Bachelor of Computer Applications (BCA)

Linux Administration

ज्ञानगंगा घरोघरी

Yashwantrao Chavan Maharashtra Open University

Dnyangangotri, Near Gangapur Dam, Nashik 422 222

Unit No. and Name	Details
Unit 1 Introduction to Linux	**Introduction to Linux**: Open Source and Red Hat, Origins of Linux, GNU & Linux Distributions, Versions of Linux, Architecture of Linux. **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.
Unit 2 Installation of Redhat Linux	Installation of Redhat Linux on Virtual Machine, Understanding Partitions of Linux, Booting and shutting down Linux, Understanding Boot loaders: GRUB & LILO, Bootstrapping, Init process, rc scripts, Enabling and disabling services. Different Run levels in Linux, Understanding Linux file system structure.
Unit 3 Using Command Line and Managing Software	**Command Line:** Working with the Bash Shell, Working with basic linux command, Working with advanced linux commands, Working with Directories, Piping and Redirection, Finding Files, Using Vi Editor **Managing Software:** Understanding RPM, Understanding Meta Package Handlers, Creating Your Own Repositories, Managing Repositories, Installing Software with Yum, Querying Software, Extracting Files from RPM Packages.
Unit 4 Working with Users, Groups and Permissions	Managing Users and Groups, Commands for User Management, Managing Passwords, Modifying and Deleting User Accounts, Configuration Files, Creating Groups, Using Graphical Tools for User, and Group Management, Using External Authentication Sources, the Authentication Process, sssd, nsswitch, Pluggable Authentication Modules, Managing Permissions, the Role of Ownership, Basic Permissions: Read, Write, and Execute, Advanced Permissions, Working with Access Control Lists, Setting Default Permissions with umask, Working with Attributes
Unit 5 TCP/IP Networking and Network File System	**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 6 Configuring DNS and DHCP	Introduction to DNS, The DNS Hierarchy, DNS Server Types, The DNS Lookup Process, DNS Zone Types, Setting Up a DNS Server, Setting Up a Cache-Only Name Server, Setting Up a Primary Name Server, Setting Up a Secondary Name Server, Understanding DHCP, Setting Up a DHCP Server
Unit 7 Connecting to Microsoft Networks	Connecting to Microsoft Networks: Installing Samba, Configuring the Samba Server, Creating Samba Users 3, Starting the Samba Server, Connecting to a Samba Client,

and Setting up a Mail Server	Connecting from a Windows PC to the Samba Server Setting up a Mail Server: Using the Message Transfer Agent, the Mail Delivery Agent, the Mail User Agent, Setting Up Postfix as an SMTP Server, Working with Mutt, Basic Configuration, Internet Configuration, Configuring Dovecot for POP and IMAP
Unit 8 Securing Server with iptables and Configuring Web Server	Securing Server with iptables: Understanding Firewalls, Setting Up a Firewall with system-config-firewall, Allowing Services, Trusted Interfaces, Masquerading, Configuration Files, Setting Up a Firewall with iptables, Tables, Chains, and Rules, Composition of Rule, Configuration Example, Advanced iptables Configuration, Configuring Logging, The Limit Module, Configuring NAT Configuring a Web Server: introducing Apache, Configuring Apache, Implementing SSI, Enabling CGI, Enabling PHP, Creating a Secure Server with SSL

Yashwantrao Chavan Maharashtra Open University

CMP515

Linux Administration

Linux Administration

Yashwantrao Chavan Maharashtra Open University
Dnyangangotri, Near Gangapur Dam
Nashik-422222

Yashwantrao Chavan Maharashtra Open University

Vice-Chancellor: Prof. E. Vayunandan

SCHOOL OF COMPUTER SCIENCE

Dr. Pramod Khandare Director School of Computer Science Y.C.M.Open University Nashik	**Shri. Madhav Palshikar** Associate Professor School of Computer Science Y.C.M.Open University Nashik	**Dr. P.V. Suresh** Director School of Computer and Information Sciences I.G.N.O.U. New Delhi
Dr. Pundlik Ghodke General Manager R&D, Force Motors Ltd. Pune.	**Dr. Sahebrao Bagal** Principal, Sapkal Engineering College Nashik	**Dr. Madhavi Dharankar** Associate Professor Department of Educational Technology S.N.D.T. Women's University, Mumbai
Dr. Urmila Shrawankar Associate Professor, Department of Computer Science and Engineering G.H. Raisoni College of Engineering Hingana Road, Nagpur	**Dr. Hemant Rajguru** Associate Professor, Academic Service Division Y.C.M.Open University Nashik	**Shri. Ram Thakar** Assistant Professor School Of Continuing Education Y.C.M.Open University Nashik
Mrs. Chetna Kamalskar Assistant Professor School of Science and Technology Y.C.M.Open University, Nashik	**Smt. Shubhangi Desle** Assistant Professor Student Service Division Y.C.M.Open University Nashik	

Writer/s	**Editor**	**Co-ordinator**	**Director**
1. Prof.Vaibhav Dabhade Assistant Professor Dept. of Computer Engineering, METs BKC IOE, Nashik 2. Prof. Tushar Kute Assistant Professor Researcher, Computer science MITU Skillologics, Pune 3. Prof. Ankur Shukla Assistant Professor Fergusson College, Pune	Mr.Kunal Ugale Lead Engineer,Fidelity National Information Services, Pune	Ms. Monali R. Borade Academic Co-ordinator School of Computer Science, Y.C.M. Open University, Nashik	**Dr. Pramod Khandare** **Director** School of Computer Science, Y.C.M. Open University, Nashik

Production

Course Objectives:

- To learn and get insights of the Linux operating system.
- To understand the duties of a system administrator.
- To learn the installation of Linux installation process.
- To learn about the Linux command line and Linux software installation process.
- To learn the techniques of Linux administration.
- To understand the TCP/IP networking in Linux with files systems.
- To learn the DHCP and DNS configuration on Linux.
- To understand the Microsoft network, mail server and web servers with iptables.

Learning Outcome:

Student will be able to-

- Get detailed insights of the Linux operating system.
- Understand the duties of a system administrator.
- Learn the detailed installation of Linux installation process.
- Use the Linux command line and learn Linux software installation process.
- Understand and use the techniques of Linux administration.
- Learn the TCP/IP networking in Linux with file systems.
- Set up DHCP and DNS configuration on Linux.
- Able to handle Microsoft network, mail server and web servers with iptables.

Unit No. & Name	Details	Counseling Sessions	Weightage
Unit 1 Introduction to Linux	**Introduction to Linux**: Open Source and Red Hat, Origins of Linux, GNU & Linux Distributions, Versions of Linux, Architecture of Linux. **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.	3	10
Unit 2 Installation of Redhat Linux	Installation of Redhat Linux on Virtual Machine, Understanding Partitions of Linux, Booting and shutting down Linux, Understanding Boot loaders: GRUB & LILO, Bootstrapping, Init process, rc scripts, Enabling and disabling services. Different Run levels in Linux, Understanding Linux file system structure.	4	10
Unit 3 Using Command Line and Managing Software	**Command Line:** Working with the Bash Shell, Working with basic linux command, Working with advanced linux commands, Working with Directories, Piping and Redirection, Finding Files, Using Vi Editor **Managing Software:** Understanding RPM,	5	10

	Understanding Meta Package Handlers, Creating Your Own Repositories, Managing Repositories, Installing Software with Yum, Querying Software, Extracting Files from RPM Packages.		
Unit 4 Working with Users, Groups and Permissions	Managing Users and Groups, Commands for User Management, Managing Passwords, Modifying and Deleting User Accounts, Configuration Files, Creating Groups, Using Graphical Tools for User, and Group Management, Using External Authentication Sources, the Authentication Process, sssd, nsswitch, Pluggable Authentication Modules, Managing Permissions, the Role of Ownership, Basic Permissions: Read, Write, and Execute, Advanced Permissions, Working with Access Control Lists, Setting Default Permissions with umask, Working with Attributes.	2	10
Unit 5 TCP/IP Networking and Network File System	**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.	4	10
Unit 6 Configuring DNS and DHCP	Introduction to DNS, The DNS Hierarchy, DNS Server Types, The DNS Lookup Process, DNS Zone Types, Setting Up a DNS Server, Setting Up a Cache-Only Name Server, Setting Up a Primary Name Server, Setting Up a Secondary Name Server, Understanding DHCP, Setting Up a DHCP Server	3	10
Unit 7 Connecting to Microsoft Networks and Setting up a Mail Server	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 Setting up a Mail Server: Using the Message Transfer Agent, the Mail Delivery Agent, the Mail User Agent, Setting Up Postfix as an SMTP Server, Working with Mutt, Basic Configuration, Internet Configuration, Configuring Dovecot for POP and IMAP	3	10

Unit 8 Securing Server with iptables and Configuring Web Server	Securing Server with iptables: Understanding Firewalls, Setting Up a Firewall with system-config-firewall, Allowing Services, Trusted Interfaces, Masquerading, Configuration Files, Setting Up a Firewall with iptables, Tables, Chains, and Rules, Composition of Rule, Configuration Example, Advanced iptables Configuration, Configuring Logging, The Limit Module, Configuring NAT Configuring a Web Server: introducing Apache, Configuring Apache, Implementing SSI, Enabling CGI, Enabling PHP, Creating a Secure Server with SSL	3	10
	Revision and Practice	3	
		30	80

Reference Books:

1. Linux kernel by linus kernel

2. Red hat Linux Networking and System Administration, Terry Collings and Kurt Wall, wiley pub.

3. Unix the ultimate guide by sumitabha das.

4. Advanced programming in the Unix environments. W.R. Stevens, O'Reilly Media,

Unit I

Introduction to Linux

Linux is a fast and stable open source operating system for personal computers (PCs) and workstations that which has professional Internet services, extensive development tools, fully functional graphical user interfaces (GUIs), and a massive number of applications ranging from office suites to multimedia applications. It was developed in the early 1990s by Linus Torvalds, as a research project. Linux does many of the same functions as Unix, Macintosh, Windows. Linux is having more power and flexibility, along with it is also freely available. Most general purpose operating systems, like Windows, began their development within the scope of small, restricted PCs, which have only recently become more versatile machines.

Linux is like a general purpose version of the Unix operating system that has been used for many years on mainframes and minicomputers and is now it is the system of choice for network servers and workstations. Linux has integrated the speed, efficiency, scalability, and flexibility of Unix operating system to our personal computers, taking benefit of all the abilities that a computer can now supply.

Linux internally contains the operating system program, known as the kernel, which is the part of originally developed by Linus Torvalds operating system. It has always been distributed with a lsrge number of software applications, including network servers and security programs, office applications as well as development tools. Linux is now the prime part of the open source software movement, in which independent programmers connected together to provide free, high-quality software to all kind of users. It has become the premier platform for open source software, much of it initiated by the Free Software Foundation's (FSF) GNU project.

Many open source software develop many organization's software are bundled into Linux operating system. Today, thousands of open source applications are available for Linux from sites like SourceForge, Inc.'s sourceforge.net, K Desktop Environment's (KDE's) kde-apps.org, and GNU Network Object Model Environment's (GNOME's) gnomefiles.org, The Document Foundations (LibreOffice). Most of these softwares are also integrated into the distribution repository, using packages that are distribution amenable. All of the Linux distributions include fast, efficient, and stable Internet servers, such as the web, File Transfer Protocol (FTP), and Domain Name Service (DNS) servers, along with proxy servers, news servers, and mail servers. In other words, Linux has everything we need to set up, support, and maintain a fully functional computer network.

With some graphical user interfaces like GNOME and KDE, Linux also provides GUIs with that same level of flexibility and power. Which includes Unity, Cinnamon, Mate, LXDE etc? Linux enables us to choose the interface that we want and with customization of adding panels, applets, virtual desktops, and menus, all with full drag-and-drop capabilities as well as Internet-aware tools and utilities.

The Linux is free of cost, including the network servers, GUI desktops and all software bundle. Unlike the official Unix, Linux is distributed freely under a GNU general public license (GPL) as specified by the Free Software Foundation (FSF), making it available to everyone who wants to use it. GNU (the recursive acronym stands for "GNUs Not Unix") is a project started and managed by the Free Software Foundation (FSF) to provide free software to users, programmers,

and developers. Linux is copyrighted, not in public domain. However, a GNU public license has much the same outcome as the software's being in the public area. The GNU general public license is designed to guarantee Linux remains free and, at the same time, standardized with all the characteristics. Linux is technically the operating system kernel, which forms of core of operating system and responsible for all prime operations like process management, file management, IO management and memory management. The power and stability have made Linux an operating system of choice as network servers.

Unlike most other operating systems, the Unix was developed in a research and academic surroundings like universities, research laboratories, data centres, and large enterprises. Its development has created a parallel world of operating system to the entire computer and communications revolution over the past several years. Computer professionals in those days often developed new computer technologies on Unix, such as those developed for the Internet applications. Although being a intelligent system, Unix was developed from the beginning to be flexible. The Unix system itself can be easily modified to make different versions. As a matter of fact, many different vendors keep various official versions of Unix OS. IBM, Sun, and HP Hewlett-Packard all deal and maintain their own versions of Unix.

In essence, Linux is publicly licensed and free—and reflects the deep roots Unix has in academic institutions, with their awareness of public service and assistance. Linux is a top-rate operating system accessible to all, free of charge.

Operating Systems and Linux

An operating system is a program that manages computer hardware and software for the user. Operating systems were originally designed to execute repetitive hardware tasks, which cantered around managing files, running programs, using memory and receiving commands from the user. We interact with an operating system using a user interface, maybe command based of GUI based, which permits the operating system to receive and interpret instructions given by the user. We need only to send an instruction to the operating system to perform a work, such as reading from a file or printing a document to printer. An operating system's user interface can be as easy as entering commands on a line or as complicated as selecting menus and icons on the desktop.

An operating system also manages software applications installed or created in the system. In order to perform different operations, such as editing documents or performing calculations, we need particular software applications. The editor is an example of a software application that enables us to edit a document, making changes, saving it and adding new text etc. The editor itself is a software program containing instructions to be executed by the computer. For the program to be used, it must first be loaded into computer memory called RAM, and then these instructions will be executed. The operating system also controls the loading and execution of all programs, including any software applications installed in the system. When we want to use an editor, simply send an instruction to the operating system to load the editor software application and execute it. File management, program management, memory management and user interaction with the system are traditional characteristics common to all the operating systems. Linux, like all versions of Unix, adds two more characteristics, which are not completely present in other operating systems.

Linux is truly multi user and multitasking system. Being a multitasking system, we can ask the system to perform several tasks or processes at the same time concurrently. While one task is being done, we can work on another. For example, we can edit a file while another file is being printed on printer and a song is getting played background. We do not have to wait for the other

file to finish printing before you edit, while songs are getting played continuously. As Linux is a multi user system, several users can log in to the system at the same time, each interacting with the system through his or her own terminal. This inherent feature in not available in other operating systems.

