the Account Settings dialog box, click choose "Outgoing Server (SMTP)" on the left window, and enter your mail server name, the example domain used is

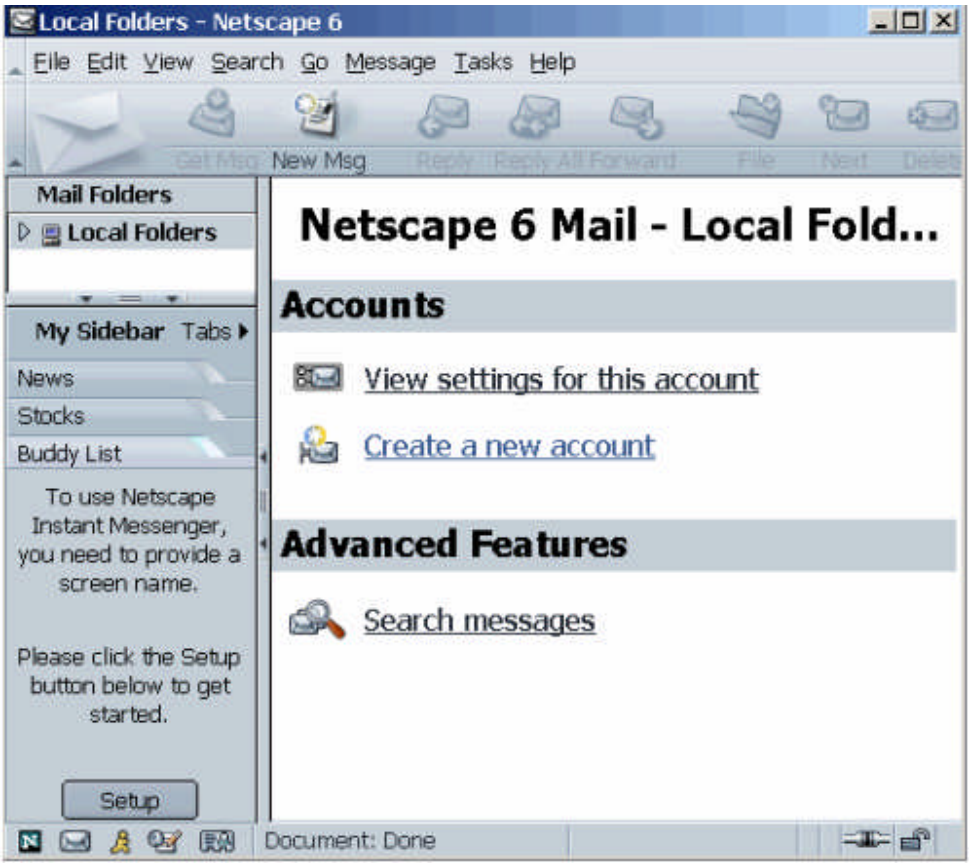
mail.yourdomain.com in the Server Name box on the right.

Please do not check the "User name and password" box. Under "Use secure connection(SSL), choose "Never." Then, click OK.

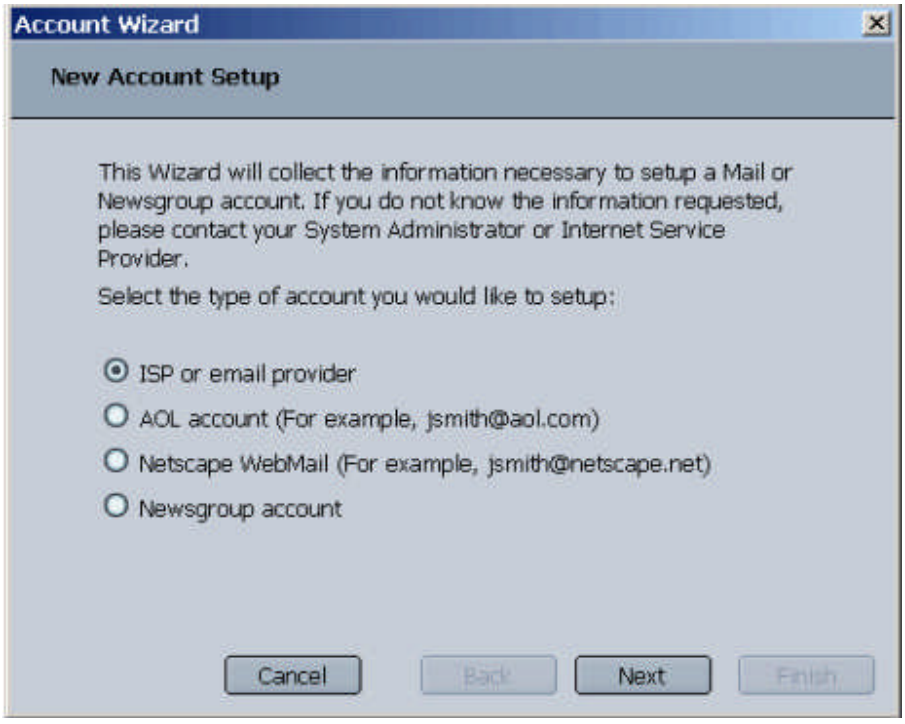
(You can click "New Account" and finish the Wizard first, and then come back and edit the Outgoing Server Settings.)



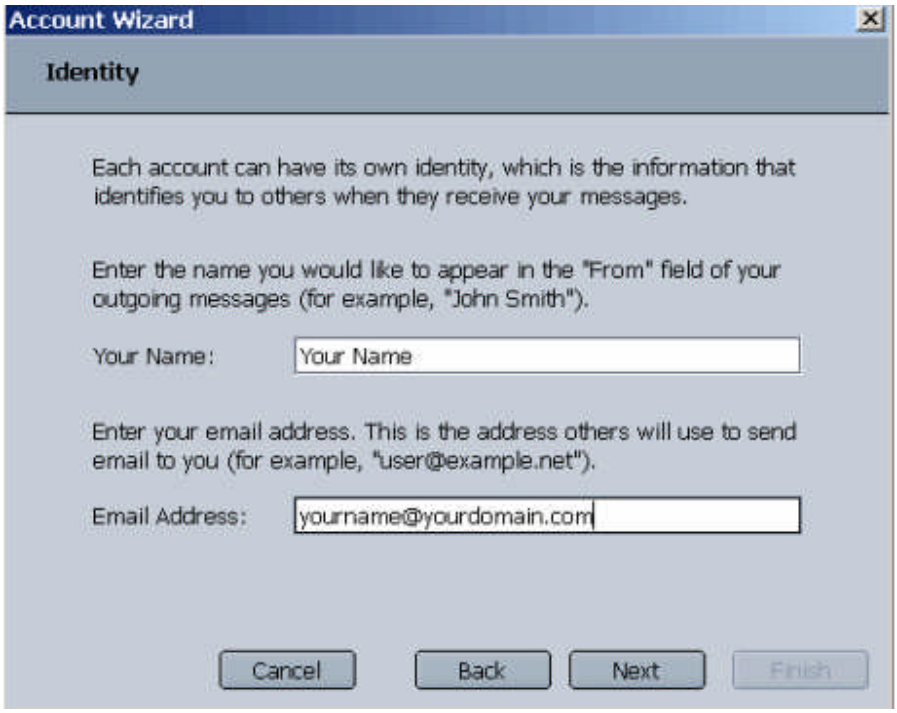
4. When you are back to the Local Folders window, click on the "Create a new account" link under Accounts to activate the New Account wizard.



5. Choose the type of account you want to set up, and click Next.

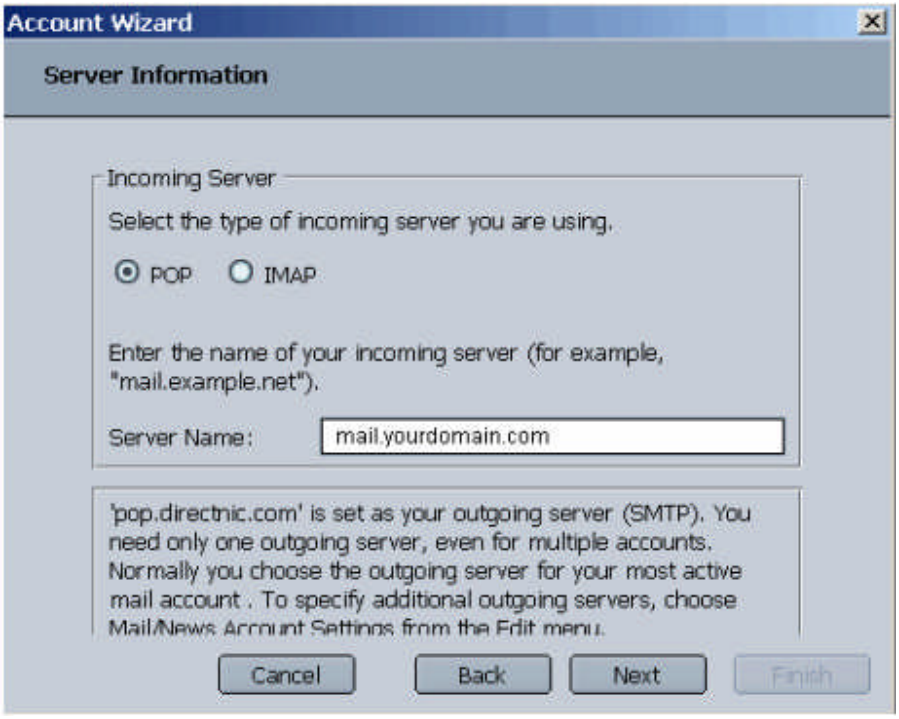


6. In the Identity section, enter your name (as you would like it to appear in the "From" fieldof messages you send) and email address (yourname@yourdomain.com), and click Next.

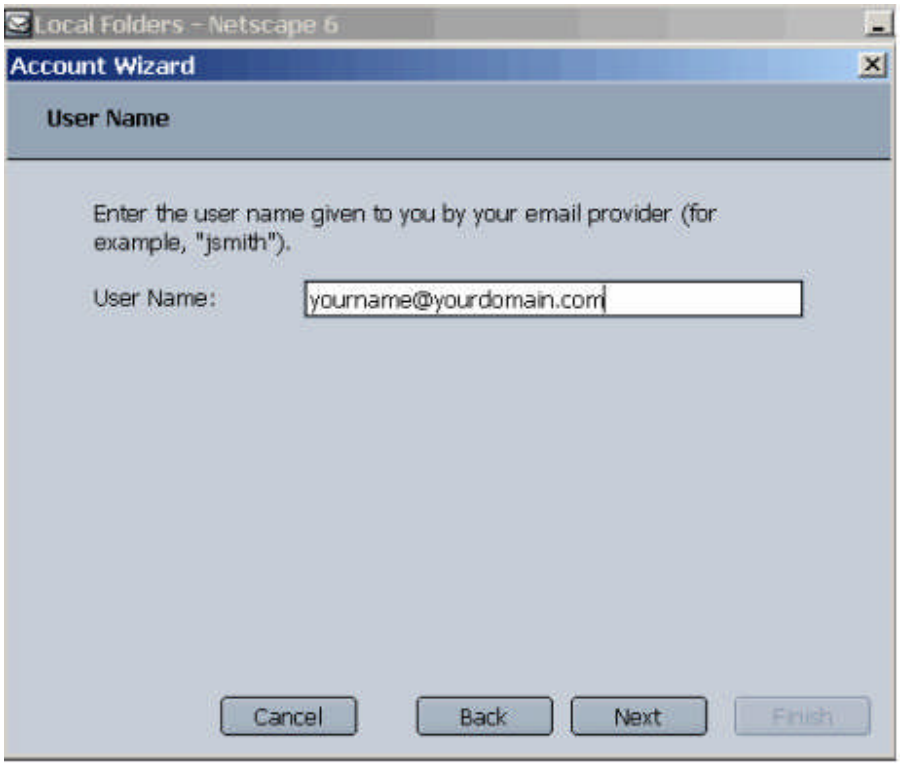


7. In the Server Information section, select the type of incoming mail server POP3. Enter the incoming server name and the outgoing (SMTP) server name. The example domain used is mail.yourdomain.com. Then click “Next.”

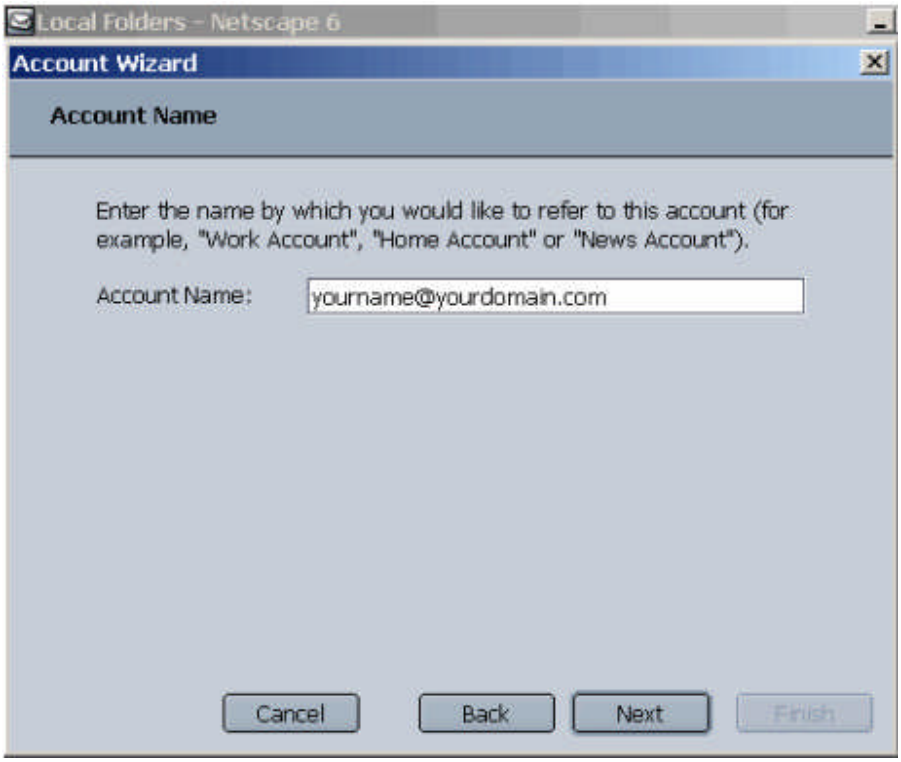
Note: Only one outgoing mail server (SMTP) needs to be specified, even if you haveseveral mail accounts. If you have not configured the SMTP settings, then you should go back to steps 2 & 3 when you finish the wizard.



8. In the User Name section, enter your full email address (yourname@yourdomain.com) and click “Next.” If it doesn't work make your account name like this: yourname%yourdomain.com

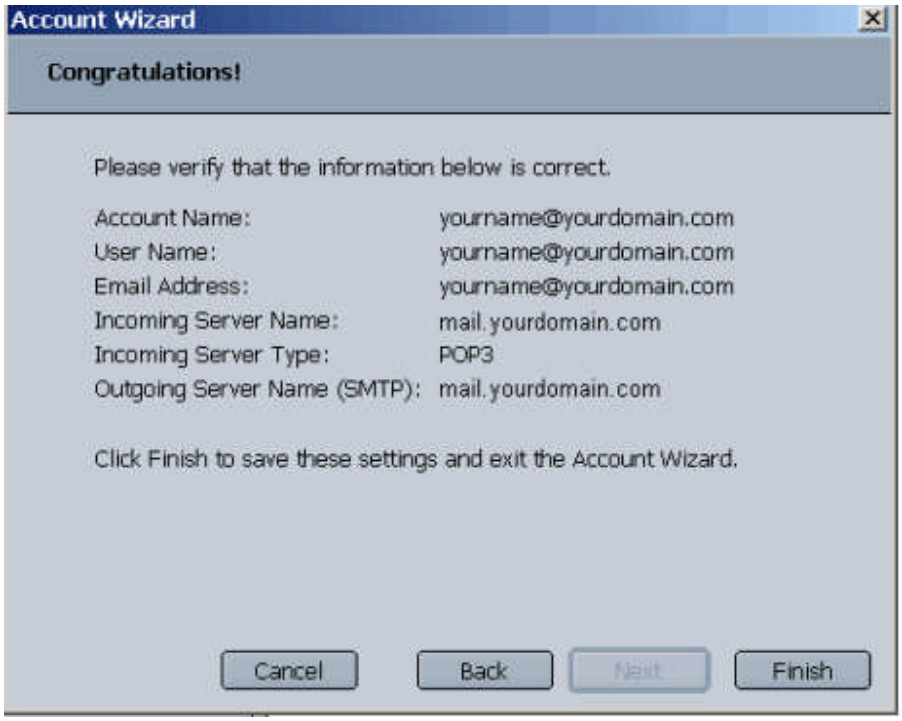


9. In the Account Name section, assign a name for this account (for example, "Work" or "Family" or simply your email address), and click “Next.



10. Verify that the information you entered is correct. If necessary, verify the information you entered with your ISP or system administrator. Then click “Finish” to set up your account.

Recent security enhancements require authentication when sending mail. Under your outgoing mail server settings choose to authenticate when sending email. If asked, the method of authentication is MD5 Challenge-Response



Configuration is complete!

❖❖❖❖

13

SENDING EMAIL THROUGH COMMAND LINE

Unit Structure

13.1. SENDING E-MAIL FROM THE COMMAND LINE

If the user does not use the GUI –style desktop like GNOME or KDE , the user only use the text based client email or console e-mail client . Text-based e-mail clients perform an interesting balancing act between efficiency and user-friendliness. Older e-mail clients, such as mail and elm, were designed when computers ran more slowly. Their lack of a graphical interface let them send and retrieve e-mail quickly.

13.2 READING MAIL WITH MAIL

Mail is oldest primitive email client .It have many advantage when installed in Linux system. It is lightweight email program which doesrequire much processing power, easy to handle , can able to run on old system which does not support GUI based email client. Infact Mail work very well in shell scripts.

Sending email by using Mail program. It has simple steps

1. At the command line , Type the Mail command followed with recipient email address

   mail<*destination@recipient.com*>
2. Press the Enter key. Mail responds with Subject: prompt.
3. Type in the subject section of email and press Enter Key
4. Now you can compose your email by using regular letter.
5. After you finished your letter , then press Enter Key
6. Type a period after the last line of the letter by itself and the press the Enter Key. This step indicate to mail program that you are done composing of email message. Mail display the EOT (End of Text) letter and you can send the message to the recipient address.