Linux shares the system's flexibility, a flexibility stemming from Unix's research origins. The Unix was developed by Ken Thompson with Dennis Ritchie at AT&T Bell Laboratories in the early 1970s, the Unix system incorporated many new developments in operating system design. Basically, Unix was designed as an operating system for researchers and developers. One major goal was to make a system that could support the researchers' changing demands over time. In order to do this, Ken Thompson had to design a system that could deal with many different kinds of operations. Like Unix, Linux has the benefit of being able to deal with the mixture of operations any user may aspect. The user is not confined to controlled and rigid interactions with the operating system. Instead, the operating system is thought of as making a set of highly effective tools accessible to the user. This user oriented philosophy means we can assemble and program the system to meet our specific requirements. With Linux, the operating system becomes operating surroundings.

History of Unix and Linux

As a flavor of Unix, the history of Linux naturally begins with the development of 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 (MIT) 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.

Unix

In 1969, Ken Thompson, Dennis Ritchie, and the researchers at AT&T Bell Laboratories created the Unix operating system, containing many of the characteristics of the MULTICS research project. They tailored the system for the requirements of a research environment, designing it to run on minicomputers. From its origin, Unix was an cheap and economic multi user and multitasking operating system.

As ,the Unix OS became favourite at Bell Labs as more and more researchers started using the system. In 1973, Dennis Ritchie collaborated with Ken Thompson to rescript the programming code for the Unix system using C programming language. Unix gradually grown from one person's tailored design to a standard software product distributed by many different vendors, such as Novell and IBM. Initially, Unix was treated as a research project. The first versions of Unix were distributed free to the computer science departments of many top rated universities. Throughout the 1970s, Bell Labs began supplying official versions of Unix and licensing the systems to different kind of users. One of these users was the computer science department of the University of California, Berkeley. The Berkeley added many new characteristics to the system that later became standard. In 1975 Berkeley released its own version of Unix, known by its distribution arm, Berkeley Software Distribution (BSD). This BSD version of Unix became a major challenger to the AT&T Bell Labs version. AT&T developed various research versions of

Unix, and in 1983 it released the first commercialized version, called System 3. This was later followed by System V, which became a supported commercial software product development.

At the same time, the BSD version of Unix was developing through various releases. In the late 1970s, BSD Unix became the base of a research project by the Department of Defense's Advanced Research Projects Agency (DARPA). As the result, in 1983, Berkeley released a powerful version of Unix called BSD R4.2. This release was included with sophisticated file management as well as networking characteristics based on Internet network protocols. The same protocols are now used for the Internet and applications. BSD release 4.2 was widely distributed and adopted by several vendors, such as Sun Microsystems as well.

In the mid-1980s, two competing standards come out, one based on the AT&T version of Unix and the other based on the BSD version. AT&T's Unix System Laboratories developed System V release 4. Several other organizations, such as IBM and Hewlett-Packard, accepted the Open Software Foundation (OSF) to create their own standard version of Unix. Two commercial standard versions of Unix existed then—the OSF version and System V release 4.

Linux

The Linux was designed specifically for Intel-based PCs. It started out at the University of Helsinki as a personal project of a computer science student named Linus Torvalds. At that time, students were making use of a program called Minix, which highlighted different Unix characteristics. The Minix operating system was created by Professor Andrew Tanenbaum and widely distributed across the Internet to students around the world. Linus's intention was to create an effective personal computer flavor of Unix for Minix users. It was named Linux by the combination of Linus + Minix, and in 1991, Linus released kernel version 0.11. Linux was widely distributed over the Internet, and in the following years, other programmers cultured and added to it, integrating most of the applications and characteristics now found in standard Unix systems. All the major window managers have now been ported to Linux operating system.

Linux is having all the networking tools, such as FTP support, web browsers, and the whole range of network services such as email, the domain name service (DNS), and dynamic host configuration (DHCP), along with FTP, web, and print servers etc. It is also having a complete set of program development utilities, such as C and C++ compilers and debuggers. Given all its characteristics, the Linux operating system remains little, stable, and rapid. In its simplest format, Linux can run effectively on only 2MB of RAM memory also. Although Linux has developed in the free and open environment of the Internet, it matches to official Unix standards. Due to the development of Unix versions in the previous years, the Institute of Electrical and Electronics Engineers (IEEE) developed an independent Unix standard for the American National Standards Institute (ANSI). This new ANSI-standard Unix is called as the Portable Operating System Interface for Computer Environments a.k.a. POSIX. This standard defines how a Unix-like system requirements to operate, with specifying information like system calls and interfaces.

The POSIX contains a universal standard for all Unix versions that must tally. Many popular versions of Unix are now POSIX - compatible. Linux was developed from the start as per the POSIX standard. Linux also adheres to the Linux file system hierarchy standard (FHS), which defines the location of files and directories in the Linux file structure.

Linux development is now under the control of The Linux Foundation (linux-foundation.org), which is a combination of The Free Standards Group (FSG) and Open Source Development Labs

(OSDL). This is the group that Linus Torvalds works for the development of new Linux versions. Which of these Linux kernels are releases are now available at kernel.org website.

Overview of Linux

The Linux is basically divided into three major components: the kernel, the environment UI, and the file structure. The kernel is the central system program that runs programs and manages hardware devices, like disks and printers. The environment provides an interface for the user maybe command line or GUI based. It gets commands from the user and sends those commands to the Linux kernel for final execution. The file structure manages the way files are stored on a storage device, such as a disk. The files are organized into directories. Each directory may contain any number of sub-directories, each holding files. Together, the kernel, the environment, and the file structure form the general operating system structure. With the help of these three, we can run programs, manage files, and interact with the computer system.

An environment provides an interface between the kernel and system user. This interface interprets commands entered by the user and sends them to the Linux kernel. Linux provides various kinds of environments: desktops, window managers, and command line shells. Each user on a Linux system has his or her own system user interface. Users can adapt their environments as per their own special requirements, whether they be command line shells, window managers, or desktop. So, for the user, the operating system functions more as an operating environment, which the user can have the control with.

In Linux, files are organized into directories, just like in Windows OS. The entire Linux file system is one big interconnected set of directories, each containing files. Some directories are standard directories reserved for the operating system's use. We can create our own directories for our own files, as well as easily move files from one directory to another directory. We can even move entire directories and share directories and files with other users on our system. With the help of Linux, we can also fit permissions on directories and files, allowing others to access them or restricting access to ourselves only. The directories of each user are, ultimately connected to the directories of other users. The directories are configured into a hierarchical tree structure pattern, beginning with an initial / root directory. The remaining directories are ultimately derived from this first root directory. With many kinds of interfaces, Linux now has a completely integrated Graphical User Interface. We can perform all of our Linux operations entirely from any one these interfaces. Now all Linux desktops are fully operational supporting drag-and-drop operations, enabling us to drag icons to our desktop and to set up our own menus on an Applications panel. These rely on an underlying X Window System, which means as long as they are both installed on our system; applications from one can run on the other desktop. The desktops are particularly helpful for documentation, news, and software you can download for those desktops. They can run any X Window System program, as well as any cursor-based program also, which were designed to work in a shell environment. A great many applications are written just for those desktops and included with our distributions. Linux desktops are now having complete sets of Internet tools, along with editors and graphics, multimedia, and system applications.

Open Source Software

Linux was developed as a collaborative open source development over the Internet, so no company or institution controls the Linux. Software developed for Linux also reflects this

background. The development often takes place when Linux users decide to work on a project altogether. The software is posted at an Internet site like sourceforge, and any Linux user can then access the site and download the software. Linux software development has always operated in an Internet environment and is global in scope, enlisting programmers from around the world. The only thing you need to start a Linux-based software project is a website.

Most of Linux software is developed as open source software a.ka. FOSS (Free and Open Source Software). It means that the source code for a software application is freely distributed along with the application. Programmers over the world can make their own contributions to a software package's development, they also modifies and corrects the source code. Linux is the open source operating system as well. The source code of Linux is included in all its distributions and is freely available on the Internet for downloading. Many major software development efforts are also open source projects, as are the KDE, GNOME desktops, Cinnamon, LXDE, Mate Desktops, LibreOffice along with most of their applications. The Firefox and Google Chrome web browser package has also become open source, with its source code freely available on web. The LibreOffice office suite supported by The Document Foundation is an open source project based on the OpenOffice office suite. Many of the open source applications that run on Linux have located their websites at SourceForge (sourceforge.net), which is a hosting website designed generally to support open source software projects.

Open source software is protected by many public licenses. These forbid commercial companies from taking control of open source software by adding a some modifications of their own, copyrighting those alteration, and selling the software as their own product. The most popular public license is the GNU GPL provided by the Free Software Foundation (FSF). Linux is distributed under this license. The GNU GPL retains the copyright, freely licensing the software with the requirement that the software and any modifications made to it forever be freely available. Other public licenses are also been created to assist the demands of different kinds of open source projects. The GNU lesser general public license (LGPL) lets commercial applications to use the GNU licensed software libraries. The qt public license (QPL) lets open source developer's use the Qt libraries essential to the KDE desktop.

Linux Distributions

Although there is only one standard version of Linux, there are actually several different distributions available. Different organizations, institutions and groups have packaged Linux and Linux software in slightly different ways. Each organization or group releases the Linux package, usually on a DVD-ROM of in iso package format. Later releases may add updated versions of programs or new software. Some of the more popular distributions are Ubuntu, Mint, OpenSUSE, Fedora, and Debian. Linux kernel is centrally distributed through kernel.org repository. All of these distributions use this same kernel, although it may be configured separately.

Linux has spread with a great variety of distributions. Many of them aim to provide a comprehensive solution providing support for any and all kind of tasks. These include distributions like OpenSUSE, Red Hat, Debian, Mint and Ubuntu. Some are variations on other distributions, like Centos, Fedora which are based on Red Hat Enterprise Linux, and Ubuntu, Kali Linux, Mint which derives from Debian Linux. Others distributions have been developed for more specialized tasks or to support certain characteristics. The distributions like Debian provide cutting edge developments projects. Some distributions provide more commercial versions, usually bundled with commercial applications such as databases or secure servers. Certain companies like Red Hat and Novell provide a commercial distribution that agrees to a supported

free distribution. The free distribution is used to develop new characteristics, like the Fedora Project for Red Hat.

Currently, the website https://distrowatch.com lists numerous Linux distributions. It contains the detailed information about all the Linux distributions, their software and utilities available under FOSS licenses.

Linux Package Management

One of the things that sets Linux apart from other operating systems is the way of software's installation and management. Traditionally when we wanted to install software on the Windows operating system we would searches the software on internet, download the software, and install it. These are steps that the end user has to perform sequentially.

In order to install software on a Linux system we use the package manager that comes with every distribution. For installation a new piece of software we search for it and install it from the operating system itself. The package manager of distribution takes care of downloading the desired software with any required dependencies and then installs all of the components in the system. Package managers not only control applications, but they can also manage the operating system itself. A package manager of Linux can updates and upgrades the system and all of its installed applications to latest available versions.

Software and applications are bundled into packages and Linux distributions are categorized by these package types. The three basic types of packages are Debian (.deb), RedHat Package Manager (.RPM), and other distributions.

Debian Based Linux Distributions

The .deb package type was created in 1993 for the Debian Linux distribution for first time. Debian is one of oldest Linux distributions and it's a very popular choice on which new distributions are based. Popular distributions that use .deb packages include:

- Debian
- Ubuntu
- Kali Linux
- Linux Mint
- SteamOS
- Bodhi Linux
- Raspbian OS

The Debian Linux

In 1993 developer Ian Murdock announced a new Linux distribution that was to be developed openly with the GNU's open source philosophy. Ian gave his distribution the name Debian which is a combination of his girlfriend's name Debra and his own name i.e. Debra + Ian = Debian. It was a small project as first, but today Debian is one of biggest open source projects in existence.

Debian is a universal operating system and supports almost all CPU architectures including arm 32 bit and 64 bits and it is a very popular in the server space. Although Debian is well known for

rock solid stable software, there are variants available. These variants are Debian old stable, stable, testing, unstable and experimental. As we go from old stable to experimental, we get newer and less stable software. As for package management, Debian uses the package manager, apt called as advanced packaging tool.

Ubuntu

It was announced in 2004 for first time. Ubuntu is based on Debian unstable Linux. It is the most widely used and most popular Linux distribution today. It's also the Linux distribution encircled by the most controversies. Ubuntu started with the Gnome desktop, but a few years ago Ubuntu developed its own desktop environment called as Unity. The Ubuntu installation process is very easy and thus is popular with those new to Linux operating system. Ubuntu also uses apt and its graphical fronted Ubuntu Software Center for package management. The Ubuntu has released many variants of its own named as Kubuntu, Lubuntu, Xubuntu, Edubuntu, MythBuntu, Ubuntu Server, Ubuntu Mate etc.

Linux Mint

Linux Mint is one more popular distribution based on Ubuntu. Linux Mint started out simply being Ubuntu with pre-installed multimedia codecs and proprietary drivers. However, it has since grown and is a very popular alternative available to Ubuntu. According to distrowatch.com, Linux mint is among top five Linux operating systems preferred by users.

Raspbian

The Raspbian is a Debian-based computer operating system for Raspberry Pi microcomputer. There are several versions of Raspbian available including Raspbian Buster and Raspbian Stretch. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers. It is compatible to all flavors of Raspberry Pi starting from 0 to 4. Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012. This operating system is still under active development. It is highly optimized for the Raspberry Pi line's low-performance ARM CPUs.

RPM Based Linux Distributions

RedHat created the rpm (Redhat Package Management) package format for use in its distribution. Popular RPM based distributions include:

- RedHat Enterprise Linux (RHEL)
- CentOS
- Fedora
- OpenSuse
- Mageia

Fedora

Fedora is the upstream open source version of the commercial RedHat Enterprise Linux distribution, or RHEL for short. What makes Fedora special is it uses newer technology and packages from the open source world than RHEL. Fedora, like RHEL, uses the yum and dnf package managers.

OpenSuse

The OpenSuse Linux started out a German translation of Slackware Linux, but eventually grew into its own distribution. The OpenSuse is known for the KDE desktop and most important is stability. For package management, OpenSuse uses zypper and its graphical fronted, as well as the Yast software center.

Mageia

Mageia Linux is a fairly new Linux distribution that is based on Mandrake Linux. Mageia is easy to install and easy to use. It utilizes urpmi and drakrpm for package management.