   Example Show below

   $ maillocalhost
   Subject: Type the Subject Here
   Dear User,
   I'm showing you how to use the mail e-mail client. When you finish the message, press Enter, and then type a period on a line by itself to indicate the end of the message.EOT

To retrieve the mail from the mail program, you have to perform the following steps

1. Type the Mail command on terminal without any arguments. Mail list informed the use about itself, the location of storage email and the email that you have received .It also provide the & ampersand sign, where you can enter command to read, write or delete the emails.
2. Type any command on the terminal,the & ampersand sign will prompt and press the Enter key.

   Type? Question Mark sign mail program will show a short help screen listing the different type of Keystroke using in the mail program

Command	Action
+	Move to the next e-mail
-	Move back to the previous e-mail.
?	Prints a summary of commands that you can use at the mail prompt. This is a handy reminder

R	Is used to reply to message — e.g. if you want to tell Fred 'Get well soon' in my example above, then you could do r 2..
d	Mark message(s) for deletion. d with no number marks the current message for deletion. d with a number (or +, -, $, etc.) will mark the specified message(s) for deletion. To delete messages 1 to 3, you could do d 1-3, or d 1 2 3 (or d *, in this example where there are only 3 messages). 'mark for deletion' instead of 'delete' because the changes you make are only saved when you type q.
h	Shows you a screenful of message headers (a "header" being the number, sender, date, size and subject). h with no message number shows the current screenful of messages (the number that make up a screenful is set with the screen variable, described below). h$ shows you the last screenful of messages — which is usually what you're interested in (this is usually the first thing I type when I start mail). h1 or h^ shows you the first screenful of messages.
n	Go to the next e-mail and list it.
q or x	q quits and saves your changes; x quits without saving your changes. If you quit with an x, then any messages you d'ed will not actually be deleted the next time you invoke mail — also the little N for unread will still be there even if you read the message, etc. It's as if you were never there. This is sometimes a quick and dirty way to recover if you've deleted the wrong message, etc. — or if you aren't sure what you've done and just want to bail out.
r	Reply to the sender and all the e-mail’s original recipients.
t	List the current message.
x	Quit, and don’t save e-mails.

S	is used to save a message to a file. For example, if you want to save the 'Reminder' message above as 'reminder' in your current directory, you could use 's reminder'. Or 's /tmp/foo'> to save as /tmp/foo, etc. If the specified file already exists, the message you save will be appended. So, this makes a mail folder. I find folders invaluable. I have a directory called /home/kerl/mail; in it are about a dozen folders. This helps me organize things that I want to save, but that I don't want to have to see cluttering up my in-box every time I read my mail. I can read a folder with mail -f (folder name) at the csh prompt, or f (folder name) at the mail prompt. A common sequence of keystrokes for me is: Read a message; type's /home/kerl/mail/eng' (or whatever folder); then 'd.'. There is no command to *move* a message to a folder, as there is in the graphical mail tool. Rather, you copy a message to a folder, then delete from you in-box, as two separate steps.
!	An be used to get a shell — but ^Z is just as useful, if not more so, since ^Z gets you back to the shell you invoked mail from, not a new one.
=	Just prints the current message number.
v	Puts the incoming message into vi; this may be a nicer way to read your mail than the PAGER
z or z-	If there is more than a screenful of messages, then z will show the next screenful, and z- will show the previous screenful.
$	The last message.
/(string)	All messages with (string) in the subject line (case ignored).
n-m	An inclusive range of message numbers.
(username)	All messages from (username).
^	The first undeleted message.

13.3 Commands while composing a message

The commands used while composing a message (say you used m or r to get a message going) begin with a ~ at the beginning of a line. This is so that q doesn't always mean quit, etc.

The most useful ones are as follows:

Command	Action
~m <numlist>	Append the contents of message numbers <numlist> indented by the value of the indent prefix variable (see below), or TAB by default. This is nice when you're replying to someone, so that they know which message of theirs you're replying *to* — their own words are right there.
~f <numlist>	Like ~m, but don't indent.
~p	Display the contents of the message so far. Just shows it to you — doesn't affect the body of the message that goes out.
~r <filename>	Read in the contents of <filename>
~s <subject>	Set the Subject: line to the string specified by <subject> Comes in handy if you forgot, or changed your mind.
~t <addrlist>	Set user addresses in <addrlist> to the To: list.
~h	Set the lists for To:, Subject:, Cc: and Bcc:, all at once.
~v	Edit the message using an external editor (default is vi). This is great when you're sending anything of any length at all — it lets you move around. When you quit out of vi, you can continue doing other things to the message (i.e. type some more, or use more ~ commands), or finish off the message with a . . Once you're in vi, you can use :r filename to include a file — perhaps a letter that you composed in vi earlier, some data or a configuration file you're sending someone, etc.
w <filename>	Write the message to <filename>. Great for keeping a personal copy of an outgoing message for reference.
~x	Quit, do not save letter. You can also use control-C twice.
~.	End of input. Like. Or control-D.
~?	Print a list of ~ commands.
~~	Quote a single tilde. Note that if you are rsh'ed to a host, rsh will 'eat' the first tilde, so you may need three of them.

13.4 USING ELM

Elm is text based email client founded on UNIX system .it is one of first email text based client user interface. It is much slower than Mail program but has many feature. It is first program that incorporate with Mail alias .It is similar to the vi editor interface so that user find it familiar and easy to use. If you want send and receive the emails by using elm .Type command elm on the command prompt where elm will display its own list of command. The list of Command as follows

Command	Action
D	Command use to delete mail.
U	Command use to undelete mail.
M	Command use to mail a message.
R	Command use to reply to an e-mail.
F	Command use to forward e-mail.
Q	Command use to quit elm.
E	Command use for an expired message
N	Command use for a new message
O	Command use for an old (i.e. not new but not read) message
C	for confidential mail
P	for a private message
A	for messages that have an action associated with them
F	for a form letter
M	for a MIME compliant message

- The third character of the status field can be a + to indicate that the message is *tagged* too.
- Continuing from left to right, the next field is the message number. For the most part you can ignore these unless you want to quickly move to a specific message (as we'll see later).
- The date associated with each message is typically the date the person actually *sent* the message.

- The next field displayed indicates whom the message is from. **Elm** will try to display the *full name* of the person who sent the message, rather than the return address or computer login. Some systems don't generate the correct headers, though, hence messages like numbers 2 and 8, where it's their return address.
- The number in parentheses is the total number of lines in the message.
- The final field is the subject of the message. Notice that messages might not have any subject, as in messages #9 and #10.
- A maximum of ten messages are displayed at one time. Further into the document we'll learn how to change pages in the folder.
- The three line menu display will always indicate the relevant commands. There are actually two possible menus that can be displayed, based on the *user level* as set from either the options screen or the .elm/elmrcfile. The alternate menu, for more advanced users, lists more options;

|=pipe, !=shell, ?=help, <n>=set current to n, /=search pattern

a) lias, C)copy, c)hange folder, d)elete, e)dit, f)orward, g)roup reply, m)ail,

n)ext, o)ptions, p)rint, r)eply, s)ave, t)ag, q)uit, u)ndelete, or e(x)it

- Finally, the @ character indicates where the cursor would be, awaiting your input.

All the functions available from the main screen:

Command	**Action**
<return> or <space>	Read current message.
|	Pipe current message or tagged messages to specified system command.
!	Shell escape.
$	Resynchronize folder.
?	Help mode - any key pressed will be explained.
+ or <right>	Display next page of subjects.
- or <left>	Display previous page of subjects.

=	Set current message to 1.
*	Set current to last message.
<number><return>	Set current message to number.
/	Search for pattern in subject/from lines.
//	Search for pattern in entire folder.
<	Scan message for calendar entries.2
>	A synonym for s - save message or messages.
a	Alias, change to alias mode.
b	Bounce - remail message (see f - forward too).
C	Copy current message or tagged messages to folder.
c	Change to another folder.
d	Delete current message.
<control>-D	Delete all messages matching specified pattern.
e	Edit current folder, resyncing upon re-entry.
f	Forward message to specified user.
g	Group reply - reply to everyone who received the current message.
h	Display message with headers.
i	Return to index screen after displaying message.
j or <down>	Set current to next message not marked deleted.
K	Set current to previous message.
k or <up>	Set current to previous message not marked deleted.
l	Limit displayed messages based on the specified criteria.
<control>-L	Rewrite screen.
m	Mail to arbitrary user(s).
n	n Read current message, then increment to next message not marked deleted.

o	Alter current system options.
p	Print current message or tagged messages.
q	Quit - maybe prompting for messages to delete, store, or keep.
Q	Quick quit - like quit but without prompting.
r	Reply to the author of current message.
t	Tag current message.
s	Save current message or tagged messages to folder.
<control>-T	Tag all messages matching specified pattern.
u	Undelete current message.
<control>-U	Undelete all messages matching specified pattern.
x Exit	Prompt if mailbox changed, don't record as read, don't save.
X Exit immediately	don't record as read, don't save.

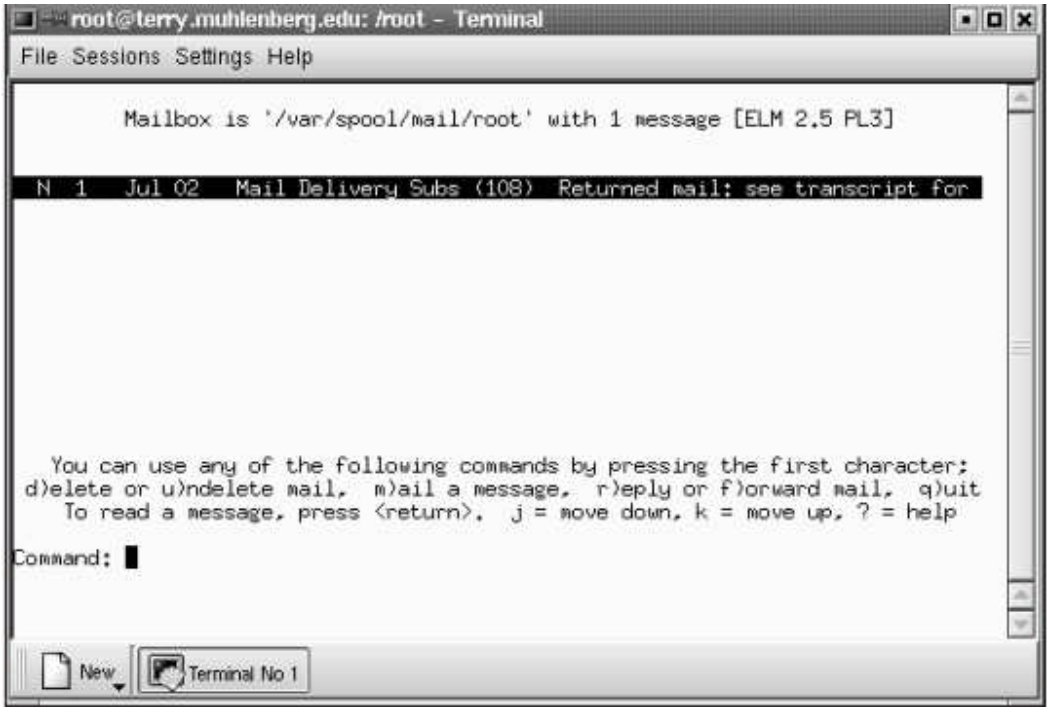


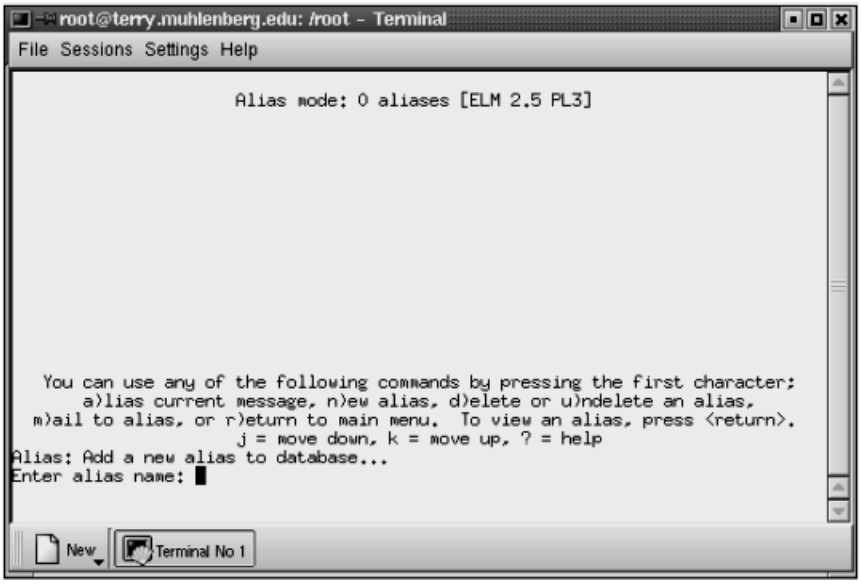
Figure 13.1 Elm Main Screen

13.5 CREATING MAIL ALIASES IN ELM

Biggest advantage of elm over the mail is you can create mail aliases. Instead of typing a long e-mail address, you can create an email aliases by using few keystroke. To create a new aliases .The following steps should be follows.

1. Type **elm** to start the elm mail program.
2. Type letter **a** to display the menu aliases.
3. Type letter **n** to create the new aliases. Elm will create a new aliases by asking for the person's first name and last name, email address and what you want the alias to be.
4. Type letter **r** to exit from the menu aliases and return to main screen of elm

Figure show of elmaliases screen



13.6 USING PINE

Pine e-mail client was developed by the University of Washington in order to provide its employee an e-email client is much easier to use and simple than elm client program. To start the Pine, Type pine at a command at the Unix system prompt.). Each Pine screen has a similar layout: the top line tells you the screen name and additional useful information, below that is the work area (on the Menu screen, the work area is a menu of options), then the message /prompt line, and finally the menu of commands. To quit: When you want to leave Pine, press Q (Quit).

The figure show a Pine main screen.

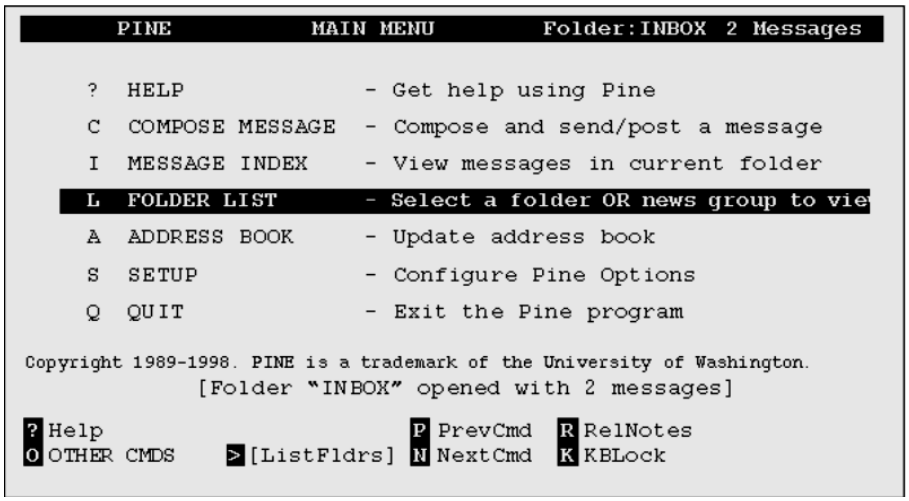


Figure 13.2 a Pine Main Menu Screen

At the bottom of the Pine main screen, you will see the list of commands, if you do not see the list of commands on the Pine screen, you want to execute the commands, then type o command, which will show the other commands you can use on the main screen.

13.7 WORKING WITH PINE ATTACHMENTS

One of the biggest advantage of using the Pine email client program is that it is first email client program in the world that handle the attachment reliably. Pine program requires the identity of file name you want to attach and rest of the process will takes care by the Pine program .To attach the file to the Pine email client program following step should be follows.

1. Type the word pine in the command line to start the Pine mail program.
2. Type letter c (for composing the mail) - this command is use to open the composition screen.
3. Use the down arrow key of the keyboard to move the cursor on the third line from the top of the pine screen, which is labelled attachment.
4. Press ctrl + J. The Pine program will ask to name of the file to be attach.
5. Enter the file name to be attached
6. Pine program prompts you for comment. You don’t have to enter one.

7. Press Enter Key and you e-mail and attachment on its way

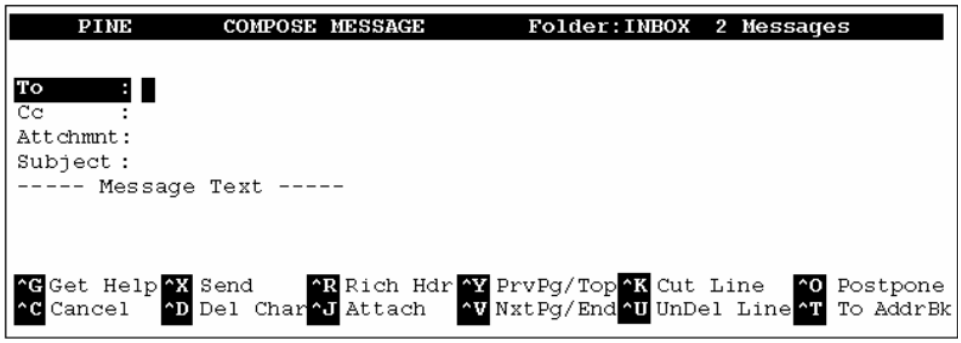


Figure 13.3 a Pine Compose Message Screen

In the command menu above, the ^ character is used to indicate the Control key. This Character means you must hold down the Control key (written in this document as) while you press the letter for each command.

Different commands are available to you when your cursor is in different fields on this screen. To see additional commands available when your cursor is in the Message Text field, type **G** (Get Help). For example, to move around, use the **arrow keys** or **N** (Next line)and **P** (Previous line); to correct typing errors, useor You might start experimenting in Pine by sending yourself a message. The following section shows you how.