Other Linux distributions

Arch Linux

The Arch Linux uses pkg.tar.xz package formats and has it's own package manager called pacman. Arch doesn't get available with a graphical installer and the whole installation process is done via a terminal only. This can be intimidating for new Linux users. The main philosophy behind Arch is KISS – keep it simple, stupid. The Arch has been forked in some popular beginner-friendly distributions like Manjaro Linux.

Slackware Linux

The Slackware was founded in 1992 by Patrick Volkerding. This is the oldest Linux distribution in use today. Slackware does not have a package manager and all the software is compiled by the system administrator or normal users of the system. Slackware packages are simply source code. If we really want to learn a lot about the Linux really works, we may use Slackware.

Gentoo Linux

Gentoo Linux is based on the portage package management system. It can be difficult to install and can even take as long as a couple of days to complete the entire installation process. The advantage of such an approach is that the software is built for the specific hardware that it will be running on. Like Slackware, Portage uses application source code. If we like the idea of Gentoo, but are looking for something beginner friendly, we may try the Sabayon Linux.

Graphical User Environments

There are many desktop managers available for Linux which has a beautiful Graphical User Interface. While Microsoft Windows users only have one desktop manager, Linux users can chose their desktop environment. The desktop environment, or the graphical user interface (GUI), is what is displayed on the screen of computer. It is how the system looks. Popular desktop managers include KDE, Gnome, Xfce, Unity, Cinnamon, Mate and LXDE.

KDE -

The KDE of K Desktops Environment was created in 1996 and is probably the most advanced desktop manager on the market that time. KDE includes several applications that every user requirements for a complete desktop environment by default. The KDE has some characteristics that are not present in other desktop managers. The KDE workspace is called as Plasma. Combine Plasma with the other KDE applications and you get what is called the KDE software compilation or KDE SC for short.

Popular distributions that use KDE include:

- OpenSuse
- Kubuntu
- Mageia
- Slackware
- Linux Mint

Gnome

Gnome is a desktop manager created for the community and by the community. This is a great example of how the open source community actually works. Gnome desktop can simply be expanded with the use of plug-ins added in it. It doesn't require a lot of resources and can be a great choice for older and slower hardware. Popular distributions that use Gnome include:

- Debian
- OpenSuse
- Fedora
- CentOS
- Ubuntu 18.04 onward

Cinnamon

The Cinnamon is a fork of the Gnome desktop manager and it is developed by the Linux Mint community. It recreates the look of Gnome 2 with a modern touch added to it. The minimum system requirements for Cinnamon are the same as they are for Gnome. Now many variants of Linux Mint comes with default Cinnamon installed in it.

XFCE

XFCE is an excellent choice for older computers. Light and fast are XFCE's two biggest characteristics. The system needs are a measly 300Mhz CPU and 192Mb of RAM. Popular distributions that use XFCE include:

- Debian
- Xubuntu
- Fedora
- OpenSuse

LXDE

The LXDE is an another fast and light desktop manager. Based on the OpenBox windows manager, LXDE is suitable for old computers too. Popular distributions using LXDE include:

- Lubuntu

- Debian
- OpenSuse
- Linux Mint
- Raspbian

Unity Desktop

Unity was developed by Canonical for their Ubuntu Linux distribution for the first time. Till date, Ubuntu is the only distribution that uses Unity. Unity requires greater hardware resources than most graphical environments. We'll need a 1 GHz CPU and 1Gb RAM in order to get Unity to work. With those specs, Unity will be so slow that it's almost unusable. For Unity, the more RAM and CPU, the better. Till Ubuntu 17.10, Unity was the default desktop manager present in the Ubuntu flavors. Now, they are switched to GNOME.

Architecture of Linux

The Linux Operating System's architecture primarily has following components:

- The Kernel
- Hardware layer
- System library
- Command Shell
- System utility

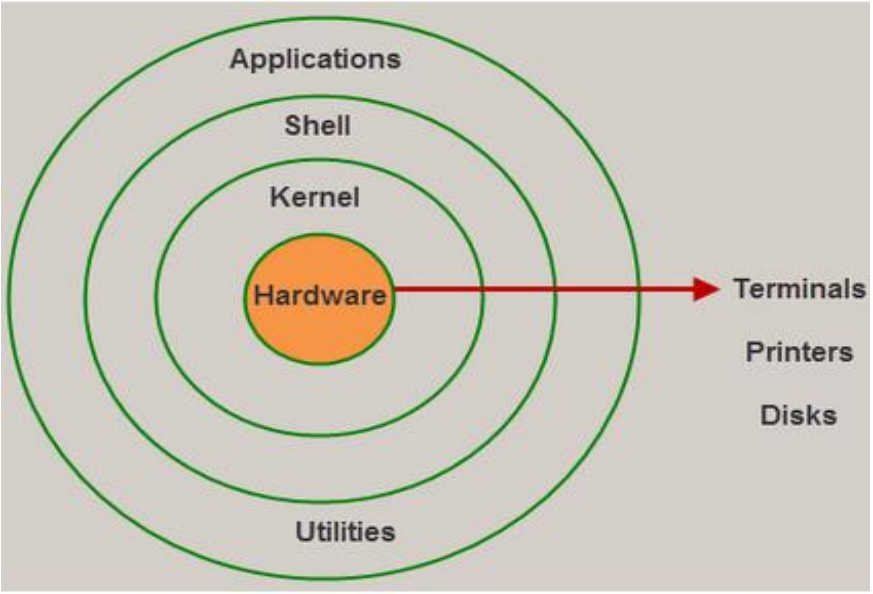


Fig. 1.1 Architecture of Linux

Kernel

The kernel is the core part of operating system. It is responsible for all the major activities of the Linux operating system including process management, memory management, device management and file management. This part of operating system contains different modules and it interacts directly with the underlying hardware. The kernel offers the required abstraction to cover up application programs or low-level hardware details to the system. Following are the types of Kernels:

- Monolithic Kernel
- Micro kernels
- Exo kernels
- Hybrid kernels

System Libraries

System libraries are special type of functions that are used to implement the functionality of the operating system and do not need code access rights of Linux kernel modules.

System Utilities

The System Utility programs are liable to do individual and specialized-level tasks by the user or by the operating system.

Hardware

The Hardware layer of the Linux operating system contains peripheral devices such as Memory-RAM, Hard Disk Drives, CPU i.e. Microprocessor.

Shell

The shell is an interface between the user and the kernel, and it provide services of the kernel. It receives commands from the user and executes kernel's functions as per. The command Shell is present in different types of operating systems, which are classified into two categories: command line shells and graphical shells.

The command line shells provide a command line interface with a set of commands, while the graphical line shells provide a graphical user interface with more interactivity. Both shells perform operations, but the graphical user interface shells perform slower than the command line interface shells due to memory utilization.

There are five types of shells present in Linux:

- Korn shell
- Bourne shell
- C shell
- Bourne Again Shell (BASH)
- Turbo C Shell

Features of Linux

The main characteristics of Linux operating system are as described below.

Portable: Linux operating system can work on different types of hardware as well as Linux kernel supports the installation of any kind of hardware platform including all microprocessor architectures.

Open Source: The source code of Linux operating system is freely available on the internet and, to enhance the ability of the LINUX operating system, many teams work in collaboration to develop it. All the software on Linux are free!

Shell: Linux operating system offers a special interpreter program, that can be used to execute commands of the OS called as command shell. It can be used to do several types of operations like call application programs, process, IO and memory operations. Many tasks which can't be done by GUI, but can be executed by command shell operations.

Security: Linux offers user security systems using authentication characteristics like encryption of data or password protection or controlled access to particular files. So it does not have any access to the viruses and several malware that are found on Windows operating system.

Multiprogramming: Linux operating system is a multiprogramming system, which means multiple applications can run at the same time concurrently. It has inbuilt support for creation of processes as well as threads too.

Multi-user: Linux operating system is a multi-user system, which means, multiple users can access the system resources like RAM, Memory or Application programs at the same time. Any number of users can be created in this operating system.

Hierarchical File System: Linux operating system affords a standard file structure in which system files or user files are arranged. For general purpose use, EXT file system is used in Linux. Currently, EXT4 version of file system is available to use.

System Administrator

Every computer in the world has a system administrator associated with it. It may be that the maximum system administrators are likely those who decided as to what software and peripherals devices would be packaged with the machine. That status remains because the maximum users who get computers for use probably do small to change in the default values. But the minute the user makes some changes in applications and decides what software and applications to run he becomes a system administrator.

High duties bring with it some responsibilities. No one whose computer is connected to the Internet, for instance, has been immune to the effects of poorly administered systems, as demonstrated by the Distributed Denial of Service (DDoS) attacks and e-mail macro virus attacks that have shaken the online world in recent years. The scope of these actions of computer vandalism would have been greatly decreased if system administrators had a better understanding of their responsibilities.

Linux system administrator is more likely to understand the requirements of active system administration than are those who run whatever comes on the computer, assuming that things came from the factory are properly configured. By its very nature as a modern and multiuser operating system, Linux requires a large degree of administration larger than that of less robust home market systems. It interprets that even if we are using a single machine connected to the Internet by a dial-up modem — or not even connected at all — we have the benefits of the same

system employed by some of the biggest enterprises in the world, and will do many of the things that the software professionals employed by those companies are paid to do the same.

Administering our own system does pertain a degree of learning, but it also means that in setting and configuring our own system we gain ability and knowing that raise us above mere "computer user" status.

Linux system administrator is the person who has "root" access, which is to say the one who is the system's "suprime user" (or root user). A standard Linux user is having some limitations. But the "root" user has uninterrupted access to everything — all user accounts, their home directories, and all of 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 simply and safely through other, more specific means.

Duties of the System Administrator

Linux System Administrator is a person who has 'root' access that is 'supreme user'. It means he has exclusive right to access everything which considers all user accounts, all system configurations, home directories with all files therein, all files in system. The Administrator has following duties and responsibilities.

1. Installing and configuring server -

A server is basically a computer program that facilitates the same computer or other computer by providing some services to them. It is most important element of Modern operating system and network design. The system administrator configures server so that the most necessary server remain inaccessible. He must be aware of types of attack as well as security faults.

2. Installing using the live server installer

Most of the time it is required to install the server by using live CD / DVD. The Linux has ability to run the system live using bootable USB stick or CD/DVD. Ubuntu Server is one of the most popular server editions these days. The basic steps to install Ubuntu Server Edition are the same as those for installing any operating system. Unlike the Desktop Edition, the Server Edition does not include a GUI based installation program. The LiveServer installer uses a text-based console interface which runs on default virtual console mode. The interface can be wholly driven by the enter, up and down arrow keys with some occasional typing.

If we require at any time during the installation we can switch to a different console (by pressing Ctrl-Alt-F<n> or Ctrl-Alt-Right) to get access to a shell. Up until the point where the installation begins, we can use the "back" buttons to go back to previous screens and choose different options.

- Download the appropriate ISO file from the Ubuntu web site https://ubuntu.com/download
- Create a bootable pen drive using Start-up Disk Creator or any disk burning software on Windows.
- Boot the system from media (e.g. USB key) containing the ISO file.
- At the boot prompt we will be asked to select a language. e.g. English (US)

- From the main boot menu there are some additional options to install Ubuntu Server Edition. We can install a basic Ubuntu Server, check the installation media for defects, check the system's RAM, or boot from first hard disk.
- After booting into the installer, it will ask us which language to use.
- The installation process begins by asking for our keyboard layout. We can ask the installer to attempt auto-detecting it, or we can select it manually from a list. Later stages of the installation will require us to type ASCII characters, so if the layout we select does not allow that, we will be prompted for a key sequence to switch between a layouts that does and the one we select. The default keystroke for this is Alt + Shift.
- The installer offers the choice to install the system as a vanilla Ubuntu server, a MAAS5 bare-metal cloud rack controller or a MAAS6 region controller. If we select one of the MAAS options we will be asked for some details.
- The installer configures the network to run Dynamic Host Control Protocol – DHCP on each network interface. If this is not adequate to get access to the internet we should configure at least one interface manually. Select an interface to configure it.
- If the Ubuntu archive can only be accessed via a proxy in our environment, it can be entered on the next screen. Leave the field blank if it is not needed.
- We can then select to let the installer use a whole disk or configure the partitioning manually. The first disk we make a partition on will be selected as the boot disk and have an extra partition made on it to contain the bootloader; we can move the boot partition to a different drive with the "Select as boot disk" button. Once we move on from this screen, the installation progress will begin. It will not be possible to move back to this or previous screens and any data on the disks we have configured the installer to use will be lost. The next screen configures the initial user for the system. We can import SSH keys from Launchpad or Github but a password is still needed to be set, as this user will have root access through the sudo utility. The final screen displays the progress of the installer. Once the installation has accomplished, we will be prompted to reboot into our freshly installed system.

3. Installing and configuring application software

To ensure right execution of environment, administrator must give software which is well configured and validated. He must ensure adequate memory allotment and resolve software failure as well as dependency issues.

He must provide a set of activities to control hardware and software configuration and maintain policies in favor of users.

It is very important for the novel Linux system administrator to understand two characteristics that set Linux apart from other popular commercial operating systems: The first is the root or super user, and the second is that Linux is a multi-user operating system. Each user has (or shares) an account on the system, it may be on a separate machine or on a single machine with multiple accounts.

Every user can install some applications in their home directories separately— drive space set aside for their own files and customization — these applications are not accessible to the other users. Apart from it, if an application is to be used by more than one user, it requirements to be installed higher up in the Linux file hierarchy, which is called as a job of the system administrator

only. The administrator can even decide which users may utilize which applications by creating a "group" for that application and inscribing individual users into that group.

The location of the installation of application usually matters only if we compile the application from source code; if we use a Red Hat Package Manager (RPM) application package, it automatically goes where it should be present.

Configuration and customization of applications is to some extent at the user's discretion, but not completely. "Skeleton" configurations — administrator can regulate default configurations. For example, if there are particular forms that are used throughout our company, the system administrator would set them up or put them available by adding them to the skeleton configuration. The same applies too, in configuring user desktops and determine what applications should appear on user's desktop menus. Our company may not want the games that come with modern Linux desktops to be available to users!

4. Creating and maintaining user accounts

Any user can access his own account but administrator has access to every other user account also. He can add, modify, delete or copy any other user account. He is responsible for keeping security by providing role on a user account that define the level of access for it.

Any person cannot log on to a Linux machine. An account must be created for each user, no one but the system administrator may do this.

We might want users to select and create their own passwords, which would be easier for them to remember, but which probably would be easier for an external factor to crack or hack it. We might assign passwords, which is more secure in theory but which increases the chances that users will write them down on a handily located scrap of paper — a risk if many people have access to the area where the machine(s) is situated. We might want that users must change their passwords periodically, and we can configure Red Hat Linux to prompt users to do so.

And what to do about old accounts? Maybe someone has left the company. What will happen to his or her account? We probably don't want him or her to continue to have access to the company's network. On the other side, we don't want to simply delete the account, because it might contain some necessary data which resides nowhere else.

There are aspects of our business that make World Wide Web access required, but we don't want user's spending their working hours spending on the Web.

The administrator or his employer must set up the policies governing the related issues— if in company, preferably in writing — for the security of all concerned.

5. Adding a new user

Before we create an account for a new user at a private organization, government, or educational site, it's important that the user sign and date a copy of our local user agreement and policy statement. Users have no specific reason to want to sign up the policy agreement, so it's to our benefit to secure their signatures while we still have some leverage. We find that it takes more effort to secure a signed agreement after an account has been free. If our process allows for it, have the paperwork precede the creation of this account. The process of adding a new user consists of several steps required by the system, two steps that establish a useful environment for the new user, and several extra steps for our own convenience as an administrator.

Required:

- Have the new user sign your policy agreement?
- Edit the **passwd** and **shadow** files to define the user's account.
- Add the user to the /**etc/group** file (not generally required, but fine).
- Set the initial password. It follows all basic rules to be a strong password.
- Create, **chown**, and **chmod** the user's home directory. It is required to setup the owner and mode of use for them.
- Configure roles and permissions for the user's accounts.

For the user:

- Copy default startup files to the user's home directory. So, these will be activated every time user starts the system.
- Set the user's mail home and establish mail aliases as per the company's requirements.

For you:

- Verify that the account is set up properly or not.
- Add the user's contact information and account status to our database.

This list cries out for a script or tool, and fortunately each of our example systems provides one in the form of a **useradd** or **adduser** command.

We must be the supreme user like root to add a user, or on AIX system, we must have User Admin privileges. This is a perfect place to use **sudo** i.e. super user do operations**.**

6. Editing the passwd and group files

If we have to add a user by hand, use **vipw** to edit the **passwd** and **shadow** files. Although it is connected to vi editor of Linux, it actually uses our favorite editor as defined in the EDITOR environment variable like nano or pico. More significantly, it locks the file so that our editing and a user's password change operations do not clash.

On Solaris, and Red Hat systems, **vipw** automatically asks if we would like to edit the **shadow** file after we have edited the **passwd** file. SUSE and Ubuntu systems use **vipw -s** for this functionality.

Both HP-UX and AIX recommend that we not to edit the password file by hand, with or without **vipw** (it is not even installed on AIX), but rather use **useradd** or their do-it-all sysadmin tools **smh** and SMIT, respectively.

If the new user should be a member of more groups than just the default group specified in the **passwd** file, we must edit the **/etc/group** file and add the user's login name to each of the additional groups.

7. Setting a password

The administrator must not leave a new account—or any account that has access to a shell—without a password. It is mandatory for today's system to have a compulsory password logins.

Password complexity can be enforced with configuration files. We can set a password for the new user with-

$ sudo passwd newusername

After this command, we will be prompted for the actual password of the system because of sudo. Some automated systems for adding new users do not require us to provide an initial password. Instead, they force the user to set a password on first login. Although this feature is handy, it's a large security hole: anyone who can guess new login names (or look them up in **/etc/passwd**) can swoop down and hack the accounts before the intended users have had a chance to log in.

Commands and configuration files for user management:

Operating System	**Commands**	**Configuration files**	**Comments**
Ubuntu	useradd adduser	/etc/login.defs /etc/default/useradd /etc/adduser.conf	Applicable to all Debian based operating systems like Debian, Mint and Kali.
OpenSUSE	useradd	/etc/login.defs /etc/default/useradd /etc/default/passwd /usr/sbin/useradd.local /usr/sbin/userdel.local /usr/sbin/userdel-pre.local /usr/sbin/userdel-post.local	For local customization For local customization For local customization For local customization
Red Hat	useradd	/etc/login.defs /etc/default/useradd	Applicable to Fedora operating system also.
Solaris	useradd	/etc/default/{login,passwd} /etc/security/policy.conf	
HP-UX	useradd smh	/etc/default/useradd /etc/default/security	GUI tool, also called sam
AIX	useradd mkuser chuser rmuser SMIT	/etc/security/user /etc/security/login.cfg /etc/security/mkuser.default	Called by **useradd** Called by **usermode** Called by **userdel** GUI tool

8. Backing up and restoring files

In order to decrease the loss of data, administrator must keep backup of files and he should restore it whenever necessary. Administrator can take backup in removable media such as hard drives or tapes as protection against loss.

Before creating backup administrator must decide the following:

What is required to backup?

What is the frequency of the backup?

Until equipment becomes absolutely failure proof, and until people lose their desire to hurt the property of others for personal welfare (and, truth be known, until system administrators become perfect), there is always a need to back up important files and data so that in the event of a failure of hardware, security, or administration, the system can be up and running again with minimal disturbance. Only the system administrator can do this. Due to its built-in security characteristics, Linux may not allow users to be able even to back up their own files to pen drives.

Knowing that file backup is our job is not sufficient. We need to develop a proper strategy for making sure that our system is not vulnerable to interruption. If we have a high-capacity pen drive and several restore hard diskettes, we might make a full system backup after every few days. If we are managing a system with scores of users, it is more sensible to back up user accounts and system configuration files, from the distribution CD or DVDs.

Once we have decided what to back up, we need to decide how often to perform backups and whether we wish to keep a series of incremental backups onwards— adding only the files that have changed since the last backup — or multiple full backups, and when these backups are to be performed — do you trust an automatic, unattended process?

A strategy should be the maintenance of complete backups without ever needing to resort to them. This means promoting users to keep multiple copies of their own important files, all in their home directories, so that we are not being asked to mount a backup so as to restore a file that a user has corrupted. And if the system is stand-alone, we as our own system administrator might want to make a exercise of backing up configuration and other important files.

The chances are that even if we are working for a company, we will make these decisions — all our boss wants is a system that works perfectly, and all the time. Backing up is only half the story. We need to formulate a plan for bringing the system back up in case of the event of a failure.

At most sites, the data stored on computers is worth far more than the computers themselves. It is also much difficult to replace. Security of this information is one of the system administrator's most important and, unluckily, most tiresome tasks.

There are hundreds of fanciful and not-so-creative ways to lose your data.

- The software bugs routinely corrupt your documents.
- Users accidentally delete and removes data files by commands.
- The hackers and dissatisfied employees erase disks.
- Hardware problems and natural disasters like floods and tsunami take out our entire machine rooms.

If execution is done correctly, backups allow an administrator to restore a file system or any portion of a file system to the state it was in at the time of the last backup. Backups must be done carefully and with a strict schedule. The backup system and backup media must also be tested regularly to verify that they are working accurate. The integrity of our backup procedures directly affects our company's bottom line. Senior management needs to understand what the backups are actually able to do, as opposed to what they want the backups to do. It may be OK to lose a day's work at a university's computer science department, but it likely is not OK at a commodity trading firm.

9. Backup Devices And Media

Many failures can harm several pieces of hardware at once, so backups should be written to some sort of removable media. A good rule of thumb is to create offline backups that no single dissatisfied system administrator could destruct. Backing up one hard disk to another on the same machine or in the same data center provides little security against a server failure, although it is surely better than no backup at all. Enterprises that back up our data over the Internet are becoming more popular, but most backups are still created locally. The important data can be stored on the net clouds like Google Drive, Dropbox or Amazon Web Services.

The following sections describe some of the media that can be used for backups.

The media are presented in rough order of increasing capacity. Manufacturers like to specify their hardware capacities in terms of compressed data; they often blindly assume a compression ratio of 2:1 or more. The compression ratio also affects a drive's throughput rating. If a drive can physically write 1 MB/s to drive but the manufacturer assumes 2:1 compression, the throughput magically rises to 2 MB/s.

Optical media: CD-R/RW, DVD±R/RW, DVD-RAM, and Blu-ray

The CDs and DVDs are an attractive option for backups of small, isolated systems. CDs hold about 700MB and DVDs hold 4.7GB. Dual-layer DVDs clock in at about 8.5GB.

Drives that write these media are available for every common bus (SCSI, IDE, USB, SATA, etc.) and are in many cases are so inexpensive as to be essentially free. Now that CD and DVD prices have equilateral, there's no reason to use CDs rather than DVDs. However, we still see quite a few CDs used in the real world for reasons that are not entirely clear.

Optical media are written through a photo-chemical processing that involves the use of a laser. Although hard data on longevity has been elusive, it is believed that optical media have a considerably longer shelf life than magnetic storage media. However, the write-once versions (CD-R, DVD-R, and DVD+R) are not as long-lived as manufactured (stamped) CDs and DVDs.

Today's fast DVD writers offer speeds as rapid as —if not faster than—tape drives. The write-once versions are DVD-R and DVD+R. DVD-RW, DVD+RW, and DVD-RAM are rewritable. The DVD-RAM system has built-in defect management and is therefore more reliable than other kinds of optical media. On the other hand, it is much more expensive.

The manufacturers estimate a potential life span of decades for these media if they are properly stored. Their recommendations for proper storage include individual cases, storage at a constant temperature in the range 41°F–68°F with relative humidity of 30%–50%, no exposure to direct sunlight, and marking only with water-soluble markers. Under average conditions, a reliable shelf life of 1–5 years is probably more practical.

As borne out by numerous third-party evaluations, the reliability of optical media has proved to be especial manufacturer dependent. This is one case in which it pays to spend money on premium quality media. Regrettably, quality varies from product to product even within a manufacturer's line, so there is no safe-bet manufacturer.

A recent entry to the optical data storage market is the Blu-ray disc, whose various flavors store from 25–100 GB of data. This high capacity is a result of the short wavelength (405nm) of the laser used to read and write the disks (hence the "blue" in Blu-ray). As the cost of media drops, this technology hope to become a good solution for backups.

Portable and removable hard disks

External storage devices that connect through a USB 2.0 / USB 3.0 or eSATA port are common. The underlying storage technology is usually some form of hard disk, but flash memory devices are common at the low end (the ubiquitous "jump drives"). Capacities for conventional hard drives range from less than 250GB to over 2TB. Solid state drives (SSDs) are based on flash memory and are also currently available in sizes up to 160GB. The limit on USB flash memory devices is about 64GB, but it is growing very rapidly these days.

The lifetime of flash memory devices is mostly a function of the number of write cycles. The Mid-range drives usually last at least 1,00,000 cycles.

The main restriction of such drives as backup media is that they are normally online and so are assailable to power surges, heating overload, and tampering by malicious users. For hard drives to be effective as backup media, they must be manually unmounted or disconnected from the server. Removable drives make this task easier. Specialized "tapeless backup" systems that use disks to emulate the off-line nature of tapes are also available.

DLT/S-DLT

The Digital Linear Tape/Super Digital Linear Tape is a mainstream backup medium. These drives are reliable, affordable, and capacious too. They evolved from DEC's TK-50 and TK-70 cartridge tape drives. DEC sold the technology to Quantum, which popularized the drives by increasing their speed and capacity and by dropping their price. In 2002, Quantum acquired Super DLT, a technology by Benchmark Storage Innovations that tilts the recording head back and forth to reduce crosstalk between adjacent tracks.

Quantum now offers two hardware lines: a performance line and a value line. We get what we pay for. The tape capacities vary from DLT-4 at 800GB to DLT-4 in the value line at 160GB, with transfer rates of 60 MB/s and 10 MB/s, respectively.

Manufacturers boast that the tapes will last 20 to 30 years—that is, if the hardware to read them still exists. The drawback of S-DLT is the price of media, which runs $90–100 per 800GB tape. A bit costly for a university; perhaps not for a Wall Street investment firm.

AIT and SAIT

The Advanced Intelligent Tape is Sony's own 8mm product on steroids. In 1996, Sony dissolved its relationship with Exabyte and introduced the AIT-1, an 8mm helical scan device with twice the capacity of 8mm drives from Exabyte. Today, Sony offers AIT-4, with a capacity of 200GB and a 24 MB/s maximum transfer rate, and AIT-5, which doubles the capacity while keeping the same transfer speed.

SAIT is Sony's half-height offering, which uses larger media and has greater capacity than AIT. SAIT tapes holds up to 500GB of data and with a transfer rate of 30 MB/s. This product is most common in the form of tape library offerings—Sony's are especially popular.

The Advanced Metal Evaporated (AME) tapes used in AIT and SAIT drives with a long life cycle. They also comprise of a built-in EEPROM that gives the media itself some smarts. Software support is needed to create any actual use of the EEPROM, however. Drive and tape prices are both roughly on par with DLT.

VXA/VXA-X -

The VXA and VXA-X technologies were originally developed by Exabyte and were acquired by Tandberg Data in 2006. The VXA drives use what Exabyte describes as a packet technology for data transfer. The VXA-X products still rely on Sony for the AME media; the V series is upgradable as larger-capacity media become available. The VXA and X series claim capacities in the range of 33–160 GB, with a transfer rate of 24 MB/s.

LTO -

Linear Tape-Open was developed by IBM, HP, and Quantum as an alternative to the proprietary format of DLT. LTO-4, the latest version, has an 800GB capacity at a speed of 120 MB/s. LTO media has an estimated storage life of 30 years but is susceptible to magnetic exposure. As with most technology, the previous generation LTO-3 drives are much less expensive and are still adequate for use in many environments. The cost of media is about $40 for LTO-4 tapes and $25 for the 400GB LTO-3 tapes.

Jukeboxes, stackers, and tape libraries -

With the low cost of disks these days, most sites have so much disk space that a full backup requires many tapes, even at 800GB per tape. One possible solution for these sites is a stacker, jukebox, or tape library.

A stacker is a simple tape changer that is used with a standard tape drive. It has a hopper that you load with tapes. The stacker unloads full tapes as they are ejected from the drive and replaces them with blank tapes from the hopper. Most stackers hold about ten tapes.

A jukebox is a hardware device that can automatically change removable media in a limited number of drives, much like an old-style music jukebox that changed records on a single turntable. Jukeboxes are available for all the media discussed here. They are often bundled with special backup software that understands how to manipulate the changer. Storage Technology (now part of Oracle) and Sony are two large manufacturers of these products.

Tape libraries, also known as auto changers, are a hardware backup solution for large data sets—terabytes, usually. They are large-closet-sized mechanisms with multiple tape drives (or optical drives) and a robotic arm that retrieves and files media on the library's many shelves. As you can imagine, they are quite expensive to purchase and maintain, and they have special power, space, and air conditioning requirements.

Most purchasers of tape libraries also purchase an operations contract from the manufacturer to optimize and run the device. The libraries have a software component, of course, which is what really runs the device. Storage Technology (Oracle), Spectra Logic, and HP are leading manufacturers of tape libraries.

Hard disks

The decreasing cost of hard drives provides disk-to-disk backups an fascinating option to believe. We do not have to duplicate one disk to another within the same physical machine, hard disks can be a good, low-cost solution for backups over a network and can dramatically decrease the time required to restore large datasets.