Writing and Sending a Test Message to Yourself
To write and send a test message to yourself:
1. Press **C** (Compose). You see the Compose Message screen.
2. In the To field, type your email address and press.
3. In the Cc field, press <Return>.
4. In the Attachment field, press.
5. In the Subject field, type **Test** and press.
6. Below the Message Text line, type **This is a test.**
If Jean Hughes, whose userid is *jhughes*at site *art.somewhere.edu*, were to compose such a test message, the completed screen would look like the following example:

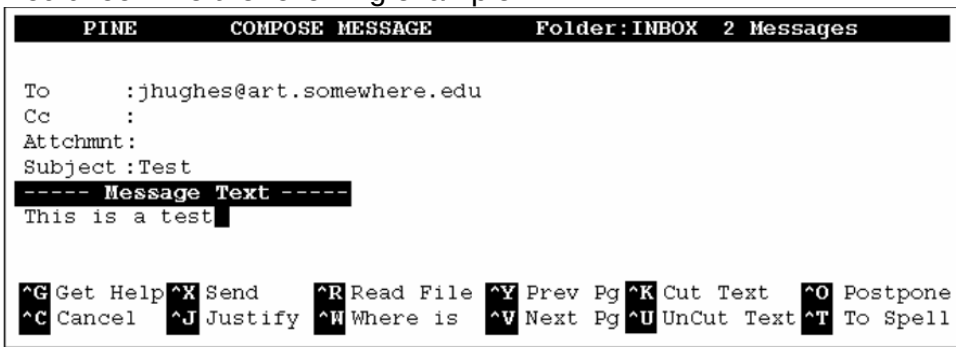


Figure 13.4 A Pine Compose Message Screen

7. To send your message, type**X** (Send).
You are asked:
Send message?

8. Press **y** (yes) or press.

The message is sent and a copy is saved to your sent-mail folder. (If you press**n** (no) the message is not sent, and you can continue to work on it.) You have just sent a basic message. There are, of course, other options you can use as you compose a message. A few are summarized in the next section, and complete information about options for the Compose Message screen is available in Pine's online help. As you compose a message, you can type **G** (Get Help) at any time to see details about your current task.

Hints for Writing a Message

To:
In this field, type the email addresses of your recipients. Separate the addresses with Commas. When you are finished, press. Always check the addresses in both the To and the Cc fields for accuracy and completeness before you send a message.

Finding and Formatting Addresses. The best way to get a person's email address is to ask him or her for it. For more information on finding and formatting email addresses on local and remote computers, type **G** (Get Help) while your cursor is in the To field.

Using the Pine Address Book. In both the To and the Cc fields, you can enter a person's email address as shown above, or you can use an entry from your Pine address book.

Cc:
In this field, type the email addresses of the persons to whom you want to send copies.
Separate their addresses with commas. When you are finished, or if you do not want to send any copies, press.

Attachment:
This is an advanced Pine feature that allows you to attach files, including word processing documents, spreadsheets, or images that exist on the *same* computer where you are running Pine. If you do not want to attach a file to your message, press for more information, place your cursor in the Attachment field, then type **G** (Get Help).

Subject:
In this field, enter a one-line description of your message. Recipients appreciate a short, pertinent description, since this is what they see when they scan their index of messages. When finished, press.

Message Text:
Type your message. To move around, use the **arrow keys**. To delete a character, press or. To delete a line, type**K**. To justify text, type **J**. (To immediately undelete a line or to unjustify text, type **U**). To check the spelling, type **T**. To see other editing commands, type **G** (Get Help).

Hints for Sending a Message
Sending a Message.
After your message is composed, type **X**, and then press **y** or press. Your message is sent and a copy is saved to the sent-mail folder. If a message cannot be delivered, it eventually is returned to you. If you want to re-send a message, you can use the **F** (Forward) command.

Changing Your Mind.
If you change your mind after typing**X** to send a message, press n instead of y to continue to work on your message. While you are writing your message, you can type **O** (Postpone) to hold your message so you can work on it later, or you can type **C** (Cancel) to delete your message entirely. You are asked to confirm whether or not you want to cancel a message.

Listing, Viewing, Replying to, and Forwarding Messages

Pine stores messages that are sent to you in your INBOX folder. Messages remain in your INBOX until you delete them or save them in other folders. (You will learn more about the INBOX and other folders in "Pine Folders".)

Listing Messages

To see a list of the messages you have received in your INBOX folder:

At the Pine Main Menu, press **I** (Message Index). The selected message is highlighted, as shown in the following example:
If you have any messages, they are listed as shown in the following example for the user named "jhughes."

If you want to list the messages in a folder other than your INBOX,

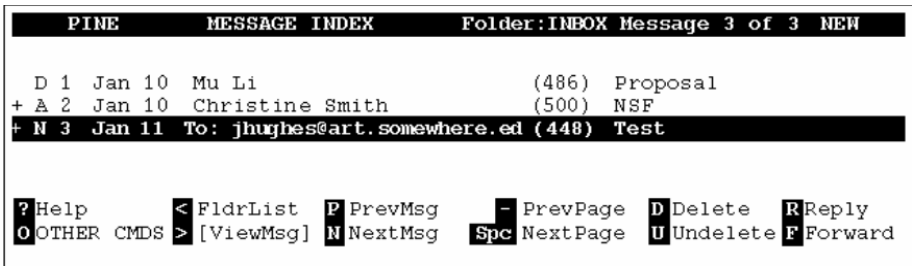


Figure 13.5 . A Pine Message Index Screen

Viewing a Message

To view a message:
1. At the Message Index screen, use the **arrow keys** to highlight the message you want to view.

2. Press **V** (ViewMsg) or press to read a selected message.

To see the next message, press **N** (NextMsg).
To see the previous message, press **P** (PrevMsg)
To return from your message to the Message Index, press **I**(Index).

Replying to a Message
To reply to a message that you have selected at the Message Index screen or that you are viewing:

Press **R** (Reply).

You are asked whether you want to include the original message in your reply. Also, if the original message was sent to more than one person, you are asked if you want to reply to all recipients. Think carefully before you answer - it may be that you want your reply to be sent only to the author of the message. Warning: It is always a good idea to check the list of addresses in the To and Cc fields before you send a message to see who will receive it.

Forwarding a Message

To forward a message that you have selected at the Message Index screen or that you are viewing:

1. Press **F** (Forward). A copy of the message opens and the To field is highlighted.

2. Enter the address of your recipient and send the message as usual. Note that you can modify the original message if you wish, for example, to forward only a portion of it or to add a message or notes of your own.

13.8 MAINTAINING E-MAIL SECURITY

Email security is the most important issue because of the possible attacks from the unknown third person that may sniff your data within the internet. To implement email security.S/MIME is the only best solution. S/MIME stands for Secure/Multipurpose Internet Mail Extensions, is the official Internet standard that specifies how messages must be formatted and exchanged between different email systems. It protects the integrity of your own and you user's email. For detail information about email security, see the sendmail website at http://www.sendmail.org/. Some common threats to email security are:

13.7.1 Protection against Eavesdropping: Eavesdropping is an unethical act of secretly tampering the private data of others without their consent. All a cracker has to do to snoop through your mail via a packet sniffer program to intercept passing mail messages. A packet sniffer program is a tool that a network administrator uses to record and analyse network traffic.

13.7.2 Using Encryption: Many email products enable your messages to be encrypted so that only you and your recipient can read them. Lotus notes provide email encryption. You can use digital signals with encryption of email to provide confidentiality to your email. Fedora core provides a full suite of digital signature and encryption services.

13.7.3 Using a firewall: You should set up a firewall to protect your network if you receive mails from people outside your network. Firewall prevents unauthorized data from reaching your network.

Prevention against bombing, spamming and spoofing: Bombing happens when someone sends the same message continually either accidentally or maliciously. Spamming is a part of bombing. A spammer sends unwanted email to many users. Spoofing happens when someone sends you email from a fake address. All three acts can cause a severe threat to your email security. You can protect your mail system from these frauds. If someone invades your mail system, you should report to the Computer Emergency Response Team (CERT).

Be careful while using SMTP: To minimize SMTP based attacks, use dedicated mail servers and have only one or a few centralized email servers. Allow only SMTP connections that come from outside your firewall to go to those few central email servers.

14

CONFIGURING FTP SERVICES

Unit Structure

14.1 Introducing and configuring vsftpd
14.2 Configuring FTP Server
14.3 Configuring User Level FTP Access
14.4 Disabling Anonymous FTP
14.5 Enabling Anonymous Uploads
14.6 Enabling Guest User FTP Accounts
14.7 Running vsftpd over SSL
14.8 Using SFTP

FTP is one of the crucial and important Internet service used by the every users in his time or another times. It is very simple task for Installation and configuration of FTP Services and it does not needs complex maintenances. Using FTP, any user can access the any files from the Internet because file is not encrypted as FTP is unsecure. The FTP carries the password as plain text file so any person can intercept traffic on the networks , while the file is being transferring over the file and cause harm to the file such as modification of content of the file , steal the data from the file etc and user are able to reconstruct the file. In Order to provides the security to the file,linuxcomes with a FTP server package, called as Very Secure FTP daemon (vsftpd).

14.1 INTRODUCING AND CONFIGURING VSFTPD

vsftpd is a very secure and extremely fast FTP server daemon run on the Linux operating system . It is stable, reliable and light weight server. Which has a project Web site at http://vsftpd.beasts.org/. According to the standard FTP services defined in RFC 959, the core Request for Comment (RFC) defines the FTP protocol, vsftpd offers the following features:

1) It Support for virtual IP configurations
2) It Support for so-called virtual users3
3) It able to run as a standalone daemon or from inetd or xinetd
4) Easily configurable
5) Bandwidth throttling
6) IPv6-ready

Installation of vsftpd

Type the following command whether vsftpd is installed in your system.

rpmquery vsftpd
vsftpd-2.0.1-5

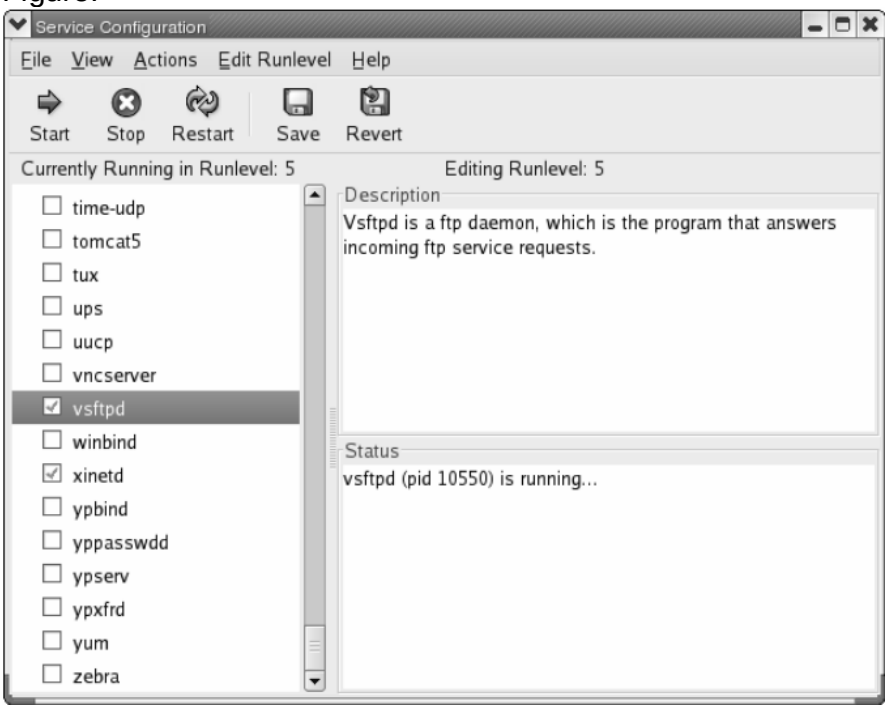
The output returns the version number of vsftpd. Ifyou see the message, "packageinstalled", install it before moving further.

rpmquery vsftpd

Package vsftpd is not installed

After installing vsftpd package configure it to start at boot time using the following command
chkconfig —levels 012B456 vsftpd off
chkconfig --levels 345 vsftpd on

You can also do the install by graphical service configuration tool. Type system-conf-services, at the terminal window to open this tool or select System -> Administration
>Services. Service configuration window appears, as shown in Figure:



In the left panel of Service Configuration window, scroll down to find vsftpd service and check the Enable button on the top menu to start vsftpd at boot time.

The vsftpd creates a basic functioning FTP server that works for users with their own laaccounts on the system and for anonymous

FTP, using either the anonymous or ftp lanames. Check for the following line to the bottom of the /etc/vsftpd/vsftpd.conf, the vsftconfiguration file:

listen=YES

Add the above line in the vsftpd configuration file, if not present. Type the following command to run vsftpd as a standalone daemon:

service vsftpd start

Now log in as an anonymous user, use a login name ftp or anonymous. Type the following bold text in the terminal window and the output returns as shown below:

$ **ftplocalhost**
Connected to localhost (127.0.0.1).
220 (vsFTPd 2.0.1)
Name (localhost:bubba): **ftp**
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>**ls -a**
227 Entering Passive Mode (127,0,0,1,100,97)
150 Here comes the directory listing.
drwxr-xr-x 3 0 0 16 Jan 22 14:17 .
drwxr-xr-x 3 0 0 16 Jan 22 14:17 ..
drwxr-xr-x 2 0 0 6 Oct 04 06:36 pub
226 Directory send OK.
ftp>**close**
221 Goodbye.
ftp>**bye**

14.2 CONFIGURING FTP SERVER

You can edit the default configuration of the vsftpd. The following files are responsible to control the server's behavior.

1) /etc/vsftpd/vsftpd.conf: It controls the operation of FTP deamon
2) /etc/vsftpd/ftpusers: It list the users that are not allowed to login via FTP
3) /etc/vsftpd/user list: It defines the user permitted access via FTP

Let's discuss configuration of user level FTP access

14.3 CONFIGURING USER LEVEL FTP ACCESS

The /etc/vsftpd/ftpusers file, contains a list of user or account names (one per line) that arenot allowed to log in using FTP. It enhances security as user accounts that are listed in this file are not permitted to log in to the system via FTP. The following code shows the default
/etc/vsftpd/ftpusers file:

```
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

In order to maintain strict limits on the FTP server access, add that user name to the end of the file from whom you want to prevent logging into the system.

The /etc/vsftpd/user_list file also limit FTP access like /etc/vsftpd/ftpusers, but it is moreFlexible. Users listed in both the files are same. Here is the main difference between these two files. The etc/vsftpd/ftpusers file unconditionally denies access to the system via FTP, but the /etc/vsftpd/user_list file deny or permit access, depending on the value of the userlist_den>directive in /etc/vsftpd/vsftpd.conf. The /etc/vsftpd/user_list file works according to these values:

1) If userlist_deny=NO, then vsftpd allows FTP access anly to the users listed in the/etc/vsftpd.user_list.
2) If userlist_deny= YES, then no user listed in /etc/vsftpd/user_list get permission to loginvia FTP.

Let's discuss configuring vsftpd features in detail.

Configuring vsftpd Features

The /etc/vsftpd/vsftpd.conf file is responsible for the important vsftpd configurations. It sets the vsftpd's behaviour. This configuration file has a defined format. Each line is either

acomment (beginning with '#') or a directive (option = value). Most of the configuration options are Boolean (on or off / YES or NO). The second group of configuration options takes numeric value and third configuration options accepts string values.

Following is the /etc/vsftpd/vsftpd.conf file content, most of the commented line has removedfrom the below file. To see complete /etc/vsftpd/vsftpd.conf file content type man /etc/vsftpd/vsftpd.conf in the terminal window.

```
anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
#anon_upload_enable=YES
#anon_mkdir_write_enable=YES
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
#chown_uploads=YES
#chown_username=whoever
#xferlog_file=/var/log/vsftpd.log
xferlog_std_format=YES
#idle_session_timeout=600
#data_connection_timeout=120
#nopriv_user=ftpsecure
#async_abor_enable=YES
#ascii_upload_enable=YES
#ascii_download_enable=YES
#ftpd_banner=Welcome to blah FTP service.
#deny_email_enable=YES
#banned_email_file=/etc/vsftpd.banned_emails
#chroot_list_enable=YES
#chroot_list_file=/etc/vsftpd.chroot_list
#ls_recurse_enable=YES
pam_service_name=vsftpd
userlist_enable=YES
listen=YES
tcp_wrappers=YES
```

The first configuration option, anonymous_enable=YES, allows anonymous FTP access. To disable anonymous FTP access set this option value to NO. The directive local_enable=YES allows local users to access the system via FTP. The write_enable=YES allows all variations of the FTP commands that allow FTP users to modify the file system such as STOR (The FTP put and mput commands for uploading files) and DELE (The FTP del command to delete files).

The directives anon_upload_enable=YES and anon_mkdir_write_enable=YES controls whether anonymous FTP users can upload files and create directories, respectively. To disable the anonymous uploads, comment out these directives. The #chown_uploads=YES directive is used to change ownership of uploaded files to the username specified by the chown_usernamedirective. Using chown_username=nobody instead of whoever is recommended as it reduce security risks.Thenobodyuser is not allowed to login via ftp.

The xferlog directives xferlog_enable, xferlog_file and xferlog_std_format control the location of the transfer log and the format of entries in these logs.Ifxferlog_enable=YES, file uploads and downloads are recorded in a log file. The default log file is /var/log/vsftpd.log. Set xferlog_std_format=NO to change the log entry format from the standard format. Add the directive log_ftp_protocol=YES to dump all FTP protocol requests into the log file.

Through dirmessage_enable=YES directive vsftpd displays the contents of a file named .message. You can change the message file using the message_file directive. The ftpd_banner directive allows you to display a site specific banner message when users connect to the servers.

Uncomment the deny_email_enable=YES, if you want to deny access to your server based on the email address provided as part of an anonymous login and put the email address you want to deny access into a file named /etc/vsftpd/banned_emails. If you want to store the banned addresses in a different file, uncomment the banned_email_file=/etc/vsftpd/banned_emails directive and change the file name.

The following two directives chroot_list_enable=YES and chroot_list_file=/etc/vsftpd/chroot_list affects the vsftpd server when it runs in chroot mode. If you uncomment chroot_list_enable=YES directive, vsftpd execute a chroot to the home directory of local users when they log in. The /etc/vsftpd/chroot_list file contains a list of the users to whom this measure is applied

14.4 DISABLING ANONYMOUS FTP

To disable anonymous FTP, this can be easily accomplished. The easiest way is to remove the ftp user from /etc/passwd and /etc/group

```
# cp -p /etc/passwd /etc/passwd.ftp
# cp -p /etc/group /etc/group.ftp
# userdel -r ftp
```

userdel: /var/ftp not owned by ftp, not removing
find / -user 50 | xargsrm -r

userdel's -r option removes files in ftp's home directory (/var/ftp), but it fail to remove the file in this case because the ftp user doesn't own /var/ftp, root does. userdel also removes the ftp user from /etc/group, so executing the groupdel command is not required. The find command locates all the files owned by the ftp user and deletes them. Use the numeric UID (50) in place of the username (ftp) because the username no longer exists.

The problem regarding to this method is that if you later decide to permit anonymous FTP, you have to recreate the ftp user and group. If the user ftp is not present in the password file, the vsftpd doesn't allow any FTP login.A more flexible approach is to add ftp to /etc/vsftpd/user_list and set userlist_deny=YES and anonymous_enable=NO in /etc/vsftpd/vsftpd.conf.

14.5 ENABLING ANONYMOUS UPLOADS

The following steps are Enabling Anonymous Uploads are :

1) Edit /etc/vsftpd/vsftp.conf
   anon_umask=077 Sets the umask used for files created by the anonymous user to 077
   anon_upload_enable=YES Permits anonymous uploads
   write_enable=YES Permits FTP write commands

2) Create a directory for anonymous uploads:
   # mkdir /var/ftp/incoming

3) Change the permissions of /var/ftp/incoming:
   # chmod 770 /var/ftp/incoming
   # chmodg+s /var/ftp/incoming

   The first chmod command give the user and group full access to /var/ftp/incoming. The second command turns on the set-group ID bit on the directory, causingany file created in this directory to have its group ownership set to the group that owns the directory.

4) Make the ftp user the group owner of /var/ftp/incoming:
   # chgrp ftp /var/ftp/incoming

5) Restart vsftpd:
   # service vsftpd restart

   These steps create a directory, /var/ftp/incoming, to which anonymous FTP users can upload files but from which no

one can retrieve files. Nor cananyone see a directory listing of the upload directory during an FTP session.

14.6 ENABLING GUEST USER FTP ACCOUNTS

A guest user is a virtual user according to vsftpd's documentation. Any non-anonymous login that uses a login name that does not exists as areal login account on the FTP server referred to as the guest user. Guest user account provides wider privileges to FTP users than anonymous FTP provides without giving local users FTP access. Guest user can access FTP server only not to the system as a whole thus provide a measure of security. You can create guest user that has the access to certain files that are not available to anonymous users. You can also create multiple guest users each with their own specific access rights.

Perform the following procedure to create a guest user account.

- Run the following command to create a guest user account:

```
#  useradd -d /var/ftp/rhlnsa3 -s /sbin/nologin authors
```

The user named authors is created in the /var/ftp/rhlnsa3 directory and a login shell of /sbin/nologin, which disables local logins for that account.

- Use the following command,to add content to this directory:

```
#  echo ‘This is a sample file. ’ > /var/ftp/rhlnsa3/sample.file
# chownauthors:authors/var/ftp/rhlnsa3/sample.fi1e
```

The first command creates simple text file /var/ftp /rhlnsa3 /sample.file.

The second command changes the user and group ownership to authors.

- Then create a text file that lists the login names and password of users that are permitted to access the authors account.

```
bubba
grits
marysue
greens
```

In this example, the users are named bubba and marysue and their passwords are grits and greens, respectively. They are stored in a file named virtusers.txt.

- Then Create a Berkeley database file from the text file named virtusers.txt.

Use the following command:

```
# db_load -T -t hash -f virtusers.txt /etc/vsftpd/vsftpd_login.db
```

This command creates a database file with the contents of virtusers .txt and stores it in /etc/vsftpd/vsftpd_login.db.

- To change the permissions on the generated file so that only root owns it:

```
# chmod 600 /etc/vsftpd/vsftpd_login.db
```

- Create a PAM (Pluggable Authentication Module) file in /etc/pam.d.These directives tell the PAM user database module to use the vsftpd_login.db file (the .db extension is assumed) for FTP logins.

auth required /lib/security/pam_userdb.so db=/etc/ vsftpd/ vsftpd_ login account required /lib/security/pam_userdb.so db=/ etc/ vsftpd/vsftpd_
login

- Edit /etc/vsftpd/vsftpd.conf

```
pam_service_name=ftp
guest_enable=YES
guest_username=authors
local_enable=YES
write_enable=NO
anon_world_readable_only=NO
chroot_local_users =YES
```

- The pam service name= ftp tells to vsftp how to use the PAM file you have created to guest _ enable=YES enables the guest user functionality.

- The guest username= authors maps all the guest logins to the authors account.

- The local enable=YES allows local users to login, which is required for guest users.

- The anon_world_readable only=NO makes the content of directory readable by the guest users

- chroot_local users— YES causes vsftpd to chroot the authors to / var/ ftp/rh1nsa3.

After making the above changes to start or restart the vsftpd service:

service vsftpd restart

Test the above configurations by using one of the user/password combinations you created above.

Let's learn to run vsftpd over SSL.

14.7 RUNNING VSFTPD OVER SSL

Vsftpd use Secure Socket Layer (SSL) support to encrypt FTP's control channel through which file transfer occurs. To use SSL with vsftpd you need to set the following configurations in the/etc/vsftpd/vsftpd.conf file.

Add the following entries to /etc/vsftpd/vsftpd.conf file:

ssl_enable=YES
allow_anon_ssl=YES
force_local_data_ssl=YES
force_local_logins_ssl=YES
ssl_tlsv1=YES

Create a self-signed RSA certificate file:
cd /usr/share/ssl/certs

Press enter and wait for the terminal to return the output. Now, start or restart the
vsftpd service:

service vsftpd start

After discussing about FTP server configurations in the previous section, let's discuss about SFTP server.

14.8 USING SFTP

With addition to running vsftpd over SSL, you can also use sftp-server. The sftp server is a part of OpenSSH suite of secure client and server programs. It implements the server side portion of the FTP protocol. The SSH daemon, sshd is used to invoke it when it receives an incoming FTP request.

Check for the required Open SSH related packages installed in your system with the following command:

```
# rpmqueryopenssh{,-{clients,askpass,server}}
openssh-4.0p1-2
openssh-clients-4.0p1-2
openssh-askpass-4.0p1-2
openssh-server-4.0p1-2
```

If these packages are not installed, install them before proceeding further. Then Check for the following line in the /etc/ssh/ sshd_config:

subsystemsftp/usr/l ibexec/openssh/sftp-server

This line tells sshd to execute the program /usr/libexec/ openssh/ sftp-server to service the SFTP subsystem.

The line to the configuration file known as /etc/ssh/sshd_config and then restart the SSH daemon using the following command:

```
# servicesshd restart
```

15

OPTIMIZATION AND SYSTEM ADMINISTRATION

Unit Structure :

15.0 OBJECTIVES

This chapter is divided into four parts. This chapter begins with offering some optimization techniques you can apply to the servers and services described in the earlier chapters. Server optimization requires analysis to narrow the problem domain, diagnosis to identify the performance problem, and experimentation to evaluate the effectiveness of your optimization.

The second part focuses on the updating of the system and upgrading the kernel. The Linux kernel has changed in a number of significant ways ever since its inception. The kernel itself is faster, more capable, and more flexible, and the build process has changed dramatically and for the better. The kernel is the core of the operating system and runs the CPU, manages system memory, controls access to disk drives, and contains device drivers that enable you to use the hardware and peripherals attached to the computer. The ability to update and customize the Linux kernel is one of the things that many people like best about Linux. Naturally, this feature is very important and enables system administrators to wring the most performance and benefit out of their existing hardware and software.

The third part discusses the finer points of user and group maintenance on Fedora Core and RHEL systems. You will learn how to add, modify, and delete user accounts and how to use Sudo to give normal users root capabilities on a limited and monitored basis.

The final part explains how users can download and install new or updated software with little or no difficulty using RPM, the Red Hat Package Manager and also the process of configuring and installing software from the source code.

15.1 INTRODUCTION

In general, anything that improves overall system performance will improve Internet services performance as a side effect. Disabling unnecessary services is a standard technique and hopefully one that you have already implemented. Centralizing Internet services on a machine that is not used by users is another general performance tweak for servers. Also, getting a fatter pipe or simply using Gigabit Ethernet (GigE) whenever possible will improve Internet server/service performance. Finally, using a higher-performance file system on the directories that hold email can substantially improve performance.

15.2 OPTIMIZING LDAP SERVICES

Several items in the slapd configuration can be tweaked to give better server performance. The items shown in the following list show configuration directives that can be modified for performance reasons:

- **Cache size modification** — You can increase the size of the cache using the cachesize directive in slapd.conf. For example, the following directive sets the number of LDAP entries stored in the cache to 100,000: *cachesize 100000*
- **Disk subsystem** — Replace IDE disks with SCSI disks, and replace or augment SCSI disks with FibreChannel. If SCSI and FibreChannel are too rich for your budget, using Serial ATA (SATA) drives (and a SATA controller, of course) hits a good middle ground because SATA is faster than IDE and less expensive than SCSI and FibreChannel. If you have multiple LDAP data stores, situate each store on its own disk and, if possible, its own dedicated I/O controller in order to minimize I/O contention with other processes.
- **Filesystem tuning** — On filesystems that support it, disable updating file access and modification timestamps, which will decrease the number of file operations that have to be performed by two-thirds. Fewer CPU cycles spent on bookkeeping means more CPU cycles spent doing actual LDAP-related work.
- **Indexing** — With limits, indexes increase performance, but at the cost of additional memory, disk, and CPU usage. Accordingly, don’t index data you don’t (often) search. By way of guidelines, index only heavily used parts of your schema.
- **Logging** — If you are persuaded that excessive message logging is hampering the performance of your LDAP server, add the following entry to slapd.conf: *loglevel 0*

This entry disables logging via the system log.

- **System memory** — In addition to adding more physical memory, increase the size of Open LDAP's cache to use more RAM.

For more details about these performance tuning tips, have a look at the Open LDAP FAQ, available on the Web at ***openldap.org/faq/data/cache/190.html***.

15.3 Optimizing DNS Services

Optimizing DNS services centers on reducing the latency involved in making DNS queries. For client programs, that is, for applications requesting DNS services, the best all around performance enhancement is to maintain a local cache of DNS information. You get the most bang for your performance buck by reducing the number of DNS queries that have to go to a remote server, even if that server is inside the subnet to which you are connected. The typical approach is to run a caching-only name server on client machines. On the server side, you have a much wider range of options.

15.3.1 Improving the performance of DNS Clients

To increase the performance (and security) of your caching-only servers on the DNS clients, several options can be modified in the */etc/named.conf* file created during the installation of BIND. The */etc/named.conf* file is shown in Figure 15-1.

The section of the file in which you are interested is the options section. Since this is a caching-only server, you can safely disable functions that are not necessary for this type of server. You can also add options that do apply to a caching-only server.

a) By default, BIND allows zone transfers to all hosts. However, zone transfers are necessary only for master and slave servers; they aren't necessary for DNS clients. You can disable zone transfers by adding the following line to /etc/named.conf:

```
allow-transfer {
none;
};
```

b) You can also configure you caching-only server to respond to regular queries only from specific hosts. BIND's default setting is to allow queries from any host. Typically, you want to allow queries only from hosts inside your firewall. You can add the following line to the options section in /etc/named.conf, replacing *www.xxx.yyy.zzz* with your network IP number:

allow-query {
www.xxx.yyy.zzz;
localhost;
};

c) You can also configure your caching-only server to respond to recursive queries from only specific hosts. BIND's default setting is to allow queries from any host. Typically, you want to allow queries only from hosts inside of your firewall. You can add the following line to the options section of /etc/named.conf, again replacing *www.xxx.yyy.zzz* with your network IP number:

allow-recursion {
www.xxx.yyy.zzz;
localhost;
};

```
// generated by named-bootconf.pl
options {
directory "/var/named";
        /*
        * If there is a firewall between you and nameservers you want
        * to talk to, you might need to uncomment the query-source
        * directive below.  Previous versions of BIND always asked
        * questions using port 53, but BIND 8.1 uses an unprivileged
        * port by default.
        */
// query-source address * port 53;
};

//
// a caching only nameserver config
//
controls {
        inet 127.0.0.1 allow { localhost; } keys { rndckey; };

};
zone "." { type hint; file "named.ca"; };
zone "localhost" {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "named.local";
allow-update { none; };
};

include "/etc/rndc.key";
```

Figure 15-1 The /etc/named.conf file

d) Typically, a caching-only server does not have direct access to the Internet, so it creates a cache file to hold DNS information.

This is the purpose of a caching-only server, and it boosts performance by eliminating the need to send queries to external servers. But if the server does not have the information it needs in its local cache, it needs to send a request to other servers. You can specify the IP address of the servers to which you want to forward requests. Add the following line to the options section of /etc/named.conf, replacing *www.xxx.yyy.zzz* with the IP address of the name server to which requests should be forwarded:

```
forwarders {
www.xxx.yyy.zzz;
};
```

e) What happens if the servers you are forwarding to are down? Your server tries to forward the request to other servers. You can prevent this from happening by adding this line:

```
forward only;
```

15.3.2. Tweaking DNS Servers

To increase the performance and security of your master domain server on the DNS clients, several options can be modified in the /etc/named.conf file created during the installation of BIND. Make changes to the options section of /etc/named.conf.

- By default, BIND allows zone transfers to all hosts. Zone transfers are necessary only between the master and slave servers, so you can specify the IP address of your slave server by adding the following entry (replace *www .xxx.yyy.zzz* with the IP address of your slave server):

```
allow-transfer {
www.xxx.yyy.zzz;
};
```

- You can also configure your master server to respond to regular queries only from specific hosts. The default setting in BIND is to allow queries from any host. Typically, you want to allow other types of queries only from hosts inside your firewall. You can add the following line to the options section:

```
allow-query {
www.xxx.yyy.zzz;
localhost;
};
```

- Replace *www.xxx.yyy.zzz* with your internal network IP number. You can also configure your caching-only server to respond to recursive queries only from specific hosts. The default setting in BIND is to allow queries from any host. Typically, you want to allow queries only from hosts inside of your firewall. You can

add the following line to the options section:

allow-recursion {

www.xxx.yyy.zzz;

localhost;

};

Again, replace *www.xxx.yyy.zzz* with your internal network IP number.

15.3.2.1 Logging

You can configure your caching-only slave and master servers to automatically rotate your /var/log/named.log file to prevent your filesystem from filling up with old information. The file /etc/logrotate.d/named should have been created during the installation of BIND and should be similar to Figure 15-2.