One obvious problem is that hard disk storage space is limited and must in time be reutilized. However, disk-to-disk backups are a fabulous way to defend against the accidental deletion of files. If we maintain a day-old disk image in a well known place that's shared over NFS or CIFS, users can recover from their own mistakes without affecting an admin.

Remember that on-line storage is usually not sufficient security versus malicious attackers or data center equipment failures. If we are not capable of actually storing our backups off-line, at least shoot for geographic diversity when storing them on-line.

Internet and cloud backup services

Service providers have recently started to offer Internet-hosted storage solutions. Rather than provisioning storage in our own data center, we lease storage from a data cloud provider. Not only does this approach provide on-demand access to nearly unlimited storage, but it also gives we a simple path to store data in multiple geographic locations.

Internet storage services start at 10/GB/month and get more costly as we add characteristics. For example, some providers let us choose how many duplicate copies of our data will be stored. This pay-per-use pricing allows us to choose the reliability that is suitable for our data and fund.

Internet backups only work if our Internet connection is rapid enough to transmit copies of our changes every night without bogging down "actual" traffic. If our organization handles huge amounts of data, it is unlikely that we can back it up across the Internet. But for small organizations, cloud backups can be a perfect solution since there is no up-front cost and no hardware to buy involved. Any sensitive data that transits the Internet or is stored in the cloud must be encrypted for security.

10. Monitoring and tuning performance

Following are the responsibilities of system administrator:

- Monitoring and tuning of performance is necessary for Linux to work more promptly.
- Administrator must denote system bottleneck and should resolve these.
- Administrator can use system tools to gain performance; he can determine when hardware need to get upgraded.
- He should identify primarily sign of failure.

On modern stand-alone systems, Linux is pretty fast, and if it isn't, there's something wrong — something that is up to the system administrator to fix it. We might have a number of people using the same file server, mail server, or other shared machine, in which small advancement in system performance can mean a great deal.

System tuning is a continuous process carried by a variety of monitoring tools and techniques. Some performance decisions are made at installation time, while others are added or configured after.

Suitable monitoring can discover a malicious application that might be consuming more system resources than it should or failing to exit completely on close. Through the use of system performance tools we can determine when hardware — such as memory, added storage, or even something as lucubrate as a hardware RAID — should be upgraded for more cost-effective use of a machine in the organization. Possibly most crucial, careful system monitoring gives us an early idea when a system component is showing early signs of failure, so that any possible downtime can be reduced.

Cautious system monitoring and built-in reconfigurability of Linux allows us to squeeze the best possible performance from our existing equipment, from customizing video drivers to applying special kernel patches to merely turning off unneeded services to release memory and processor cycles.

11. Configuring a secure system

It is a responsibility of an administrator to involve tasks and decisions to run secure Linux system and maintaining data integrity of it.

It provide strong security to individuals, corporate bodies as well as protecting parts of system even if it is under attack. The administrator should ensure-

- System has firewall.
- Not allow connection from unknown network.
- Not install software if not needed.

The Linux Administrators most important responsibility is the, protection of the computer and data integrity. The system administrator's most important task, first and foremost, is to make certain that no data on the machine or network are likely to become vitiated, whether by hardware or power failure, by mis-configuration, or by deceptive or inadvertent intrusion from anywhere.

Every person involved in computing are aware of the accelerative serious attacks upon machines connected to the Internet. The absolute majority of these have not targeted Linux systems, but that doesn't mean that Linux systems have been entirely resistant, either to direct attack or to the effects of attacks on machines running other operating systems. In one such Distributed Denial of Service (DDoS) attack aimed at several major on-line companies, many of the "zombie" machines (Machines unknowingly used to spread malware without its owners permission) — so that the ruiner could apply thousands of machines instead of just a few — were running Linux that had not been patched to defend against a well-known security fault. In the various "Code Red" attacks of the summer of 2001, Linux machines themselves were defendable, but the large amount of traffic generated by this "worm" infection nevertheless prevented many Linux machines from getting much Web-based work done for several weeks, so furious was the storm raging across the Internet. While this infection did not vitiate Linux machines as it did those running a different operating system.

The protection can be as easy as turning off unnecessary services, monitoring the Red Hat Linux security mailing list to make sure that all security announcements are followed, and otherwise engaging in good computing practices to ensure the system runs robustly without any run time errors. Or it can be an almost full-time job involving levels of security authorization within the system and systems, to which it is connected, expatiate fire walling to protect not just Linux machines but machines that, through their use of non-Linux software, are far more defenceless, and physical security — making sure no one steals the machine itself! For any machine that is

connected to any other machine, security means hardening against the attacks and making certain that no one is using our machine as a platform for launching attacks on the others. If we are running Web, ftp, or mail servers, it means giving entry to those who are eligible to it while locking out everyone else. It means making sure that passwords are not simply guessed and not made available to any unauthorized persons.

So our job as a system administrator is to pitch just the right proportion between maximum utility and maximum guard, all the while bearing in mind that confidence in a secured machine.

There are many tools and techniques that Red Hat Linux supplies to help us guard against intrusion and intruders, even to help us to prevent intrusion into non-Linux machines that may reside on our network. Linux is designed from the starting with security in mind, and in all of our works we should keep that same protection consciousness.

Is Unix / Linux Secure?

The answer to this question is – not. Neither UNIX nor Linux is secure, nor is any other operating system that transmits on a network. If we must have dead, total, unbreakable security, then we require a measurable air gap between our computer and any other device. Some people argue that we also require to enclose our computer in a special room that blocks electromagnetic radiation.

We can work to make your system somewhat more tolerant to attack. Even so, several fundamental faults in the UNIX model ensures that we will never reach security nirvana:

- UNIX is optimized for comfort and doesn't make safety easy or natural. The system's overall philosophy emphasizes casual manipulation of data in a networked, multiuser situation.
- The software that runs on UNIX systems is formed by a huge community of programmers. They range in experience level, attending to detail, and knowledge of the system and its inter-dependencies. As a result, even the most well-intended new features can initiate large security loopholes.
- Most administrative functionalities are enforced outside the kernel, where they can be scrutinized and tampered with. Hackers will have the broad access to the system.

On the other side, since some systems' source code (e.g., Linux, Open Solaris) is accessible to everyone, thousands of people can (and do) examine each line of code for possible security danger. This planning is widely believed to outcome in better security than that of closed operating systems, in which a controlled number of people have the chance to examine the code for loopholes.

Many sites are a release or two behind, either due to localization is too difficult or because they do not sign into a software care service. In any case, when security loopholes are patched, the window of chances for hackers often does not disappear all-night.

It might seem that protection should step by step improve over time as security problems are determined and corrected, but unluckily this does not seem to be the case. The system software is growing ever more complex, hackers are becoming better and better arranged, and computers are connecting more and more nearly on the Internet. Security is an ongoing fight that can never really be won. The more secure our system, the more affected us and our users will be.

Using tools to monitor security

The Linux is the preferable operating system who demands secured networks, purchase it can be easily crack by hackers.

It is necessary for administrator to be conscious of tools and techniques that hackers use and software used to monitor and counter such activity. It is responsibility of administrator to forbid unauthorized use of his system.

The crackers are people who, for purposes or to entertain themselves, like to break into other people's computers — to steal information are a smart bunch. If there is any weakness in a system, they will find it. Fortunately, the Linux development community is quick to find potential exploits and to discover ways of slamming shut the door before crackers can get into. Luckily, too, Red Hat is persevering in making available new, patched versions of packages in which expected exploits have been saved. So our first and best security tool is making sure that whenever a security advisory is issued, we should download and install the repaired package. This line of defence can be bothersome, but it is nothing compared to reconstruction of a compromised system.

The bug trackers are, sometimes their job is reactive. Forbidding the use of our machine for dreadful purposes and guarding against invasion of intruder are, in the end, our duty lone. Red Hat Linux provides us with tools and techniques to discover and deal with unauthorized access of many kinds.

References:

- UNIX and Linux System Administration Handbook 4th Edition by Evi Nemeth, Pearson Education
- Linux System Administration Recipes 1st Edition, by Kemp Juliet, Publisher: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG
- Linux: The Complete Reference, Sixth Edition, by Richard Pearson, Tata McGraw Hill Company Limited.

Unit 2

Installation of Redhat Linux

2.1 Installation of Redhat Linux in virtual machine

Data centres today are having a mix of Windows and Linux workloads. IDC (International Data Corporation) estimates in 2008 that 68 percent of all physical servers shipped are Windows based, compared to 23 percent that are Linux based. However, the development of Linux environments is steadily increasing. From 2006 to 2011, IDC forecasts the compounded annual growth rate (CAGR) of physical server units running Linux at 28.1 percent, with Windows trailing at 25.0 percent. As more data centres are virtualized with VMware Infrastructure 3, it makes sense that these virtual environments are also trending towards increased use of Linux. The CAGR of virtual server units running Linux is fore-casted by IDC at 44.1 percent, with Windows behind at 39.0 percent. The Linux operating system now hosts applications from databases to Web servers to application servers and file servers, like their Windows counterparts. Linux guest operating systems are here, and VMware is devoted to supporting them.

Linux Support on VMware ESX

The VMware ESX supports the widest range of Linux guest operating systems of any virtualization product. ESX supports Red Hat Enterprise Linux, Open SUSE Linux Enterprise Server and Ubuntu Linux with regular and LTS systems. In addition, ESX supports almost all kinds of updates to these releases as well as specialized variants of these. Choosing a Linux distribution from this list offers performance benefits over unsupported Linux distributions because VMware products optimize hypervisor settings based on the guest operating system types.

Installing Linux in a Virtual Machine

When we create the virtual machine in which we plan to install our Linux guest operating system, be sure that its devices are set up as per our expectation. We can create a Linux virtual machine from installable media. Once we have created a virtual machine, we can create templates and clones from this base virtual machine. This enables us to have a provision for future virtual machines rapidly.

Memory Recommendations

Be sure the virtual machine is configured with at least 512MB of memory for the Operating Systems Red Hat Enterprise Linux 5 or with 256MB of memory for Red Hat Enterprise Linux 3 or Red Hat Enterprise Linux 4. If the memory in the virtual machine is less than the recommended values, Red Hat Enterprise Linux presents an error message as it loads certain VMware drivers.

Network Adapter Recommendations

We have to be sure to select the correct network adapter for network communication. For most 32 -bit guest operating systems, we can select Flexible or Enhanced vmxnet . And for most 64 - bit guest operating systems, we can select E1000 or Enhanced vmxnet. Enhanced vmxnet is not

supported on every 32 bit and 64 bit Linux distribution, but if the option is existing, it is recommended to select Enhanced vmxnet as our network adapter.

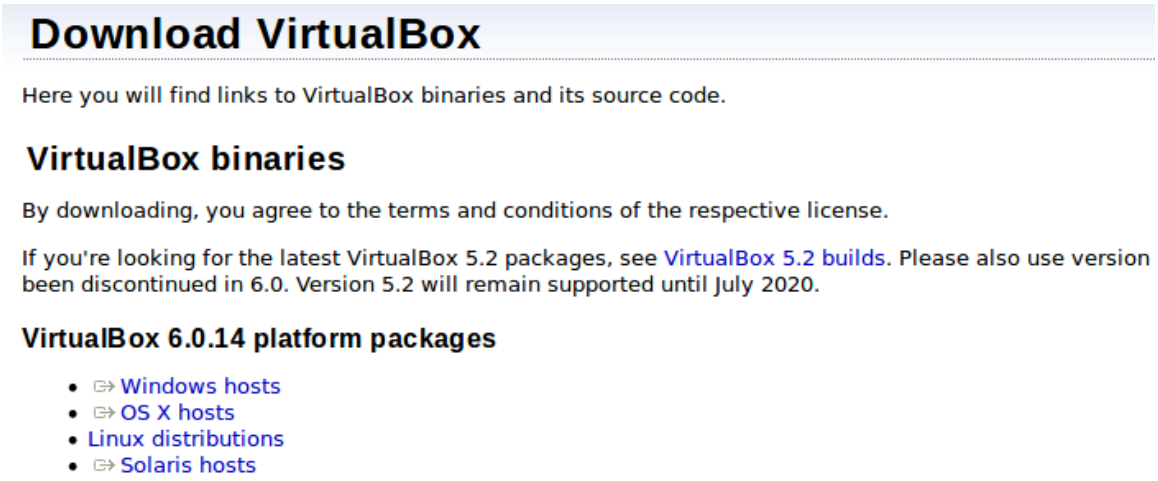
SCSI Adapter Recommendations

While creating the virtual machine, be sure to select the LSI Logic SCSI adapter. Red Hat Enterprise Linux 5 does not include a driver for the BusLogic SCSI adapter internally. Many Linux based guest operating systems encounter problems in a virtual machine configured to use the BusLogic virtual SCSI adapter. In most of the cases, VMware recommends that we use the LSI Logic virtual SCSI adapter with all Red Hat guest operating systems. However, ESX Server 2.5+ versions support only the BusLogic SCSI adapter. The VMware provides a separate BusLogic driver for Red Hat Enterprise Linux 4 Upgrades for the further versions. For instructions on downloading and installing a driver for the BusLogic adapter, we can install Red Hat Enterprise Linux 5 in a virtual machine using the standard Red Hat distribution CD, via the boot from network method, or from a PXE server. If we plan to use a PXE server to install the guest operating system over a network connection, we don't require the operating system installation media. Rather than installing from a physical CD/DVD ROM, we can create an ISO image file from the installation CD ROM or download it from the internet. Using an ISO image file in this way can be particularly convenient if we need to install the same operating system in multiple virtual machines. We can store the ISO file on the host machine or on a network drive whic accessible from the host machine.