```
/var/log/named.log {
        missingok
        create 0644 named named
        postrotate
                /sbin/service named reload  2> /dev/null || true
        endscript
}
```

Figure 15-2 The /etc/logrotate.d/named file controls log rotation of BIND's log files.

15.4 OPTIMIZING MAIL SERVICES

To improve the speed of your mail services, you can take one of several approaches.

a) Busy sites often use multiple mail servers in order to spread the mail processing load across a number of systems. This reduces the demand on any single system.

b) Another common performance enhancement is to replace Sendmail with another mail server, such as Postfix. If your mail server supports a number of mailing lists, you might consider handling list traffic on one server and regular (nonlist) mail traffic on another server.

15.4.1 Getting More from Sendmail

The /etc/mail/sendmail.cf file contains many options that can be tweaked to give better performance and increase the efficiency of your mail server. The most common Sendmail tweak if to change the frequency with which it runs the queue. You can modify this by editing /etc/sysconfig/send mail and changing the line that reads QUEUE=1h, which causes Send mail to process the mail queue every hour, to, for example, QUEUE=15m, which runs the queue every 15 minutes. If you modify this file, you have to restart Send

mail. A good source for some performance-tuning tips can be found at the Send mail Web site at send mail. org/~ca/ email/ doc8. 12/TUNING.

15.4.2 Getting More from Postfix

If you have a lot of mail that just seems to sit in Postfix's outbound queue, you may be trying to deliver mail to a site that is quite busy. One way to work around this problem is to create a transport map entry for such a site that enables multiple parallel connections and then to give each connection to that site a shorter timeout. Next, create a corresponding entry in /etc/postfix/main.cf that increases the number of simultaneous connections, which allows more mail to be transmitted at once. The entries from /etc/postfix/master. cf tell Postfix that SMTP connections to these busy sites should timeout after 5 seconds and that, similarly, an SMTP transaction must commence within 5 seconds of the HELO command, or the connection will be closed.

If incoming mail seems to queue up while outbound mail gets delivered, then outgoing mail is crowding out incoming mail. Postfix can waste a great deal of time waiting for connections to time out, so, again, the solution is to reduce the connection timeout for incoming email.

If you see that Postfix pegs disk I/O when processing incoming mail, the *real* solution is to get faster disks or to allocate one disk for logging, one disk for the mail queue, and a third disk for user mailboxes. Postfix-caused disk saturation is especially a problem if you are serving multiple virtual hosts on a single system. One workaround is to configure multiple IP addresses for the machine and to run a Postfix instance for each IP address, where each Postfix instance writes to a different disk. It is easier to configure than it might seem at first glance. The key is starting each Postfix instance with a different configuration directory.

If Postfix responds too slowly to incoming SMTP connections but POP or IMAP connections are acceptably fast, you need to run more SMTP server processes. Edit the smtpd entry in the master.cf file and increase the process limit. Alternatively, increase the default_process_limit setting in the main.cf file.

NOTE: Anytime you edit one of Postfix's configuration files, be sure to use the *postfix reload* command to activate the changes.

15.5 OPTIMIZING FTP SERVICES

Out of the box, vsftpd is pretty darn fast and makes lightweight demands on a system's memory and CPU resources. If its speed fails to suit you, the following tips, adapted from the vsftpd documentation, might help:

a) If possible, disable the NIS and NIS+ (nis and nisplus) for passwd, shadow, and group lookups in /etc/nsswitch.conf. The idea with this tip is to avoid loading unnecessary runtime libraries into the vsftpd's memory space and to avoid using NIS for lookups that can be resolved more quickly by resorting to file-based lookups.

b) Break directories with more than a few hundred entries into smaller directories. Many file systems, such as ext2 and ext3, do not handle such cases efficiently at all, and the process of creating listings of large directories (with, for example, the ls or dir commands) causes vsftpd to use moderate amounts of memory and CPU. If you are stuck with large directories, use a file system, such as XFS, JFS, or Reiser FS, designed to work with large directory structures.

c) Limit the number of simultaneous connections to the FTP server.

d) More drastically, if the load on your FTP server is bogging down the system, you could disable anonymous FTP altogether or dedicate a machine to providing FTP services.

e) Take advantage of vsftpd's bandwidth throttling features to limit the network bandwidth consumed by any one connection or connection classes.

15.6 OPTIMIZING WEB SERVICES

Chapter 13 briefly touched on Apache configuration settings you can modify that affect Apache's performance. The settings mentioned in that section are good starting points for fine-tuning Apache, but they do not exhaust the possibilities. To further that discussion, the following tips and suggestions appear in no particular order. Your mileage may vary, and if breaks, you get to keep both pieces. Some of the following might work better than others; others ideas might fail miserably. If your server is running a lot of CGI scripts or using PHP markup, you should look into resources that discuss Apache tuning in depth. The overhead requirements of PHP and CGI scripts involve creating new processes rather than merely additional RAM, network, or disk I/O.

a) Set *Hostname Lookups* to *Off*. Each resolver call impairs performance. If you need to resolve IP addresses to hostnames, you can use Apache's log resolve program or one of the resolver programs available in the log reporting and analysis packages.

b) Similarly, use IP addresses instead of host names in *Allow from domain* and *Deny from domain* directives. Each such query, when *domain* is a name, performs a reverse DNS query followed by a forward query to make sure that the reverse query is not being spoofed. Using IP addresses avoids having to resolve names to IP numbers before performing the reverse and forward queries.

c) If you do not use *Options Follow Sym Links*, or if you *do* use *Options Sym Links If Owner Match*, Apache performs extra system calls to check symbolic links. If a client requests */index.html*, Apache performs an *lstat ()* system call on */var, /var/www, /var/www/htdocs*, and */var/www/htdocs/index.html* to check the owner matching of the symbolic link. The overhead of these *lstat()* system calls occurs for *each* request, and Apache does not cache the results of the system calls. For the best performance (and, unfortunately, the least security against rogue symlinks), set *Options Follow Sym Links* for all directories and never set *Options Sym Links If Owner Match*.

d) A similar performance problem occurs when you use *.htaccess* files to override directory settings. In this case, Apache attempts to open *.hatches* for each component of a requested filename. For the best performance use *Allow Override None* everywhere in the Web space Apache is serving.

e) Unless you rely on the *Multi View* option, turn it off. It is perhaps the single biggest performance hit you can throw at an Apache server.

f) Do not use NFS mounted file systems to store files that Apache serves unless absolutely necessary. Not only is the read performance of NFS slower than the read performance of a local file but also the file being served via NFS might disappear or change, causing NFS cache consistency problems. Moreover, if the Apache server is somehow compromised, the NFS mount will be vulnerable.

g) If you must use NFS-mounted file systems, mount them as read-only. Read-only NFS mounts are significantly faster than read/write mounts. Not only will this improve performance, disabling write access adds another barrier to bad guys who might compromise the system.

h) The single most important system resource that Apache uses is RAM. As far as Apache is concerned, more RAM is better because it improves Apache's ability to store frequently

requested pages in its cache. You can also help by limiting the non-Apache processes to the absolute minimum required to boot the system and enable Apache to run — that is, run a dedicated Web server that doesn't need to share the CPU or memory with other processes. Naturally, a faster CPU, a high-speed Ethernet connection, and SCSI disks are preferable.

15.7 UPDATING YOUR SYSTEM

The Red Hat Network up2date agent is a program that is installed by default when you install Fedora Core or Red Hat Enterprise Linux. The Red Hat Network up2date software will give you visual notification of the update right on your desktop. This might not sound like much at first, but think about the many steps involved in keeping your system up to date with the latest versions of the hundreds of packages that are installed on your system. The Red Hat Network practically eliminates the need for you to search for these packages because you can receive this information by email.

Whenever your system needs to be updated, the Alert icon will appear as a red circle containing an exclamation point. You can roll the mouse over the Alert icon to view a small pop-up window that gives additional information. If your system needs to be updated the pop-up window will show the number of updates available.

There are multiple ways to start the up2date agent, here is one way. To start the up2date agent to update your system, do the following:

1. Right-click the Alert icon and select Launch Up2date from the contextual menu. You see the Red Hat Update Agent Welcome screen.
2. Click Forward to continue to the Channels dialog box, as shown in Figure 15-3.
3. The channels dialog box lists the channels that will be searched from which the updated packages will be obtained. You can think of the channels as file repositories on various servers in many locations. Click Forward to continue. The program connects to the selected channel to search for package updates. By default the kernel packages are not automatically updated and will be listed as packages to be skipped.

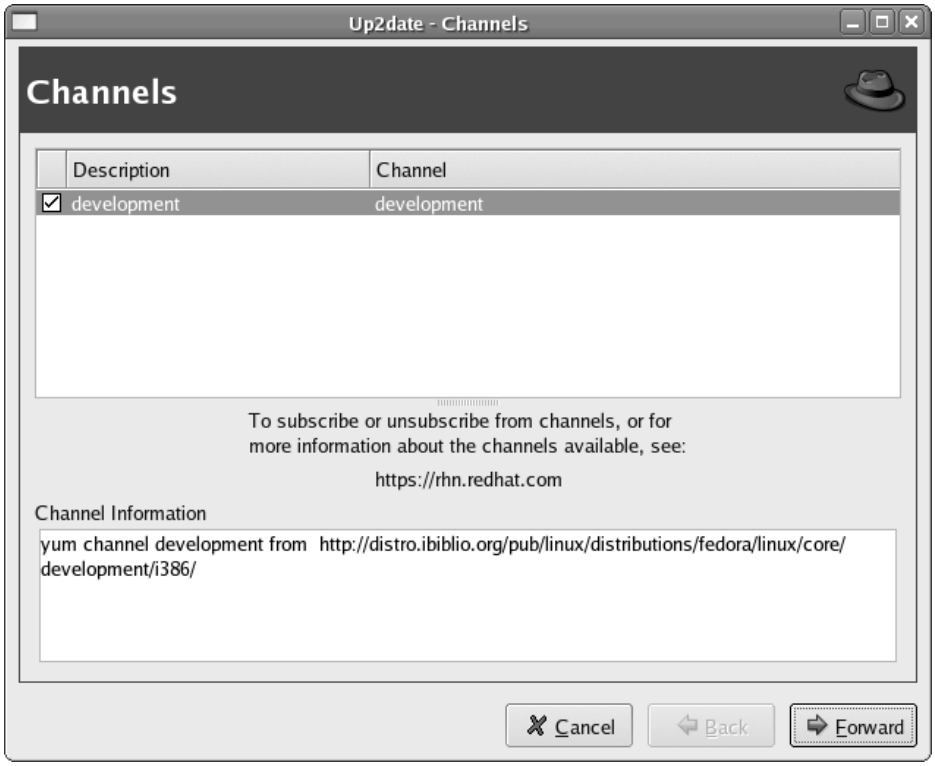


Figure 15-3 The Channels dialog box.

1. If you want to update the kernel packages check the appropriate box and then Click Forward to continue. You see the Package List dialog box with the available packages, as shown in Figure 15-4. If your system is updated, you won't see any packages listed.
2. You can select packages individually by selecting the check box in front of the package name, or you can mark the Select All Packages check box to select all packages. After you finish selecting packages, click Forward to begin package retrieval. You see the Package Retrieval dialog as shown in Figure 15-5.
3. The up2date program gets the packages and prompts you to continue after the packages have been retrieved. Click Forward to install the packages.
4. You see a progress dialog box during the package installation. After all the packages that you selected for installation are installed, you see a dialog box indicating the package installation has finished. Click Finish to complete the update process.

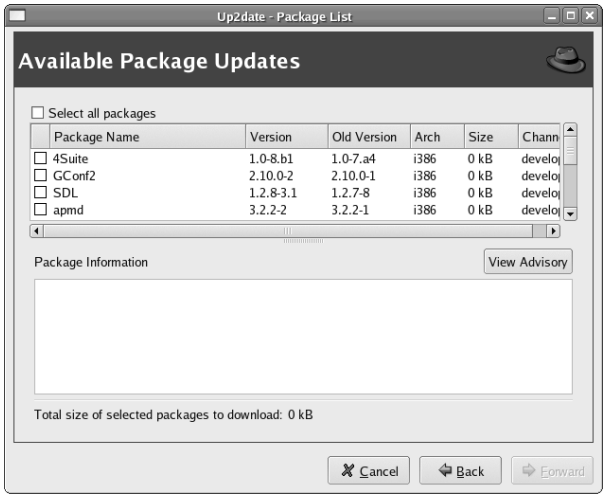


Figure 15-4 The package list dialog box

Figure 15-5 The package retrieval dialog box

15.8 UPGRADING AND CUSTOMIZING YOUR KERNEL

The kernel is the core of the operating system and runs the CPU, manages system memory, controls access to disk drives, and contains device drivers that enable you to use the hardware and peripherals attached to the computer. The ability to update and customize the Linux kernel is one of the things that many people like best about Linux. Naturally, this feature appeals most to incorrigible tweakers and tinkerers, but it also appeals to system administrators who are responsible for wringing the most performance and benefit out of their existing hardware and software. In some cases, rebuilding the kernel is required to support new hardware that is not supported, or that is poorly supported, by your system's existing kernel. **The following steps need to be considered when upgrading or customizing the**

kernel:

1. Determining Whether to Upgrade to a New Kernel
2. Upgrading versus Customizing
3. Preparing to Upgrade
4. Installing a Kernel RPM
5. Getting the Kernel Source
6. Configuring the Kernel
7. Reviewing the Configuration Options
8. Compiling the Kernel
9. Installing the Kernel
10. Updating GRUB

15.8.1 Determining Whether to Upgrade to a New Kernel

Should you upgrade to a new kernel? Strictly speaking, no, it is rarely *necessary* to do so. The kernel provided with Fedora Core and RHEL is deliberately configured to support the widest possible array of existing PC hardware. Moreover, practically all of the functionality that most users need is already available in the current kernel. The fact is that most users do not need to do this because they use perhaps 20 percent of the feature sets of existing software. Adding still more unused features contributes to software bloat and, potentially, to system instability.

When is it necessary to rebuild the kernel? Often as not, you rebuild the kernel usually to provide better support for the odd hardware device, to add support for new hardware you have added to an existing system, to add a driver for a device not supported by the existing kernel, to fix the occasional bug, or to close a security hole.

15.8.2 Upgrading versus Customizing

Customizing the kernel and upgrading the kernel refer to two different procedures. *Customizing* the kernel refers to reconfiguring an existing kernel source code tree, recompiling it, installing the new kernel, and booting it. You can download either a complete source tree (now over 30 MB even when compressed) or one or more patches. Of the two options, downloading a series of patch files is faster than downloading an entire kernel source tree. *Upgrading* the kernel means a new kernel image, probably from kernel RPMs provided by the Fedora Project or Red Hat Software (RHEL users). Whether you customize or upgrade, though, the end result is the same: a new kernel configured to your liking.

15.8.3 Preparing to Upgrade

Upgrading your kernel is a major change, so prudence and experience dictate taking some preventive measures first. Make sure that you have a working boot disk for your system in case a

problem occurs. You will not be able to boot your system unless you have a boot disk that you know works.

1. Insert a disk into the disk drive, and perform a low-level format
2. Create an ext2 file system on the floppy disk
3. Create the mount point for the disk. Mount the disk so the GRUB installer can access it. Install the GRUB boot loader on the floppy's MBR.
4. Copy the GRUB configuration file onto the mounted floppy disk. Sync the disks and then unmount the floppy.
5. Reboot your system and boot from the GRUB boot floppy you just created to make sure that it works. If it does, you're ready to proceed.

15.8.4 Installing a Kernel RPM

In most cases, updated kernel RPMs will be installed automatically if you subscribe to the Red Hat Network or use up2date (or one of its analogues, such as yum). If the kernel is upgraded automatically by the Red Hat Network, the only task left for you is to reboot and select the new kernel (if it isn't the default) at the GRUB boot prompt. As remarked earlier, you might need to rebuild any LKMs provided by third parties against the new kernel. Of course, you don't have to rely on the Red Hat Update agent. To see if an updated kernel RPM is available for your version of RHEL, go to https://www.redhat.com/security/updates, select the version of RHEL you are using (such as Red Hat Enterprise Linux AS, version 4), and see if the errata include any kernel updates. If there is an update, which is usually due to a security advisory, the errata will indicate the kernel RPM you need to download *and* any additional RPM packages that are required.

15.8.5 Getting the Kernel Source

To obtain the kernel source, you have two options, downloading the kernel source in RPM format from Red Hat or the Fedora Project and downloading a kernel source code archive from one of the many Linux kernel mirror sites.

Using the source RPM (SRPM) is simpler because the build and installation process involves only a few steps. Another benefit is that the SRPM contains a number of modifications made by Red Hat engineers and contributors to the Fedora project. On the downside, however, the SRPM makes it difficult to modify the kernel configuration to your liking without doing some additional work that involves modifying the SRPM.

Downloading the source code from one of the kernel mirror sites is more involved but ultimately gives you more control over the kernel configuration and build process. The additional work involves

finding a kernel mirror to use, downloading a 35-MB archive file (of course, the SRPM is just as large), and of course, the actual configuration process. The payoff, however, is a much more customized kernel. Which method you use is up to you.

Using the Kernel Source RPM

Installing the kernel SRPM is simplicity itself. To install the SRPM after you have downloaded it, execute it using the rpm command. To be clear, you *have not* just installed a new kernel. Rather, you have only installed the source code for the kernel that happened to be in RPM (well, *source* RPM) format. This kernel source code archive is the kernel source as released by Linus Torvalds. The patch files are modifications made by Red Hat. To turn this source code into a kernel, execute with the rpmbuild command and then take a coffee break, because it might take a while to complete. The slower your system, the longer your coffee break.

When the process completes, you *still* haven't installed new kernel. The output of this process is a set of binary RPMs, one or more of which you *can* install. The three RPMs created (which you can find in /usr/src/redhat/RPMS/i686) are:

- kernel-2.6.10-1.770_FC3.root.i686.rpm
- kernel-smp-2.6.10-1.770_FC3.root.i686.rpm
- kernel-debuginfo-2.6.10-1.770_FC3.root.i686.rpm

Most people will install the first RPM. If you have an SMP system, install the second RPM. If you are a kernel developer or intend to be debugging the Linux kernel, install the third RPM.

Using Pristine Kernel Source

The method most long-time Linux users prefer for upgrading and customizing the kernel is to work with the pristine (unmodified) source code available from the various kernel archive sites scattered around the Internet. Why? Each major Linux vendor, including Red Hat, applies patches to the kernel source code that support the hardware of their strategic partners, to implement features requested by customers but not yet in the "official" kernel, and to differentiate themselves from their competitors. The primary site for the kernel source code is www.kernel.org (see Figure 15-6). The main kernel archive site is always busy, especially after a new release, so you are better off using one of its many mirrors throughout the world.

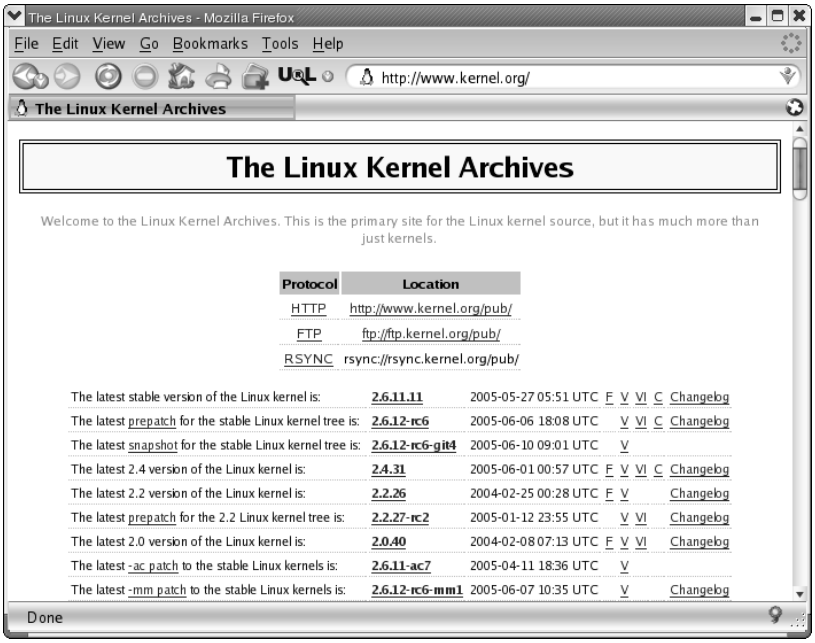


Figure 15-6 The Linux Kernel Archives home page.

Each mirror has a full archive of /pub/linux, the top-level kernel source directory, but it might not carry the source code in both gzip and bzip2 compression formats. The bzip2 format takes less time to download than gzip format, but takes longer than gzip format to decompress. After locating an archive that is near you, in network terms, download the desired file using your Web browser or an FTP client.

Verifying and Unpacking the Archive

Before you unpack the archive, you should check its signature to make sure that it has not been tampered with. Files placed in the Linux Kernel Archives are Open PGP-signed, and you can use these digital signatures to prove that files you have downloaded from the kernel archive site really originated at the Linux Kernel Archives. The current Linux Kernel Archives Open PGP key is always available from www.kernel.org/signature.html. The first step is to import the Linux Kernel Archive key. Then verify it with the archive kernel file that has been downloaded. As long as you see the two lines of output beginning with gpg:, the file is authentic and has not been tampered with or modified. Now you are ready to unpack the archive. Depending on the type of compressed file you have downloaded (bzip2 or gzip format) use the appropriate command to decompress the file. The result of either command is the same: a decompressed archive file that contains the Linux kernel source code.

Patching the Kernel

If you have already downloaded the main source code tree, you can save both bandwidth and time by downloading and

applying patches. Patches contain only changes to the underlying files from one kernel version to the next. To apply the patches, change directories to the directory in which you unpacked the kernel source code. The exact list of filenames varies from patch to patch, and some patches change more files than other patches do. The result, however, is a kernel source tree updated to the latest version. Finally, execute the ***make mrproper*** command to ensure that you are working with an unblemished source code tree. The command *make mrproper* removes any detritus remaining from previous kernel compiles — if you are starting from scratch, you can skip this step, but it does no harm to include it, either. You are ready, finally, to configure the kernel.

15.8.6 Configuring the Kernel

The kernel configuration process has changed significantly. Long-time Linux users will be pleased to know that *make config, make oldconfig, and make menuconfig* still work as they always have, albeit with more and different options. Those who have used *make xconfig*, however, are in for a bit of a surprise. The old Tk-based configuration tool has been replaced by kernel configuration interfaces based on Qt and GTK+. When you start the X-based configuration process using *make xconfig*, the kernel configuration tool tries to load a Qt-based tool (named *qconfig*). If the Qt toolkit isn't found, the process stops with an error. In this case, you should try *make gconfig* to invoke the GTK+-based kernel configuration tool. If that fails, then you'll need to use the ncuruses-based configuration tool by executing the command *make menuconfig*.

As a practical matter, it is easier to build the kernel in a no system directory because you don't have to mess with running as root. That's right, you don't have to be root to build the kernel. You only need root access for the postbuild steps, which include installing the kernel, installing the modules, and updating the GRUB configuration file.

Selecting a Kernel Configuration File

If you are unfamiliar with kernel configuration, you might want to consider using an existing kernel configuration file as a starting point for your custom configuration. If you are using kernel source code from kernel.org or one of its mirrors, you have a couple of options. For most architectures, you can look for files named *defconfig* or *defconfig.mumble* in the arch directory hierarchy. Each *defconfig* file represents a default configuration (hence the name, defconfig) with a standard, reasonably well-tested set of features and sane defaults for the specified architecture.

If your architecture or platform has a *defconfig*, you can use it by executing the '*make defconfig*' command in the top-level kernel source directory. This command creates a new configuration file

using the defaults in the *defconfig* file for your architecture. This creates a known starting point for your customized configuration file.

Configuring the Kernel with xconfig

To start configuring a kernel, change directories to the top level of your kernel source directory, type *make xconfig*, and press Enter. After a few seconds, you should see a screen resembling Figure 15-7.

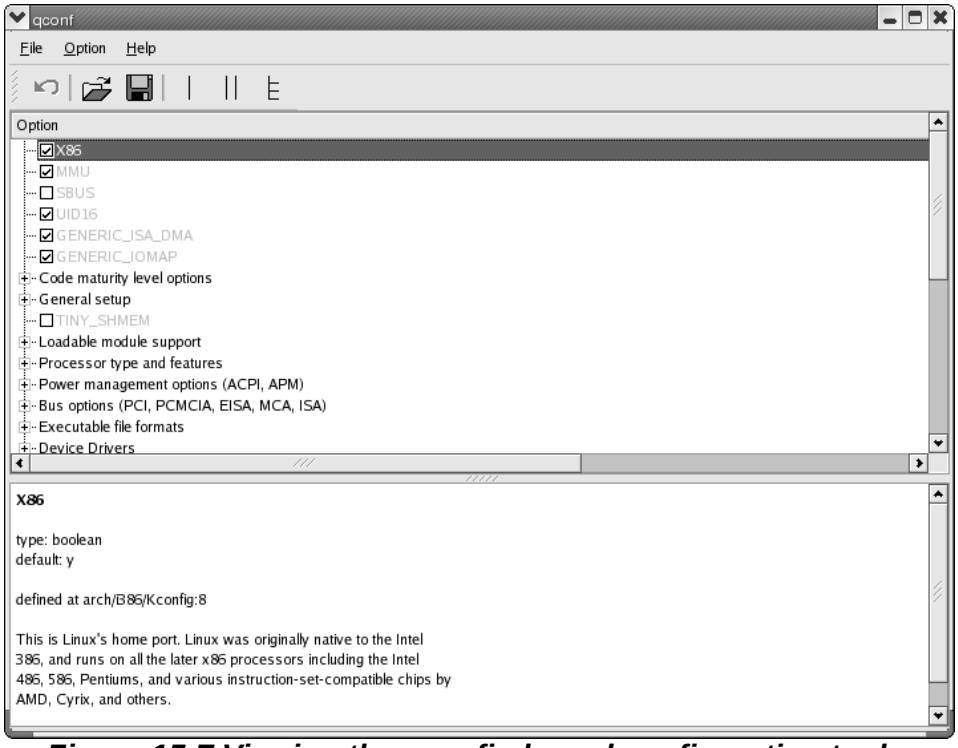


Figure 15-7 Viewing the xconfig kernel configuration tool.

For each configuration option, a blank box indicates that the corresponding feature is disabled; a check mark indicates that the corresponding feature is enabled; a dot indicates that it will be compiled as a module. A module, specifically, a *loadable kernel module*, or LKM, is a section of kernel code that can be inserted into and removed from the kernel dynamically (while the system is running) without needing to restart the system. LKMs are used to provide drivers and other kernel functionality that isn't always needed and so doesn't need to be compiled into the kernel. Compiling drivers as LKMs rather than compiling them into the kernel makes the resulting kernel smaller, use less memory, and load faster. To change an option, click the box to cycle through the three states. In some cases, you might not see an option, such as a device driver or a feature that should be present. If this occurs, select Option ➪ Show All Options.

To get the most information about various kernel configuration options, select Option ➪ Show Name to show the name of the kernel configuration option as it appears in the configuration file, Option ➪ Show Data to show the value of the kernel configuration options, and Option ➪ Show Range to show the range of possible values for kernel configuration options. To save your changes, click the floppy disk icon on the toolbar to save the configuration to the default file, .config, in the kernel directory (or select File ➪ Save from the menu. To save your changes to a different file, select File ➪ Save As from the menu and type a filename in the dialog box. You can load an existing configuration by clicking the folder icon on the toolbar or by selecting File ➪ Load from the menu and selecting the configuration file you want to use.

15.8.7 Reviewing the Configuration Options

Many configuration options need to be reviewed before compiling and installing the kernel. If you need additional information beyond what is provided by the help text in the tool itself, the kernel ships with a large amount of documentation in the form of text files in the Documentation subdirectory of the kernel directory. The configuration options that need to be reviewed are as follows:

- **Code Maturity Level Options:** Some kernel features and drives might not be fully tested. The Code maturity level options section enables you to choose whether you even see options that are known or suspected to have problems.
- **General Setup:** The General setup options enable allow you to set global kernel characteristics and enable kernel features that don’t fit neatly into the other categories.
- **Loadable Module Support:** The Loadable module support options enable you to control how (and if) the kernel will support loadable modules. Kernel modules are small pieces of compiled code that can be inserted into a running kernel to add support for infrequently used hardware devices and to enable support for other types of functionality.
- **Processor Type and Features:** The kernel features in this section enable you to customize the kernel for the specific processor (and, in some cases, processor features) in your system. The information you provide here is used to configure the kernel and to configure the build process itself to generate code that takes advantage of specific CPU features.
- **Power Management Options:** If you are configuring a kernel for a desktop system, you can jump ahead to the next section, because power management is not an issue for you. Notebook users will have to configure this option. If you enable Power

Management Support, you can choose between APM (Advanced Power Management) support and ACPI (Advanced Configuration and Power Interface) support.

- **Bus Options:** If you need support for one of the listed data busses, enable that support.
- **Executable File Formats:** Selecting executable file formats to support is simple. Stick with the defaults, enabling Kernel support for ELF binaries (Linux's native binary format) and Kernel support for MISC binaries, which allows you to execute arbitrary binary formats provided a module is available that supports the format.
- **Device Drivers:** The Device Drivers section is the longest section and has the most options. The general advice is simple and obvious: if your system does not have a class of devices, don't include support for that class of devices in your kernel. Including support for devices or features you don't have or don't need only makes the kernel larger. Even if you build support for unavailable devices as modules that you never load into the running kernel, all you succeed in doing is taking up disk space and making kernel compilation last longer than it needs to.
- **File Systems:** The File systems configuration section enables you to configure support for the file systems you expect to use or access from your system. One of Linux's greatest strengths is the breadth of its support for non-native file systems, which makes it possible for Linux to interoperate with almost any other operating system used on more than five computers.
- **Profiling Support:** Profiling support, if enabled, activates the kernel's support for the hardware performance counters built into modern CPUs and chipsets. By itself, this option does nothing unless you also enable Oprofile system profiling as a module in order to create data that you later turn into information.
- **Kernel Hacking:** The collection of features in the Kernel hacking section are not meant for mortal users. Our recommendation is to disable all kernel-hacking features unless you are specifically asked to enable them. These options and features make debugging the kernel easier and provide information that kernel developers can interpret or understand, but they have little value to anyone else.
- **Security Options:** The Security options section is another section best left alone unless you fully understand the consequences and usage of the security feature or tool in question.
- **Cryptography Options:** The Cryptographic options section enables you to select which of a multitude of cryptographic APIs

(ciphers) that you want to support in the kernel. The reason to include the cipher algorithms in the kernel is that kernel mode code executes faster and is more secure than code that executes in user mode.

- **Library Routines:** The configuration section for Library routines is a common source of confusion. The three options in this section exist for the purpose of supporting kernel modules built outside of the kernel tree that require cyclical redundancy check (CRC) support routines.
- **Saving the Kernel Configuration:** After you have worked your way through all of the configuration options, click the Save button on the toolbar or select File ➪ Save to save your configuration choices to the file .config in the kernel source directory. Then select File ➪ Exit to close xconfig and complete the configuration process.

15.8.8 Compiling the Kernel

To build the kernel and any loadable module you want, type make in the top-level kernel directory and take a coffee break. On an AMD Athlon Thunderbird 1200 with 512 MB of RAM, the compilation process took just over 50 minutes. Your mileage may vary.

15.8.9 Installing the Kernel

Happily, installing the new kernel takes considerably less effort and time than configuring and building it do. You just install the modules, copy the compressed kernel image into place, and create an initrd image. These steps require root access.

15.8.10 Updating GRUB

To make GRUB aware of the new kernel, open /boot/grub/grub.conf in the text editor of your choice and add a stanza at the bottom providing information about the kernel as shown in the following example:

```
title Kurt's Kustom Kernel (2.6.11.7)
          root (hd0,0)
          kernel /vmlinuz-2.6.11.7-custom ro root=/dev/hda3
          initrd /initrd-2.6.11.7.img
```

Figure 15-8 Sample Grub entry for new kernel

15.9 ADMINISTERING USERS AND GROUPS

Administering users and groups, or, more precisely, administering user and group *accounts*, is a fundamental Linux system administration activity. Ordinarily, most people understand user accounts as accounts tied to a particular physical user.

However, RHEL supports three fundamental user account types - root, normal and service.

The root user possesses full powers on the system. It is the superuser or the administrator that has full access to all services and administrative functions. This user is automatically created during RHEL installation. The normal users have user-level privileges. They cannot perform any administrative functions, but can run applications and programs that they are authorized to execute. The service accounts are responsible for taking care of the installed services. These accounts include apache, games, mail, printing and squid.

User account information is stored in four files - /etc/passwd, /etc/shadow, /etc/group and /etc/gshadow. These files are updated when a user account is created, modified or removed. The same files are referenced when a user attempts to log in to the system and, therefore, the files are referred to as user database files.

15.9.1 User Database Files

The /etc/passwd File

The /etc/passwd file contains vital user login data. Each line entry in the file contains information about one user account. There are seven fields per line entry separated by the colon (:) character. A sample entry from the file is displayed in Figure 15-9. The table 15-1 describes the fields in /etc/passwd file.

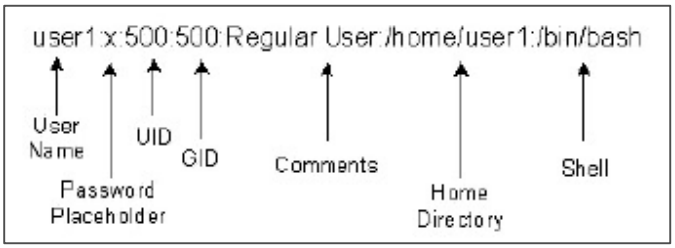


Figure 15-9 Sample Entry from /etc/passwd file

Table 15-1 Fields in the Password File

FIELD	DESCRIPTION
Username	The user's account name on the system
password	username's encrypted password or an x (points to the /etc/shadow file for the actual password) , or an asterisk "*" character (denotes that the account is disabled).
uid	username's numeric UID (user ID)
gid	username's numeric primary group ID (group ID)
gecos	An optional field used for informational purposes that usually contains username's full name
home	username's home directory
shell	username's login shell

The /etc/group File

The / etc/group file contains group information. Each row in the file contains one group entry. Each user is assigned at least one group, which is referred to as the users primary group. In RHEL, by default, a group name is same as the user name it is associated with. This group is known as users private group (UPG) and it safeguards the user' s files from other users' access. There are four fields per line entry in the file and are separated by the colon (:) character . A sample entry from the file is exhibited in Figure 15-10.

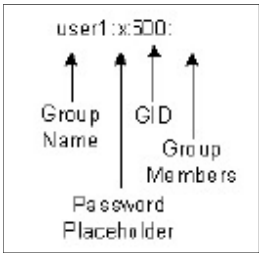


Figure 15-10 Sample Entry from /etc/group file

Here is what is stored in each field:

- The first field contains a unique group name, which must start with an alphabet. By default, each user gets a unique group whose name is the same as the user name.
- The second field is not typically used and is left blank. It may, however, contain an encrypted group - level password (copied and pasted from /etc/shadow file) or an 'x' which means that the actual password is defined in the /etc/gshadow file.

- The third field defines the GID, which is placed in the GID field of the /etc/passwd file. By default, groups are created with GIDs starting at 500 and match the username that they are assigned to.
- The last field holds usernames that belong to the group. Note that a user's primary group is defined in the /etc/passwd file, and not here.

The /etc/shadow File

The implementation of shadow password mechanism in RHEL provides an added layer of user password security. With this mechanism in place, not only the user passwords are encrypted and stored at an alternate location in a more secure file */etc/shadow,* but also certain limits on user passwords in terms of expiration, warning period, etc. can be implemented on a per-user basis. This is referred to as *password aging.* The *shadow file* is only readable by the root user, which makes the contents of the file concealed from everyone else. With shadow password mechanism active, a user is initially checked in the passwd file and then in the *shadow* file for authenticity.

The *shadow* file contains extended user authentication information. Each row in the file corresponds to one entry in the passwd file. There are nine fields per line entry and are separated by the colon (:) character. A sample entry from this file is exhibited in Figure 15-11.

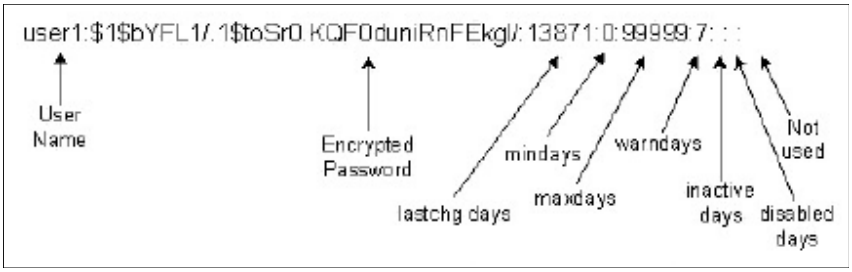


Figure 15-11 Sample Entry from /etc/shadow file

Here is what is stored in each field:

- The first field contains a login name as appeared in the */etc/passwd file.*
- The second field contains a combination of random letters and numbers, which represents a user password in an encrypted form.
- The number of days since 1 January 1970 that the password was last changed
- The number of days permitted before the password can be changed

- The number of days after which the password must be changed
- The number of days before the password expires that the user is warned the account will expire
- The number of days after the password expires before the account is disabled
- The number of days since 1 January 1970 after which the account is disabled
- Reserved for future use

The /etc/gshadow File

The shadow password implementation also provides an added layer of protection at the group level. With this mechanism activated, the group passwords are encrypted and stored at an alternate location in a more secure file */etc/gshadow,* which is only readable by the root user, and, therefore, hides the contents from everyone else.

The gshadow file contains group encrypted password information. Each row in the file corresponds to one entry in the *group* file. There are four fields per line entry and are separated by the colon (:) character. A sample entry from this file is exhibited in Figure 15-12.

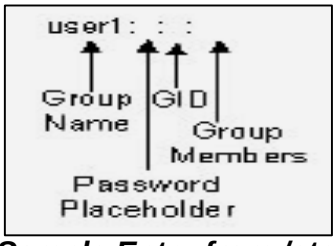


Figure 15-12 Sample Entry from /etc/gshadow file

Here is what is stored in each field:

- The first field contains a group name as appeared in the */etc/*group file.
- The second field may contain a combination of random letters and numbers if a group password is set using the *gpasswd* command. These characters hold the password in an encrypted form.
- The third field lists usernames of group administrators that are authorized to add or remove members to and from this group with the *gpasswd* command.
- The last field holds usernames that belong to the group.

15.9.2 Working with User Accounts

One of the most common administrative tasks is working with user and group accounts. Although some administrators find

the traditional command line tools for managing users and groups tedious or inconvenient to use, the section titled “Using the User Manager” covers the User Manager tool, a GUI application for creating, modifying, and deleting both users and groups. The commands for adding, modifying, and deleting user accounts are as follows:

- **useradd** — Create user login accounts
- **userdel** — Delete user login accounts
- **usermod** — Modify user login accounts
- **passwd** — Set or change account passwords
- **chage** — Modify password expiration information

Creating a User Account

Use the *useradd* command to create a user account. This command adds entries to the passwd file and optionally to the *group* file. It also inserts entries to the *shadow* and *gshadow* files provided password shadowing is enabled. The command creates a home directory for the user and copies default user Initialization files from the skeleton directory */etc/skel into* the users home directory. The syntax of useradd command is:

```
useradd [-c comment] [-d dir] [-e date]
        [-f time] [-g initial] [-G group[,...]]
        [-m [-k dir] | -M]
        [-p passwd] [-s shell] [-u uid [-o]]
        [-n] [-r] username
```

Table 15-2 lists the options useradd accepts

OPTION	DESCRIPTION
`-c comment`	Uses `comment` for the name field
`-d dir`	Names the new user’s home directory `dir`
`-e date`	Sets the account’s expiration date to `date`
`-f time`	Disables the account `time` days after the password expires
`-g group`	Sets the user’s primary group membership, or login group, to `group`
`-G [group[,...]]`	Makes the user a member of each supplemental group `group`

`-m`	Creates the home directory if it does not exist and copies the files and directory structure in `/etc/skel` to the new directory
`-k dir`	Copies the files and directory structure in `dir`, not `/etc/skel`, to the new home directory; must be specified with `-m`
`-M`	Disables creation of the home directory; cannot specify `-m` and `-M`
`-p passwd`	Sets the account password to the encrypted password `passwd`
`-s shell`	Sets the user's default shell to `shell`
`-u uid`	Sets the user's UID (User ID) to `uid`, which must be a unique number
`-o`	Allows the UID specified with `-u uid` not to be unique; must be specified with `-u`
`-n`	Disables use of Red Hat's user private groups
`-r`	Creates a system account (an account with a UID less than 100) but does not create a home directory
`username`	Sets the login name to `username`

Deleting a User Account

Deleting a user account deletes the user from the system. Execute the userdel command for this purpose.

```
userdel [-r] username
```

username identifies the user account to delete. Using -r deletes the corresponding home directory and mail spool. Without -r, userdel removes only the account references in the user and group database files. You cannot delete the account of a logged in user, so userdel fails if username is logged in.

Modifying a User Account

You can modify a user account with the *usermod* command. The syntax of the command is very similar to that of the *useradd* command. Its syntax is:

```
usermod [-c comment] [-d dir [-m]] [-e date]
        [-f inactive] [-g group] [-G group[,...]]
        [-l new_username] [-p passwd]
        [-s shell] [-u uid [-o]] [-L|-U] username
```

usermod accepts the options and arguments listed for useradd and adds three new ones, -l new_username, -L and -U. -l new_username changes the account name from username to new_username. -L disables (locks) username's account by placing a ! in front of the user's encrypted password in /etc/shadow. -U enables (unlocks) the account by removing the !. At least one option must be specified, but -p, -U, and -L may not be used

together in any combination. If username is logged in, usermod fails because you cannot change the login name of a logged-in user.

Setting the Password

The passwd command, generally regarded as “the password changing utility”, actually has more capabilities than merely changing passwords. In general, it updates all of a user’s authentication tokens, of which the login password is only one. Its syntax is:

```
passwd [-dkluf] [-S] username
```

-d removes the password for username, disabling the account. -k causes passwd to update only expired authentication tokens (passwords, in this case). -l or -u lock or unlock, respectively, username’s password by placing and removing a ! in front of username’s password in /etc/shadow. Finally, the -S option displays a short status message about username, indicating whether the account is locked or unlocked, the kind of encryption used, and so forth.

Setting and Modifying Password Aging

The *chage* command is used to set and alter password aging parameters on a user account. Table 15-3 lists and describes key options available with this command. Its syntax is:

```
chage [-l] [-m mindays] [-M maxdays] [-d lastday] [-I inactive]
      [-E expiredate] [-W warndays] username
```

Option	Description
–d (--lastday)	Specifies a date or number of days since the epoch time when the password was last modified. Corresponds to the third field in the *shadow* file.
–m (--mindays)	Specifies that the password cannot be changed before this many days are elapsed. Corresponds to the fourth field in the *shadow* file.
–M (--maxdays)	Denotes maximum days of validity of the password before a user starts getting warning messages to change password. Corresponds to the fifth field in the *shadow* file.
–W (--warndays)	Defines number of days a user gets warning messages to change password. Corresponds to the sixth field in the *shadow* file.
–I (--inactive)	Defines number of days of inactivity after a password is expired and before the account is locked. Corresponds to the seventh field in the *shadow* file.
–E (--expiredate)	Specifies a date or number of days since the epoch time on which the user account becomes deactivated. Corresponds to the eighth field in the *shadow* file.
–l	Lists password aging attributes set on a user account.

Table 15-3 chage command options

15.9.3 Modifying Multiple Accounts Simultaneously

Using useradd to add one or two accounts is relatively simple, but it quickly becomes tedious if 10 or 20 accounts need to be created. Fortunately, the shadow password suite includes the newusers utility, which can be used to create and update multiple user accounts. One of the advantages of command line tools is that they can be used to perform bulk or mass changes. Two commands, chpasswd and newusers, make multiple changes to the user password database in a single operation. The syntax is:

```
newusers userfile
```

userfile is the name of a text file consisting of lines in the same format as the standard password file, subject to the following exceptions:

- The password field appears as clear text — newusers encrypts it before adding the account.
- The pw_age field is ignored for shadow passwords if the user already exists.
- The GID can be the name of an existing group or a nonexistent GID. If the GID is the name of an existing group, the named user is added to that group, but if it is a nonexistent numeric value, a new group with the specified GID is created.
- If the specified home directory refers to a nonexistent directory, newusers creates it. If the directory already exists, ownership of the directory is set to that of the named user.

The following code shows the contents of newusers.txt, which is passed to newusers to create three new user accounts, Ram, Laxman, and Bharat:

```
ram:mypass:901:901:RAM:/home/ram:/bin/bash
laxman:yourpass:902:902:LAXMAN:/home/laxman:/bin/bash
bharat:somepass:903:903:BHARAT:/home/bharat:/bin/bash
```

After executing the command newusers newusers.txt, you will see the entries in /etc/passwd, /etc/group, and /etc/shadow.

15.9.4 Viewing Login and Process Information

To view current and past login information and to determine what processes users are running, you can use one of the following commands:

- **last** — Displays historical login information
- **who** — Displays information about currently logged in users
- **w**— Displays a user's currently running process

For all logins, last prints the user name, TTY, date, time, elapsed time, and the host name or IP address of the remote host, if applicable, from which the login originated of all user logins,

starting with the most recent login. Its syntax is:

```
last [-R | [-ai]] [-num |-n num] [username] [tty]
```

By default, last lists all the entries in /var/log/wtmp, so you can use -num and -n num to specify the number of output lines to display. Ordinarily, last displays the hostname in the third column, but using -a places the hostname in the rightmost column, -i shows the hostname's IP address, and -R completely suppresses display of the hostname. To view the login activity of a specific user, use the username argument. tty enables you to view logins per TTY. Multiple usernames and ttys can be listed.

The who command displays information about currently logged-in users. Its default output includes the user name, login TTY, and the date and time each user logged in. who's syntax is:

```
who [-Hil] | [-q]
```

Using the -H option adds column headings to who's output. Specifying -I adds each user's idle time to the display. Use -l to force who to show fully qualified domain names (FQDNs). To obtain the total number of logged-in users, use the -q option by itself.

The w command is very similar to who, except that it also displays the command line of each user's currently running process and a summary of each user's CPU usage. w's syntax is:

```
w [-husf] [username]
```

By default, w prints header information when it starts; -h disables the header. -s generates a short output format that omits the login time and the CPU usage. -f disables displaying the host from which users are logged in. Specifying username lists only username's login session and process information.

15.9.5 Working with Group Accounts

Unlike user accounts, group accounts always represent some sort of logical organization of users. Like user accounts, groups have group identification numbers, or GIDs, and it is common for users to be members of several groups. Groups are used to tie one or more users together to simplify administrative tasks.

Creating Groups

To create a new group, use the groupadd command. Its syntax is:

```
groupadd [[-g gid [-o]] [-r] [-f] groupname
```

groupname is the only required argument and must be the name of a nonexistent group. When invoked with only the name of the new

group, groupadd creates the group and assigns it the first unused GID that is both greater than 500 and not already in use. Specify -f to force groupadd to accept an existing groupname. Use the -g gid option if you want to specify the new group's GID, replacing gid with a unique GID (use the -o option to force groupadd to accept a nonunique GID). To create system group, one that has special privileges, use the -r option.

Modifying and Deleting Groups

After creating a new group, you will likely want to add user accounts to it. Two commands modify group accounts, each serving different purposes. groupmod enables you to change a group's GID or name, and gpasswd enables you to set and modify a group's authentication and membership information. You should rarely need to change a group's name or GID; you're on your own to read the groupmod's short manual page. We're more interested in gpasswd, which enables the root user to administer all aspects of a group account and to delegate some administrative responsibilities to a group administrator. For simplicity's sake, the following discussion explains the uses of gpasswd *only* available to root. Then it covers the gpasswd calls a group administrator can perform. From root's perspective, gpasswd's syntax is:

```
gpasswd [-A username] [-M username] groupname
```

Root can use -A username to assign username as groupname's group administrator. -M username adds username to groupname's membership roster. Assigning a group administrator using -A does not make the administrator a member of the group; you have to use -M to add the administrator as a member of the group.

15.9.6 Administering Users and Groups with User Manager

User Manager is a graphical tool for administering user and group accounts. To use it, you must be logged in as root or otherwise have root access. To start User Manager, click Main Menu ➪ System Settings ➪ Users and Groups. You can start from a command line using the command system-config-users in a terminal window. The initial screen resembles Figure 15-13.

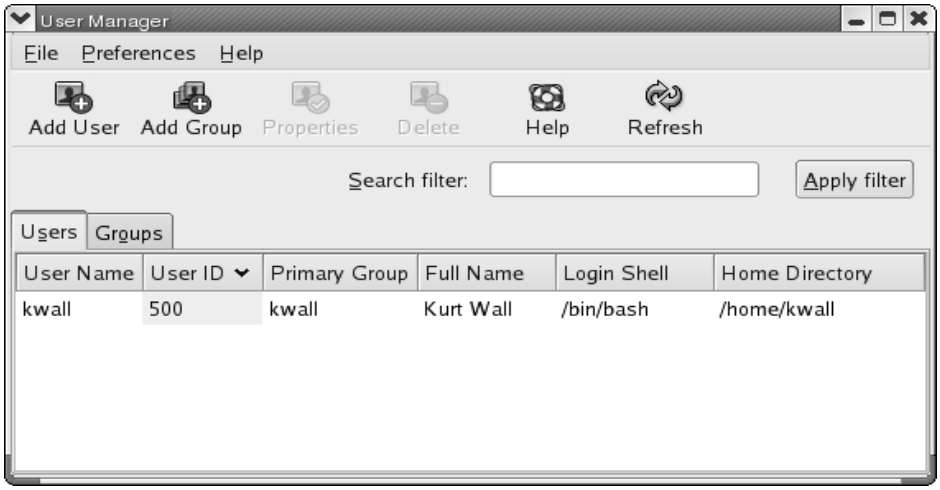


Figure 15-13 The main Red Hat User Manager dialog box.

From this screen you can view, modify, and delete existing user and group accounts or create new ones. To reduce the list of displayed accounts or to search for a specific account, type the first few letters of an account name in the Filter by text box and click the Apply filter button. You can update most windows by clicking the Refresh button on the toolbar. To get context-sensitive help, click the toolbar's Help button or, to view the entire User Manager manual, select Help ➪ Manual from the toolbar.

Creating User Accounts

To add a new user:

1. Click the Add User button. The Create New User dialog box (Figure 15-14) appears.

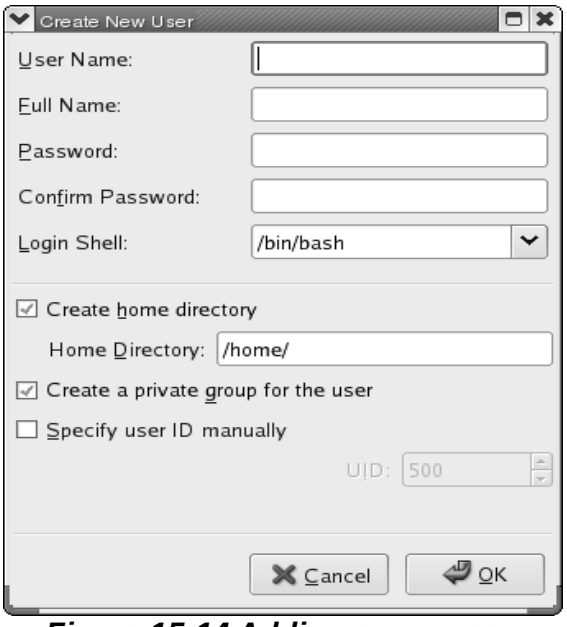


Figure 15-14 Adding a new user.

2. Type the new account name in the User Name text box.
3. Type the user's full name in the Full Name text box.
4. Type the user's password in the Password and Confirm Password fields. The password must be at least six characters.
5. Select a login shell. If you choose not to accept the default shell, select an alternative shell from the Login Shell drop-down box.
6. As noted earlier in this chapter, the default home directory is /home/username. You can change the home directory by editing the Home Directory text box or not create a home directory at all by clearing the Create home directory check box.
7. To prevent creation of a user private group, remove the check from the Create new group for the user check box.
8. Click OK to create the user.

Modifying and Deleting User Accounts

After you have created a user account, you can configure additional properties by clicking User Manager's User tab, selecting the user, and clicking the Properties button to open the User Properties dialog box. To add the user to additional groups, click the Groups tab. Click the check box next to the groups of which the user should be a member, then click the Apply button (See Figure 15-15).

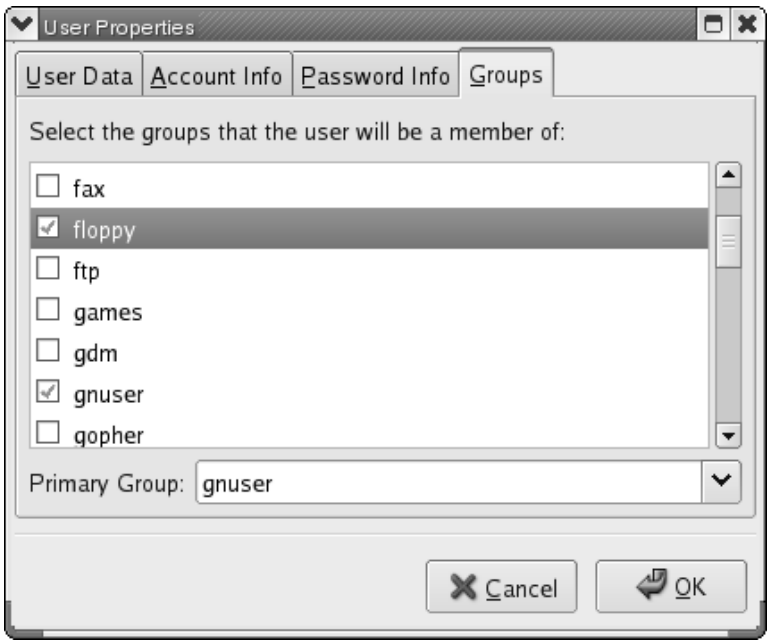


Figure 15-15 Adding a user to additional groups.

Other account data you can modify from the User Properties window includes the basic user information you supplied when you created the user (the User Data tab), account information (the

Account Info tab), and password expiration information (the Password Info tab). On the Password Info tab, click the Enable account expiration check box to set the user account's expiration date if you want the account to expire on a certain date. To prevent this user account from logging in, place a check mark in the User account is locked check box.

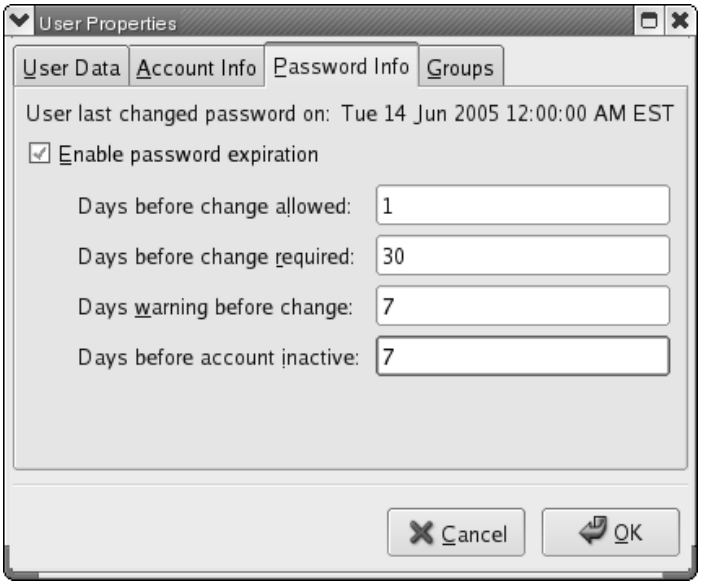


Figure 15-16 Modifying user account password expiration information.

Click the Password Info tab to view and change the account password expiration information. (See Figure 15-16.) The date that the user last changed her password appears across the top of the tab. Click Enable password expiration to force a password change after a certain number of days, and then enter the number of days between required password changes in the Days before change required text box. You can also set the number of days before the user can change her password, the number of days before the user is warned to change her password, and the number of days before the account becomes inactive. When you have finished modifying the user account properties, click OK to apply the changes and close the User Properties dialog box. Finally, to delete a user account, click the account to delete on User Manager's Users tab, and then click the Delete button.

Creating Group Accounts

To add a new user group, click the Add Group button. In the Create New Group dialog box, type the name of the new group, and then click OK to create the group.

Modifying and Deleting Group Accounts

To view or modify the properties of an existing group, select the group to modify from the group list on the Groups tab and click the Properties button. The Group Properties dialog box appears.

The Group Users tab, shown in Figure 15-17, displays the users that are members of the group. To add other users to the group, place a check mark next to the user account names in the list, and deselect account names to remove them from the group. Click OK to apply the changes and close the Group Properties box. After you have finished adding or modifying user and group accounts, click File ➪ Quit or press Ctrl+Q to save your changes and close User Manager.

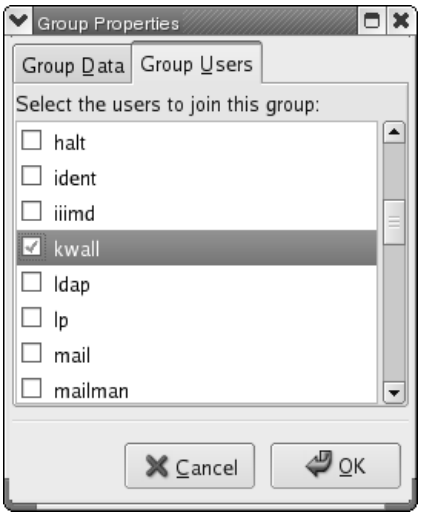


Figure 15-17 Modifying group properties.

15.9.7 Understanding the Root Account

With very few and limited exceptions, the root account has unlimited power on any Linux or UNIX system, and, in this respect, Red Hat Linux is no exception. The root account can access any file and modify any process. Indeed, it is for this reason that root is often called the superuser — root is effectively omnipotent.

The exceptions to root's capabilities are few. As explained earlier, root on an NFS client (that is, a system mounting an NFS exported file system from an NFS server) typically cannot exercise root privileges on the exported file system because the NFS server exports the file system using the root_squash option. On the local system and for local resources, root always has unlimited power and control. It is only in the case of remote or networked resources where root's power is subject to special considerations and restrictions on root's power emerge.

The ext2 and ext3 file systems also restrict root's power, although only slightly. The ext2 and ext3 file systems support a

number of special file attributes, including immutability. Using the *chattr* utility, root can set a file's immutable attribute, which prevents all users, including root, from modifying the file; it cannot be deleted, renamed, or written to, and hard links cannot be created to it until the immutable attribute is cleared. You guessed it — only root can set or clear a file's immutable attribute.

Implementing Sudo

Considering root's privileges, you can easily understand why root access on a Linux system is carefully protected and the root password tightly guarded. Nevertheless, it is often desirable to grant privileges to a nonroot user (humorously referred to as merely mortal user) that have traditionally been solely root's domain, such as printer management, user account administration, system backups, or maintaining a particular Internet service. In other operating systems, such users are often called wheel users or administrative users.

How do you grant administrative privileges to merely mortal users without providing unfettered root access? In many situations, Sudo, a mnemonic for superuser do, is one solution. Sudo enables you to give specific users or groups of users the ability to run some (or all) commands requiring root privileges. Sudo also logs all commands executed, which allows you to maintain an audit trail of the commands executed, by whom they were executed, when they were executed, and so on. As the README in the source distribution states, Sudo's "basic philosophy is to give as few privileges as possible but still allow people to get their work done." Sudo's features include:

- Enabling the ability to restrict the commands a given user may run on a per-host basis.
- Maintaining a clear audit trail of who did what. The audit trail can use the system logger or Sudo's own log file.
- Limiting root-equivalent activity to a short period of time using time-stamp based "tickets," thus avoiding the potential of leaving an active root shell open in environments where others can physically get to your keyboard.
- Allowing a single configuration file, /etc/sudoers, to be used on multiple machines, permitting both centralized Sudo administration and the flexibility to define a user's privileges on a per host basis.

Deciphering Sudo's Configuration File

Sudo's configuration file, /etc/sudoers, is the key file. It contains three types of entries: alias definitions, privilege specifications, and global configuration defaults.

Alias definitions are variables or placeholders that you can reuse throughout the configuration file. They come in four flavors: user aliases, command aliases, so-called runas aliases, and host aliases. The rationale for aliases is to simplify maintaining the configuration file — rather than editing multiple user or command lists when you update /etc/sudoers, you simply modify the appropriate alias and let sudo substitute the alias definition in each place where it is used. Privilege specifications define which users may execute what commands. Global configuration defaults are general settings that control sudo's overall behavior.

The general procedure is to use visudo to edit /etc/sudoers and create the following:

- A user alias defining the users to whom you are granting access to one or more commands
- A command alias that represents the command or commands to execute
- A host alias to identify the host or hosts on which the named users are permitted to execute the named command (if necessary)
- A runas alias that identifies the user a command should run as (again, if necessary)
- A user privilege specification to connect the necessary aliases together to form a Sudo rule

15.10 Installing and Upgrading Software Packages

RHEL provides a rich set of tools for installing and managing software packages on the system. Red Hat software packaging is based on a special format called *Redhat Package Manager* (RPM). All packages available in and for RHEL are in this format. An RPM package contains necessary files, as well as metadata structures such as ownership, permissions and directory location for each individual file included in the package.

15.10.1 Using the Red Hat Package Manager

RPM is a powerful software configuration manager and the preferred tool for installing, removing, verifying, and updating software packages on Fedora Core and RHEL systems. RPM consists of two components: a set of databases that store information about installed software and the programs that interface with the databases. RPM can work with binary and source packages.

Binary packages, referred to simply as RPMs, contain compiled software ready for installation. They use the file extension

.rpm. Source packages, more often called *source RPMs* or *SRPMs*, are uncompiled packages containing source code, patches, and build instructions, all of which are used to create binary RPMs. SRPMs have a .src.rpm file extension. Because RPM offers a rich feature set that makes it seem complex and difficult to learn to use, the following sections each explore one of RPM's modes, in order to simplify the discussion:

- **General options** - The general options control the overall behavior of the rpm command line tool, such as displaying basic usage information.
- **Querying** - The query functions can be used to obtain a considerable amount of information about installed software.
- **Package maintenance** - Package maintenance enables package installation, removal, and upgrading.
- **Administrative and Miscellaneous options** - The administration and miscellaneous modes, finally, affect RPM itself, rather than software packages. They are used to fix possible database corruption and to determine RPM's general configuration.
- **Package verification** - Package verification gives system administrators the ability to compare the present state of files installed by an RPM against information taken from the original package.

15.10.2 Managing Packages with rpm

This section discusses package management tasks including listing, installing, upgrading, freshening, querying, removing, extracting, validating and verifying packages using the rpm command. Before getting into details, let us take a look at Table 15-4, which provides a list of options commonly used with the rpm command. These options may be used in either way.

Option	Description
–a (--all)	Displays all packages.
–c (--configfiles)	Displays configuration files.
–d (--docfiles)	Displays documentation files.
–e (--erase)	Removes a package.
–f (--file)	Displays information about the specified file.
–F (--freshen)	Upgrades an existing package. An older version of the package must exist in order to upgrade it.
--force	Installs a package even if the same version already exists.
–h (--hash)	Displays progress of package installation / upgrade.
–i (--info)	Displays basic information about a package.
--import	Imports the specified public key.
–i (--install)	Installs a package.
–K (--checksig)	Validates the signature and also the package integrity.
–l (--list)	Lists files in a package.
–p (--package)	Verifies an installed package against an installable package.
–q (--query)	Queries and displays packages. You can use the *rpmquery* command instead.
--replacepkgs	Overwrites existing packages.
–R (--requires)	Lists dependencies without which a package cannot be installed.
–U (--upgrade)	Upgrades an existing package or installs if not already installed.
–v or –vv	Displays detailed information.
–V (--verify)	Verifies the integrity of package files. You can use the *rpmverify* command instead.

Table 15-4 rpm command options

Listing Installed Packages

Run either of the following to list all installed packages:

```
rpm -qa
rpmquery -a
```

These commands list an updated list of all packages currently loaded on the system. Alternatively, you can view the contents of the /var/log/rpmpkgs file, which maintains a list of installed packages. This file might not have the latest information as it is updated once a day.

Installing a Package

Installing a package creates directory structure for the package and installs the required files. The basic syntax for installing an RPM is:

rpm -i [options] package [...]

package is the complete name of the RPM to install and options refines the installation process. Table 15-5 lists commonly used options values. See the rpm man page for a comprehensive listing.

OPTION	DESCRIPTION
`--force`	Install the package even if it is already installed, install an older package version, and replace files already installed. `--force` also ignores dependencies.
`-h`	Print up to 50 hash marks (#) to illustrate the progress of the installation.
`--nodeps`	Do not perform a dependency check before installing or upgrading a package.
`--test`	Do not install the package or update the database, just identify and display possible conflicts or dependency errors.
`-v`	Be slightly verbose and show some useful information during the installation.

Table 15-5 Common rpm installation options

The following command demonstrates installing an RPM:

```
# rpm -ivh fortune-mod-1.2.1-1.i386.rpm
Preparing...                ########################################### [100%]
   1:fortune-mod            ########################################### [100%]
```

Upgrading a Package

Upgrading a package upgrades the specified package if an older version of the package is already installed. If an older version is not already there, it will go ahead and install it. The options for upgrading existing RPMs come in two flavors, -U, for upgrade, and -F, for freshen. What is the difference between upgrading a package and freshening it? Upgrading a package, using -U, installs it even if an earlier version is not currently installed, but freshening a package, using -F, installs it only if an earlier version is currently installed. Other than this subtle but important difference, -U and -F are identical to -i, even down to the options they accept (see Table 15-5).

The following sequence of commands illustrates how to upgrade an RPM and the difference between the -U and -F options:

```
# rpm -Fvh fortune-mod-1.0-13.i386.rpm
# rpm -q fortune-mod
package fortune-mod is not installed
```

Hmm. Nothing happened. The rpm command line used -F, so it did not install the fortune-mod package because an earlier version did not exist.

```
# rpm -Uvh fortune-mod-1.0-13.i386.rpm
Preparing...                ########################################### [100%]
   1:fortune-mod            ########################################### [100%]
```

With -U, RPM “upgraded” the fortune-mod package, even though an earlier version was not installed.

```
# rpm -Fvh fortune-mod-1.2.1-1.i386.rpm
Preparing...                ########################################### [100%]
   1:fortune-mod            ########################################### [100%]
```

This time, the freshen operation succeeded.

Removing a package

Removing or deleting RPMs and their contents is easy, perhaps frightfully so. The general form of the command is:

```
rpm -e package [...]
```

The -e option is a mnemonic for expunge. package is the name, only, of the RPM to remove. Multiple packages can be removed simultaneously by listing each package on the command line. For example, the following command removes the fortune-mod and whois RPMs:

```
# rpm -e fortune-mod whois
```

Notice that successful removal generates no additional output.

Querying a Package

RPM's query mode is one of its most powerful and useful features. The general form of an RPM query is:

```
rpmquery [query_opts]
```

rpmquery (or, if you prefer the old style, rpm -q or rpm --query) specifies a query operation and query_opts specifies what to query, the type of query, how the query should run, or the format of its output. You can use the command rpmquery in place of rpm -q or rpm --query. Most commonly, queries use the following general syntax:

```
rpmquery [query_opts] package [...]
```

package names the RPM to query. Table 15-6 lists many but not all of the options available in query mode. The Type column uses S to mark a package selection option and I to mark an information selection option. Unless mentioned otherwise, all options require at least one package name as an argument.

OPTION	TYPE	DESCRIPTION
`-a`	S	Queries all installed RPMs. Does not require a package specification.
`-c`	I	Lists only the configuration files stored in the queried RPM(s).
`--changelog`	I	Displays change information about the queried RPM(s).
`-d`	I	Lists only the documentation files stored in the RPM.
`--dump`	I	For each file stored in the queried RPM(s), displays its path, size, modification time, MD5 checksum, permissions, owner, group, and whether it is a configuration file, documentation file, a device, or a symlink (must be used with -l, -c, or -d).
`-f file`	S	Queries the RPM that owns `file`. Does not require a package specification.
`-g group`	S	Lists the packages in the RPM group named `group`. Does not require a package specification.
`-i`	I	Displays complete information about the queried RPM(s).
`-l`	I	Lists all of the files stored in the RPM.
`--last`	I	Displays the installation date and time of each RPM queried, starting with the most recently installed RPM.
`-p package [...]`	S	Queries the uninstalled RPM named `package`.
`--provides`	I	Lists all of the capabilities the queried RPM(s) provides.
`--qf 'format_str'`	I	Creates a customized output format for displayed information, using `format_str` as the model.
`--querytags`	I	Prints all known tags for use with the --qf option. Does not require a package specification.
`--requires`	I	Lists all RPMs on which the package depends.
`-s`	I	For each file in the original RPM, displays its state, which is one of normal, not installed, or replaced.
`--whatprovides capability`	S	Queries all RPMs that provide `capability`.
`--whatrequires capability`	S	Queries all RPMs that need `capability` in order to function properly.

Table 15-6 RPM Query Mode Options

Querying Package Dependencies

The --provides, --requires, --what requires, and --what provides options allow you to identify dependencies between packages. The capability argument represents the dependency itself, which is often the name of another RPM or the name of a

particular file. RPM uses dependencies to maintain system integrity, so, for example, if one RPM requires something a second RPM provides, you cannot, in normal usage, delete the second RPM.

To query whether a package is installed , run any of the following:

```
# rpm -q sendmail
# rpmquery sendmail
sendmail-8.13.8-2.el5
```

To display basic information about a package , run any of the following :

```
# rpm -qi sendmail
# rpmquery -i sendmail
Name        : sendmail                     Relocations: (not relocatable)
Version     : 8.13.8                       Vendor: Red Hat, Inc.
Release     : 2.el5                        Build Date: Tue 28 Nov 2006 09:59:05 AM EST
Install Date: Mon 24 Dec 2007 12:22:26 PM EST      Build Host: ls20-bc1-14.build.redhat.com
Group       : System Environment/Daemons   Source RPM: sendmail-8.13.8-2.el5.src.rpm
Size        : 1370608                      License: Sendmail
Signature   : DSA/SHA1, Wed 17 Jan 2007 03:38:27 PM EST, Key ID 5326810137017186
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : A widely used Mail Transport Agent (MTA).
Description :
The Sendmail program is a very widely used Mail Transport Agent (MTA).
.....
```

15.10.3 Building Packages Using Source RPMs

In the simplest case, building and installing software from SRPMs requires one or possibly two commands. The same unpack/configure/build/install procedure described in the previous section takes place, but RPM handles each of these steps for you. In this section, you will learn how to use the two command cases (building and installing an RPM), and how to invoke each step of the RPM build process. The general form of the command to build a binary RPM from a source RPM is:

```
rpmbuild -b[stage] spec_file [...]
```

Any of the values listed in Table 15-7 is a valid value of stage.

STAGE	MNEMONIC	MEANING
a	All	Builds both binary and source RPMs
b	Binary	Builds only a binary RPM
c	Compile	Compiles the source code
i	Install	Installs the files
l	List	Makes sure that all the package files exist
p	Prep	Unpacks the source code and applies any patches
s	Source	Builds only a source RPM

Table 15-7 Valid Build Stages for RPM'S -b Mode

Once the packages are built, you can install them as you would any other binary RPM, using the command discussed above for installing a package.

15.10.4 Installing Software from Source

After downloading either an RPM, SRPM, or a source archive such as a tarball, naturally, you will want to install it. The section titled “Using the Red Hat Package Manager” detailed how to install binary RPMs and how to create binary RPMs from SRPMS. This section shows you how to install software from source code, that is, to unpack, configure, build, and install a software package you download as uncompiled source code. Before package-management suites such as RPM became popular, you upgraded and installed software by using the following steps:

- **Configuring the build environment:** For end users and system administrators, a build environment consists of the compiler, gcc, and its supporting libraries, the make utility for automating compiler invocations, a few key development libraries (mostly consisting of header files), and the install utility for handling the details of copying files and assigning the proper ownership and setting file permissions appropriately.
- **Downloading a gzipped tarball (a tar archive compressed using the gzip utility):** You can download the package from the source website or any of the mirror sites provided.
- **Unpacking it:** After downloading the package, move it to a location where it will not interfere with the system. You need to decompress and unpack the archive. The tar command combines decompression and unpacking the tar archive. Then use the gunzip command to send the result of the decompression to standard output.
- **Configuring it by manually editing one or more header files or using a configure script that automatically customized the package to your system:** Now that the package has been unpacked, the next step is to configure it for your system. In most cases, customizing a package for your system boils down to specifying the installation directory, but many packages allow you to request additional customizations. Alternatively you can use a configure script to automate configuration and customization. A *configure* script is a shell script that makes educated guesses about the correct values of a variety of system-specific values used during the compilation process. In addition, configure allows you to specify the values of these same values, and others, by invoking configure with command line options and arguments. Values that configure “guesses” and that you pass to configure on its command line are normally written to one or more makefiles, files that the *make* program

uses to control the build process, or to one or more header (.h) files that define the characteristics of the program that is built.

- **Executing make to build it:** To build the package, type ***make*** and press Enter. Depending on the size and complexity of the program you are building, make's output might be extensive.
- **Testing the build:** Many programs, especially those from the GNU projects, include some sort of test suite to validate the program. The idea is to make sure that the program works properly before installing it. In some cases, you execute the make test command to run the test suite. In other cases, a special subdirectory of the build tree, conveniently named test or Test, contains the test suite. Each package handles testing slightly differently, so read the package documentation.
- **Executing make install to install it:** In the case of many programs installed from source, installing the built and tested program is simply a matter of executing the command ***make install*** in the build tree's base directory. Programs that are more complex might have additional commands, such as ***make install-docs*** to install only documentation, that break up the installation into more steps or that perform only part of the installation. Still other packages might use scripts to perform the installation. Regardless of the process, however, the goal is the same: Install program executables and documentation in the proper directories, create any needed subdirectories, and set the appropriate file ownership and permissions on the installed files.

At this point, package installation is complete. One final exhortation before proceeding to the next section: ***Read the documentation!*** Most software you obtain in source code form includes one or more files explaining how to build and install the software; we strongly encourage you to read these files to make sure that your system meets all the prerequisites, such as having the proper library versions or other software components. The documentation is there to help you, so take advantage of it and save yourself some frustration induced hair loss!

15.11 SUMMARY

- Fedora Core and RHEL systems are commonly deployed to provide Internet services, so this chapter mentioned some methods you can use to improve the performance of several key Internet services: LDAP, DNS, email, and Web services.
  - We could list only some of the areas to consider when tuning LDAP, because LDAP performance tuning is a complex subject best addressed by the LDAP authorities.

  - DNS is more easily tuned. A DNS client's performance can often be improved simply by running a caching nameserver, while there are several methods available for getting better query performance from a server.
  - Mail servers are high-volume, heavy throughput systems requiring careful tuning, but sometimes, simply replacing Sendmail with Postfix can fix slow mail-processing times.
  - We also mentioned a number of methods you can use to get faster page-serving behavior from Apache.
- One of Linux's most persuasive selling points is the ability it gives you to fine-tune the kernel for your specific needs.
  - Often, it isn't necessary or advisable to build your own kernel, especially if you have a support agreement for RHEL.
  - The challenge when creating the do-it-yourself kernel is to understand all of the possible configuration options available to you or at least to know which of the options you don't need.
  - Whether you use a prebuilt binary kernel or make your own, make sure that you have a boot disk, perhaps even two, so you can still get into your system if the upgrade process fails.
- **In terms of Users and groups,**
  - You started off with building an understanding of /etc/passwd, /etc/shadow, /etc/group and /etc/ gshadow files. You looked at what the files contained, the syntax, and how to verify consistency.
  - You studied password shadowing and password aging. You learned user management including creating, modifying and deleting user accounts. You looked at how to set and modify password aging attributes on user accounts.
  - You learned a few simple commands that allowed you to switch into another user account, run privileged commands, display currently logged in users, and display recent user logins. Likewise, you studied group management including creating , modifying and deleting group accounts.
  - You also learned to create and manage user and group accounts using the User Manager graphical administration tool.
  - A brief recap of the power of the root account on a Red Hat Enterprise Linux system and showed you how to delegate some of that power to nonroot users using Sudo.

- Installing and Upgrading software packages becomes easy using Red Hat Package Manager (rpm).
  - RPM is a powerful software configuration manager and the preferred tool for installing, removing, verifying, and updating software packages. RPM can work with binary and source packages.
- Software packages can also be installed using the traditional methods (from source tarball) and tools. The basic procedure is the same for all packages: unpack the source archive, configure it as necessary, build it, test the program, and then install it.

15.12 Review Questions

i. How can performance of DNS client be optimized?

ii. What are the steps for optimizing FTP services?

iii. Illustrate the steps for optimizing Web Services.

iv. Explain the steps that need to be considered when upgrading or customizing the kernel.

v. Differentiate between upgrading the kernel and customizing the kernel

vi. Explain the files that hold the user account information.

vii. State and explain the commands used for

a) Adding a user b) Modifying the user c) Deleting the user

viii. Explain the chage command in detail.

ix. List and explain the commands used for viewing login and process information

x. How to create and manage user and group accounts using the User Manager graphical administration tool.

xi. What are the exceptions to root's capabilities?

xii. What is sudo? Explain sudo's features.

xiii. What is rpm? Explain some commands used for managing packages with rpm.

xiv. What is the difference between upgrading a package and freshening it?

xv. Write a short note about querying a package using rpm.

xvi. What steps are required to install the software from source? Explain in detail with suitable example.

15.13 BIBLIOGRAPHY, REFERENCES AND FURTHER READING

- Beginning Linux Programming 4th Edition by Neil Mathew, Richard Stone. Wiley Publishing
- Linux Administration: A Beginner's Guide, Fifth Edition, Wale Soyinka, Tata McGraw-Hill
- Linux: Complete Reference, 6th Edition, Richard Petersen, Tata McGraw-Hill
- Red Hat Linux Networking and System Administration 3rd Edition by Terry Collins and Kurt Wall.
- Sybex – RHCE Red Hat Certified Engineer Study Guide
- Red Hat Certified Technician & Engineer by Asghar Ghori.
- www.thegeekstuff.com
- www.tlpd.org
- www.linuxtopia.org