Download Oracle VirtualBox

If we don't already have VirtualBox, download it from the below link for our platform of choice.

https://www.virtualbox.org/wiki/Downloads



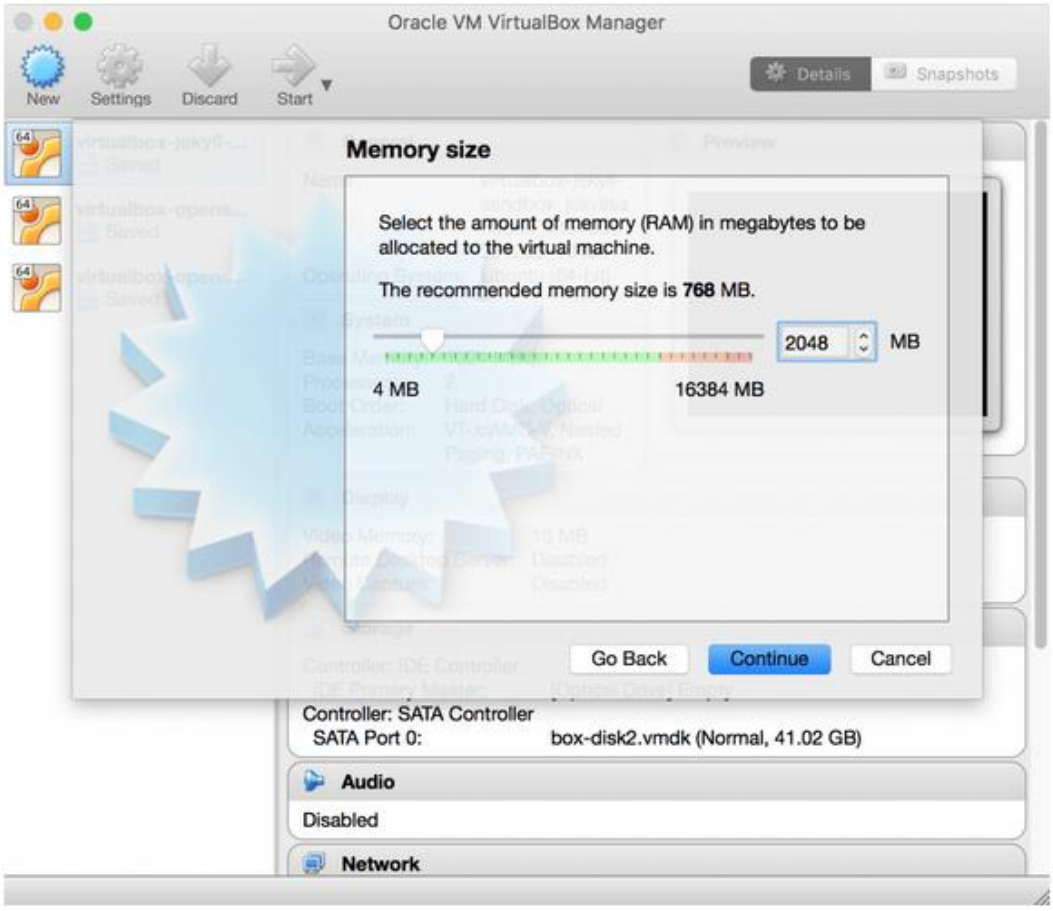
Once we are configured, we will be all set to get started.

Creating your RHEL VM in VirtualBox

The steps here are simple. We need to start by creating our new machine. Set the OS type as Linux, and choose Red Hat 64-bit for the version:

Fig. 2.1 VM - Select the Name of Operating System

Next, we need to choose the memory i.e. RAM. For speedier response, let's pick 2GB (aka 2048 MB) for our memory:



\

Fig 2.2 Define the RAM size

Next, we choose to create a new virtual disk for the operating system:

Fig. 2.3 Create virtual disk drive

After this, select VDI as the disk type:

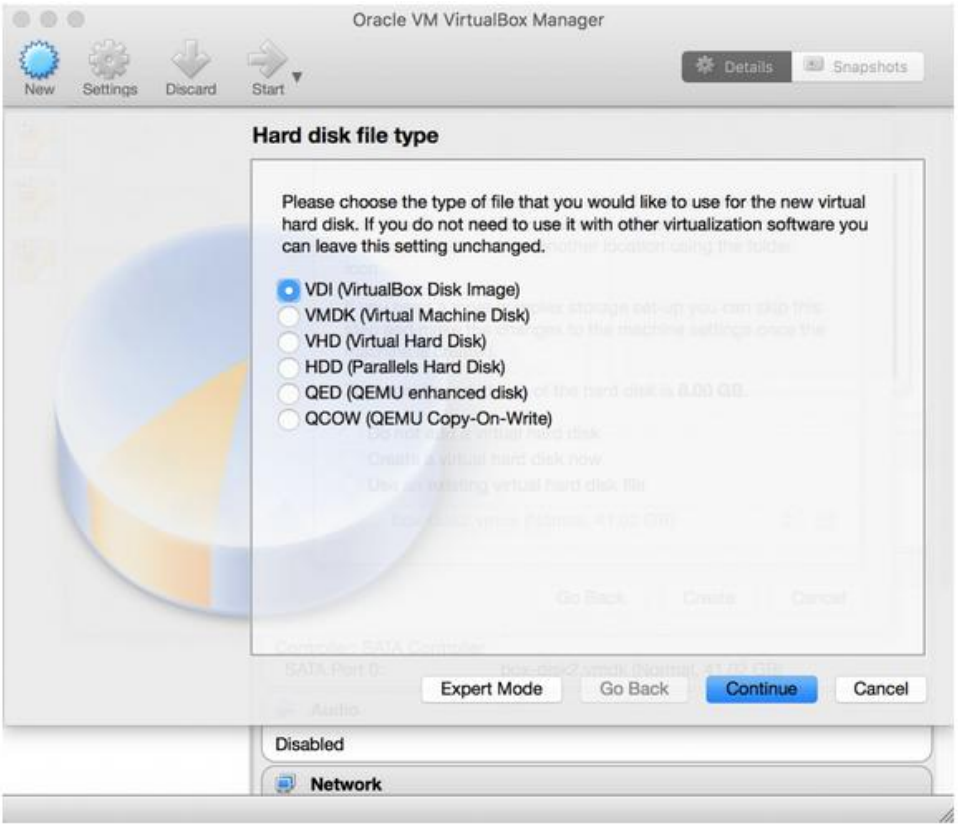


Fig. 2.4 Hard Disk type selection

Now, select dynamically allocated as the growth method:

Fig. 2.5 Select the growth method

We want a little bit of play room for our operating system, so make the disk 20 GB. Because it's dynamic, it will only use disk space as it actually gets filled, so we should be ok as long as we have room locally:

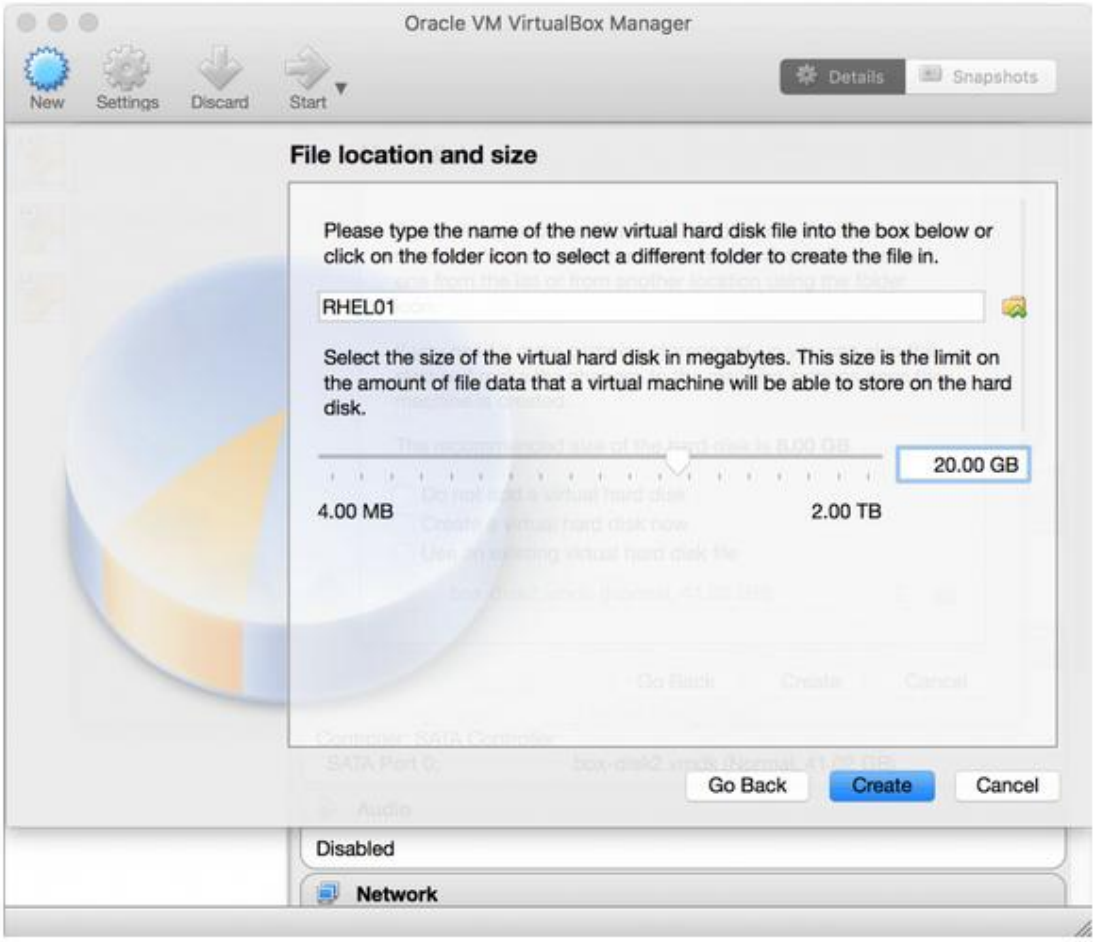


Fig. 2.6 Select file location and size

Now we have the new VM ready to start with, so click the green Start arrow and let's build our server:

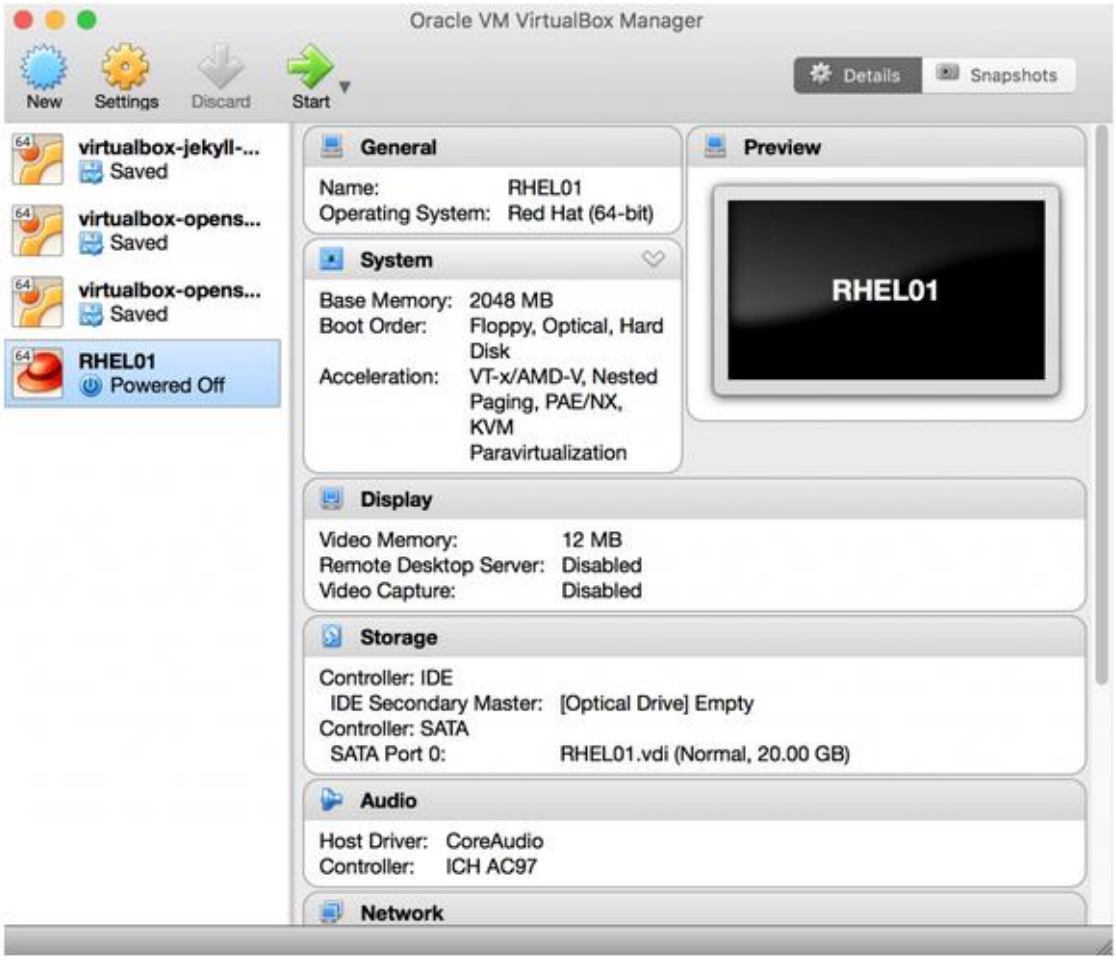


Fig. 2.7 Starting Virtual Machine

The Virtual Machine may complain about having nothing on the disk, so just let that error sit and click the little CD icon on the bottom bar of the VM window, then select Choose Disk Image and browse to find the RedHat Linux ISO file we have downloaded:

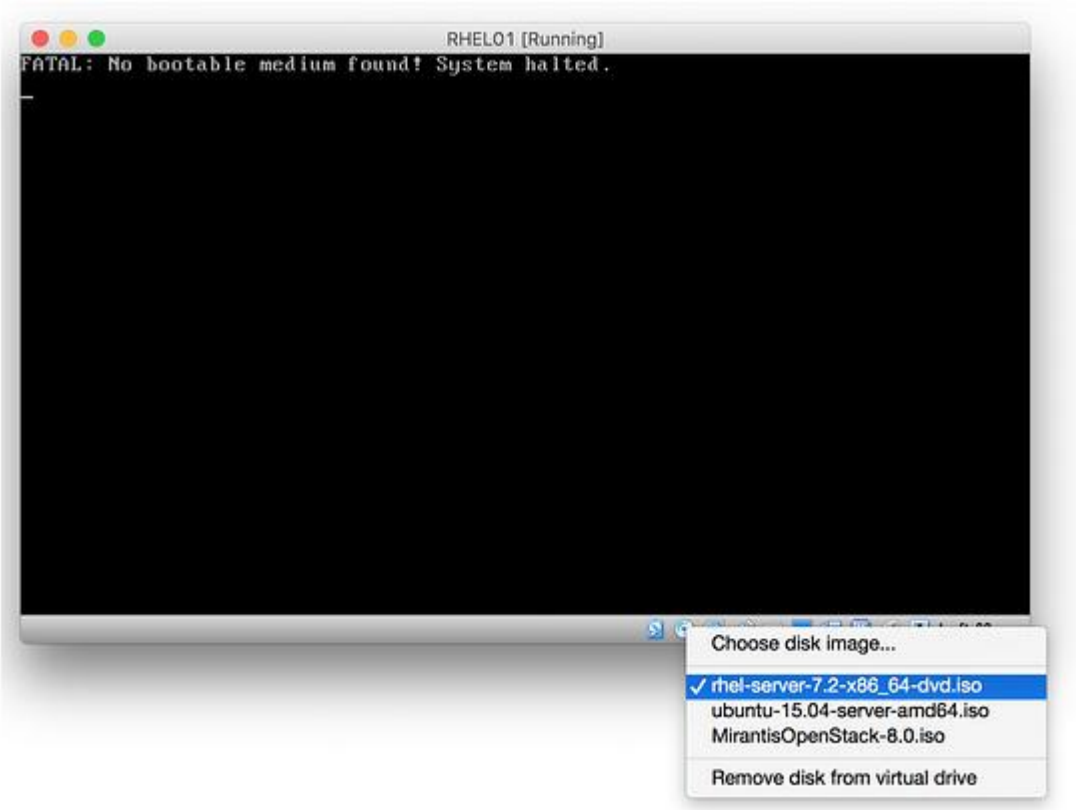


Fig. 2.8 Select the ISO file of Operating System

We can reset the VM to force the restart into the install CD now:

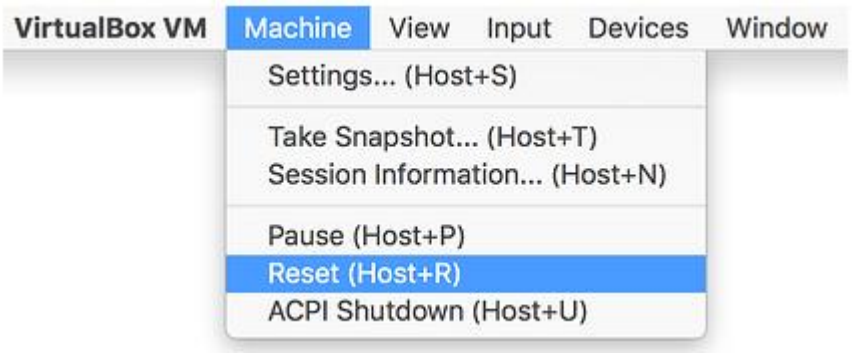


Fig. 2.9 Reset the Virtual Machine Window

This brings us to the boot screen and we can select the first option to start the installation:

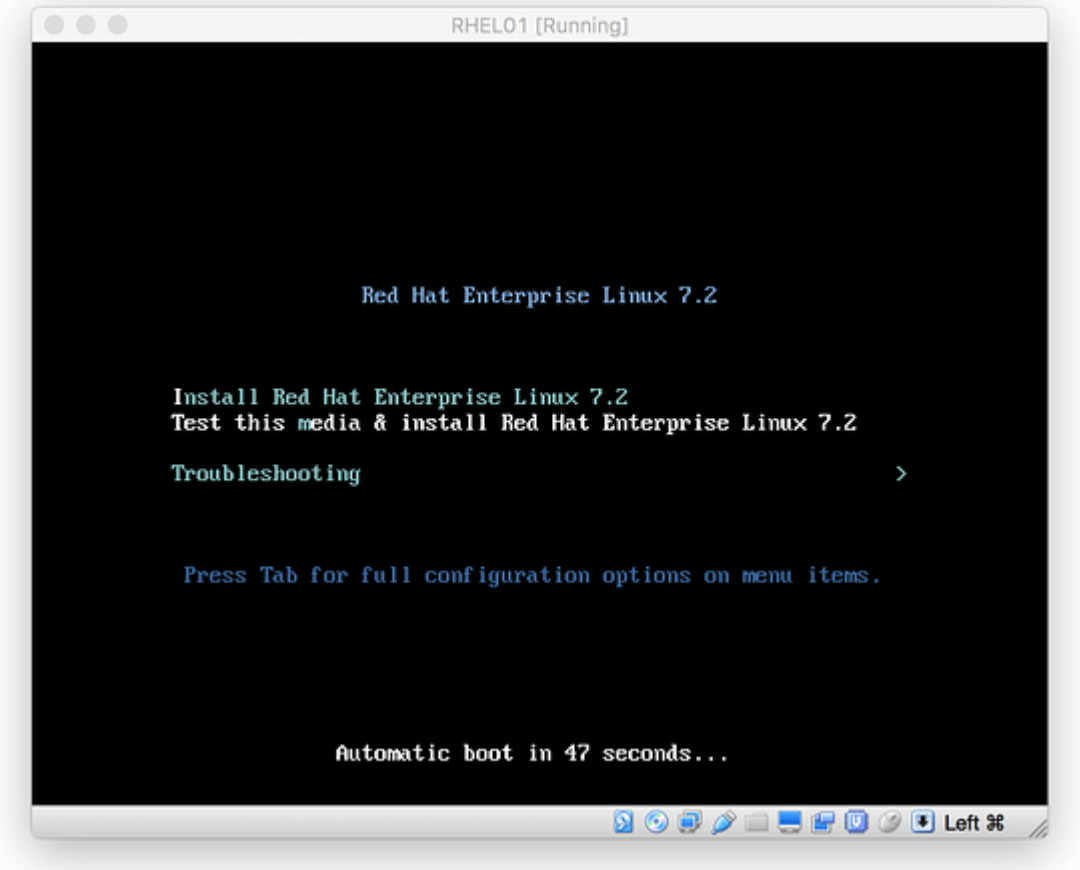


Fig. 2.10 Starting the installation

Select your language of choice at the opening menu:

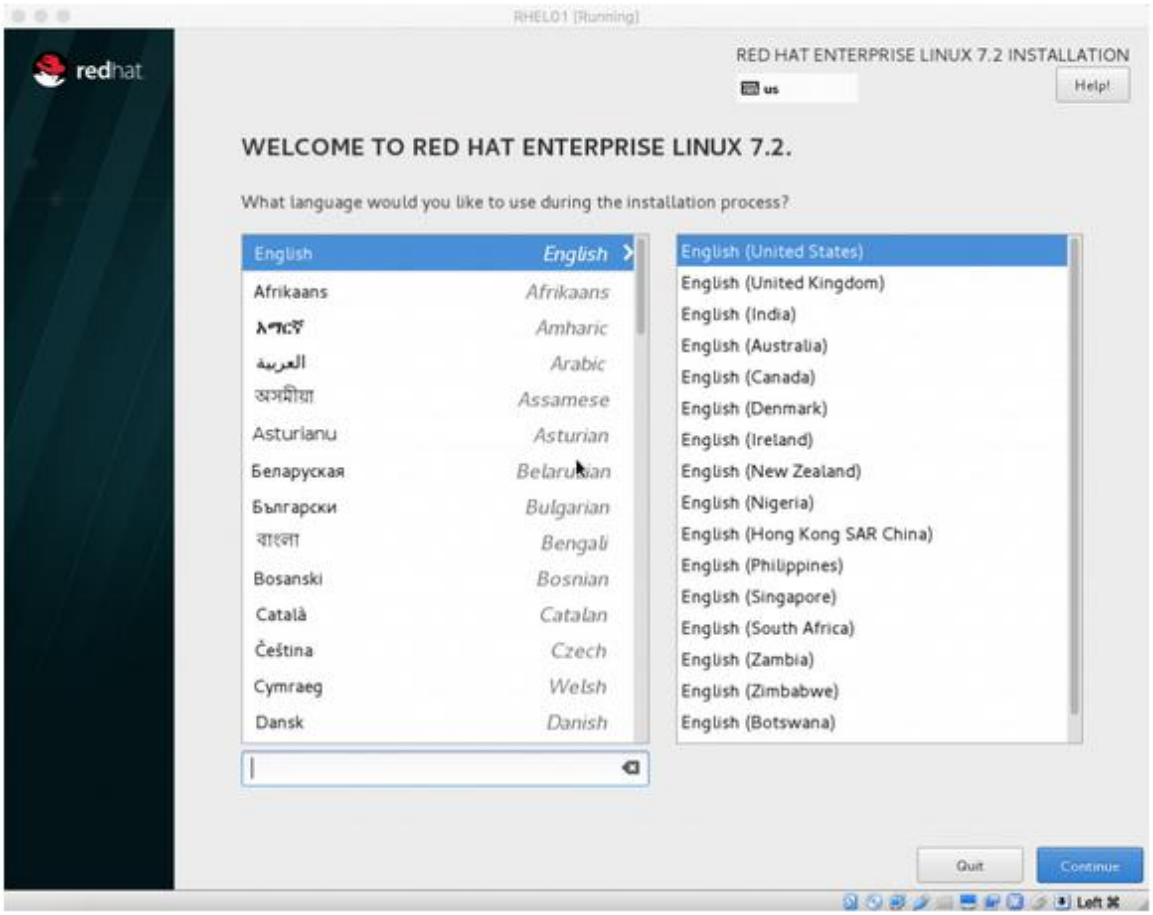


Fig. 2.11 Language Selection for OS

This brings us to the options page. We will see that the Begin Installation button is not lit because we need to set up a few things first.

Start by choosing the install type we want. For simplicity, we can select Minimal Install:

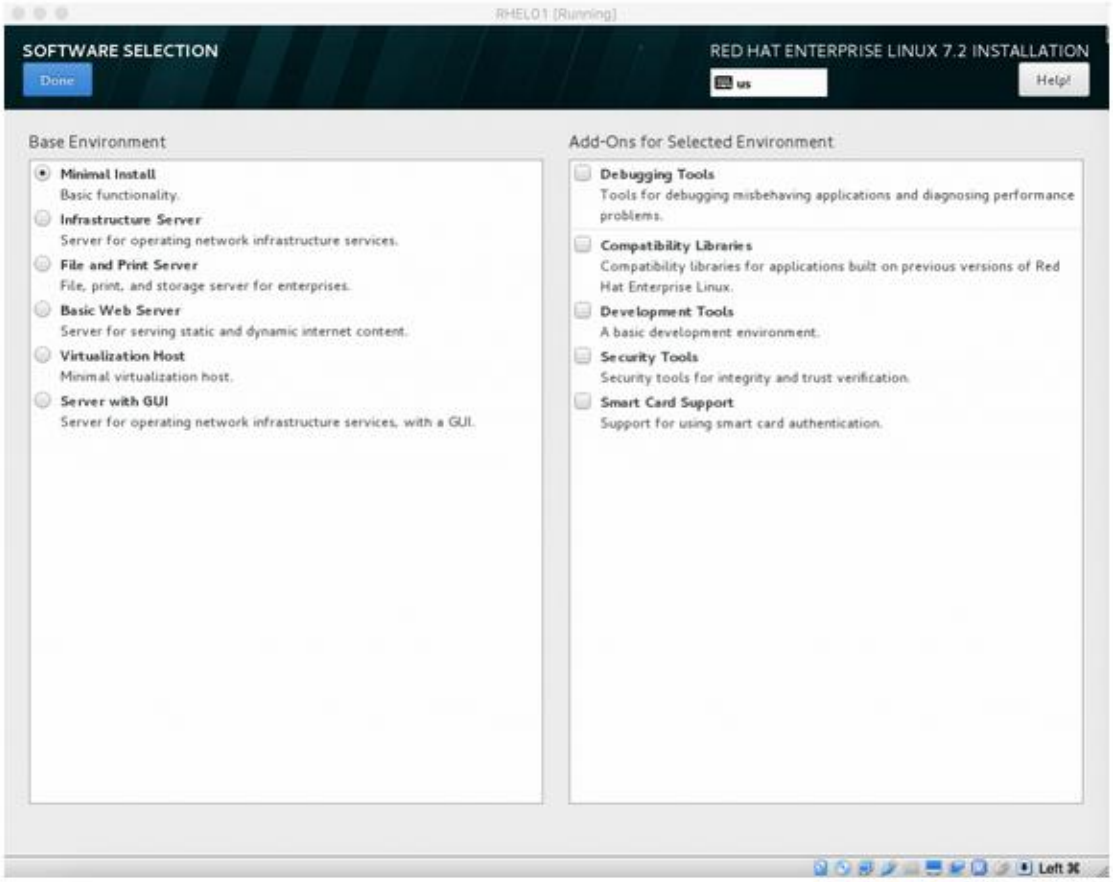


Fig. 2.12 Software Selection

We need to configure the network settings, so scroll down enter the Networking and Host Name :

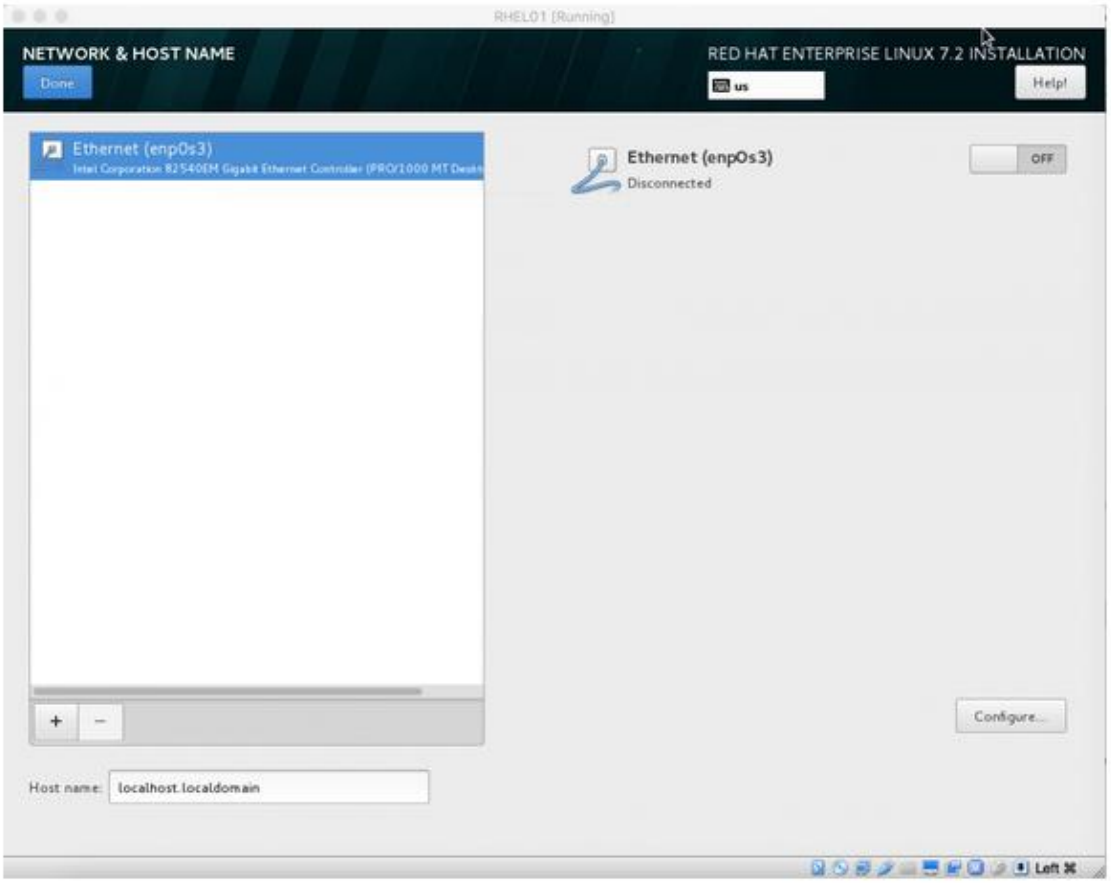


Fig. 2.13 Network Selection

Now, enable the network card in the right hand side with the toggle button:

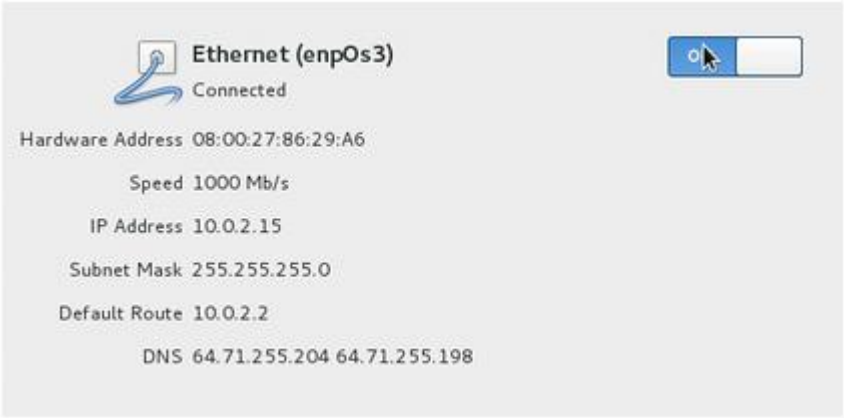


Fig. 2.14 Networking Enable

Back at the configuration screen, go to the host name option and choose our host name:

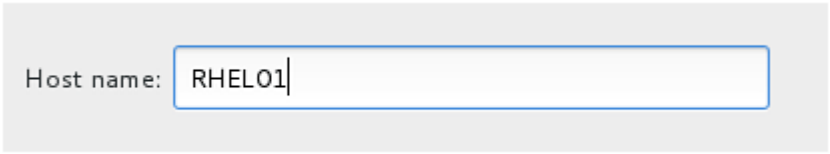


Fig. 2.15 Host name choosing

Open up the System section to choose the hard disk:

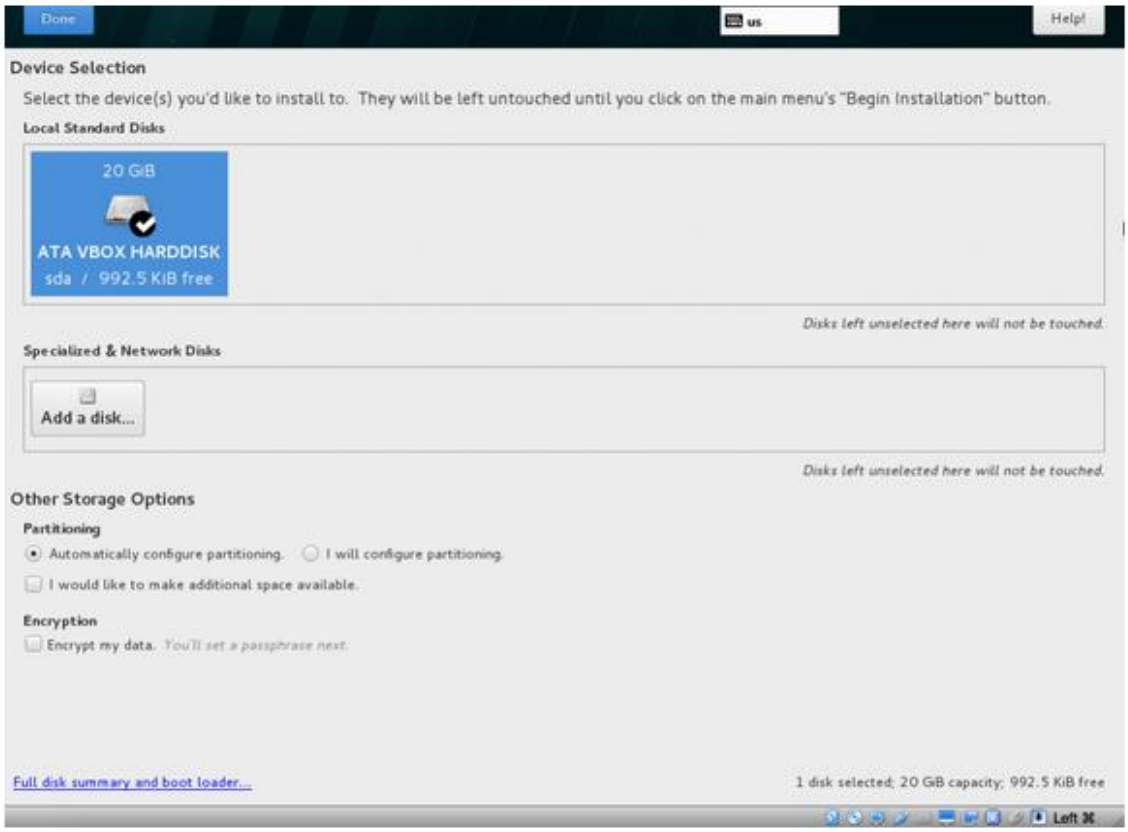


Fig. 2.16 Drive Selection for Installation

Now we will see the Begin Installation button is lit up and ready to go.

With the install underway, we can configure the root password and also set up a non-root user for you to use.

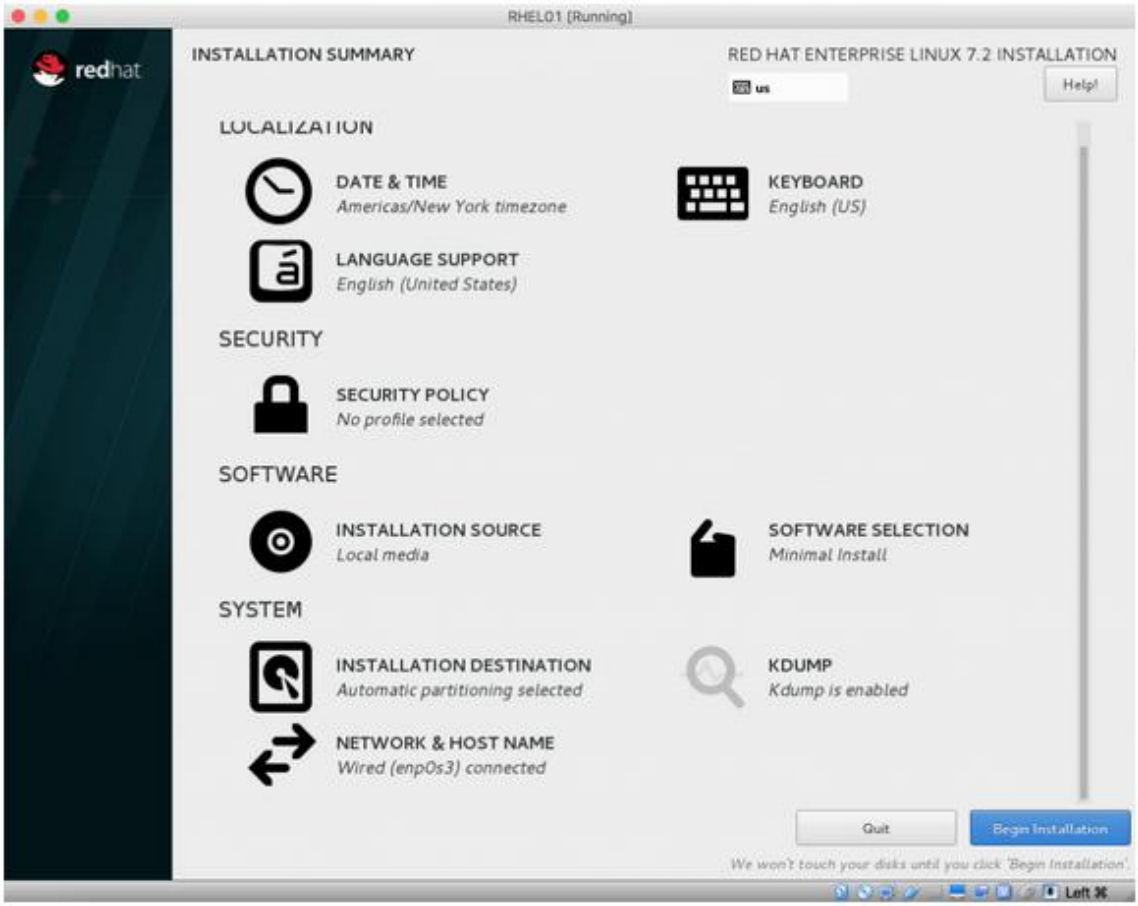


Fig. 2.17 Installation Summary

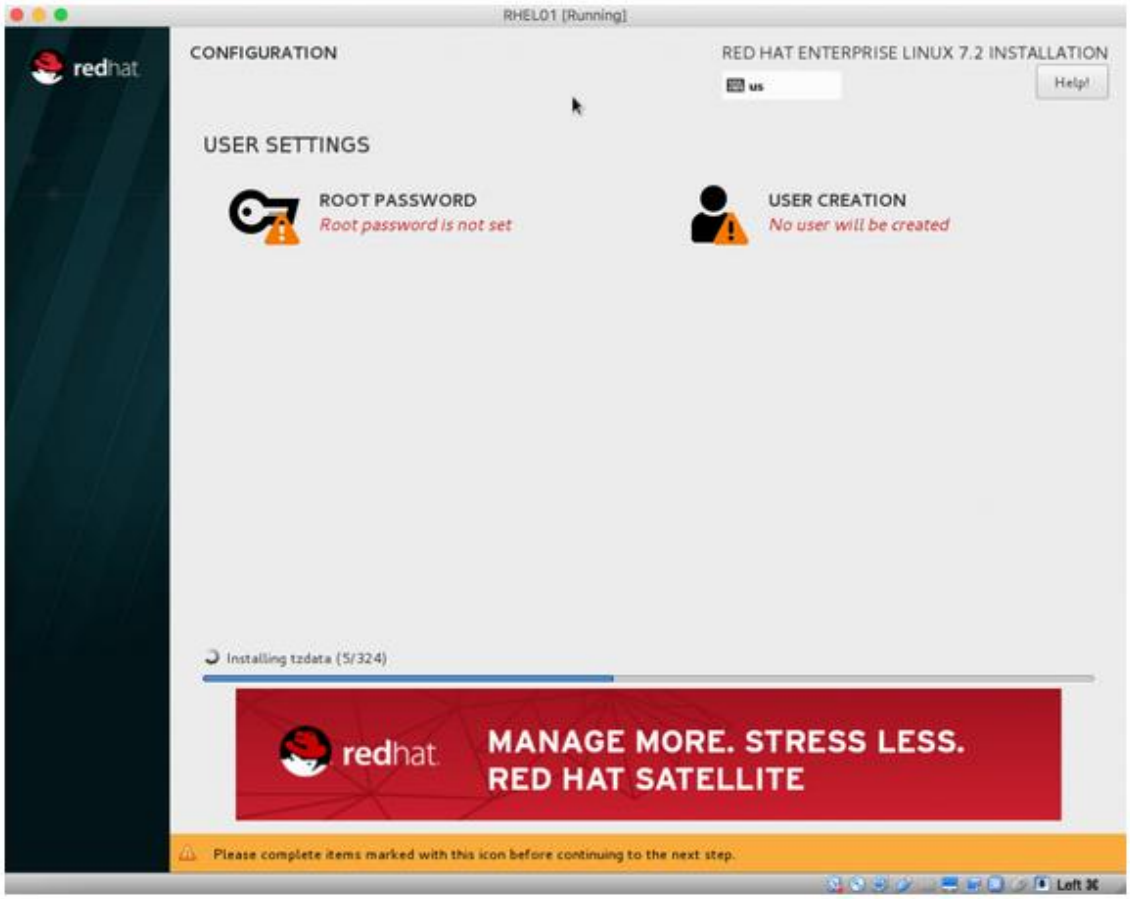


Fig. 2.18 Set the passwords

The root password needs to be reasonably complex:

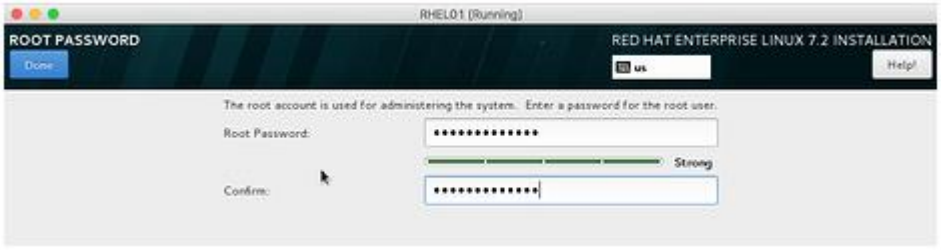


Fig. 2.19 The root passwords

Next up we can configure the non-root user for daily administration. Obviously, we will fill in our information.

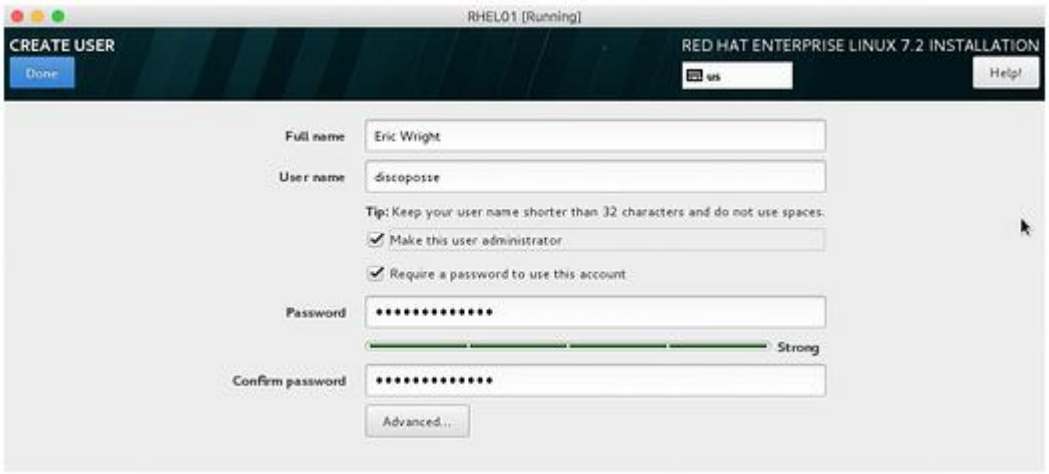


Fig. 2.20 Create the users

The installation is completed. The results may vary depending on Internet and hard drive speeds.

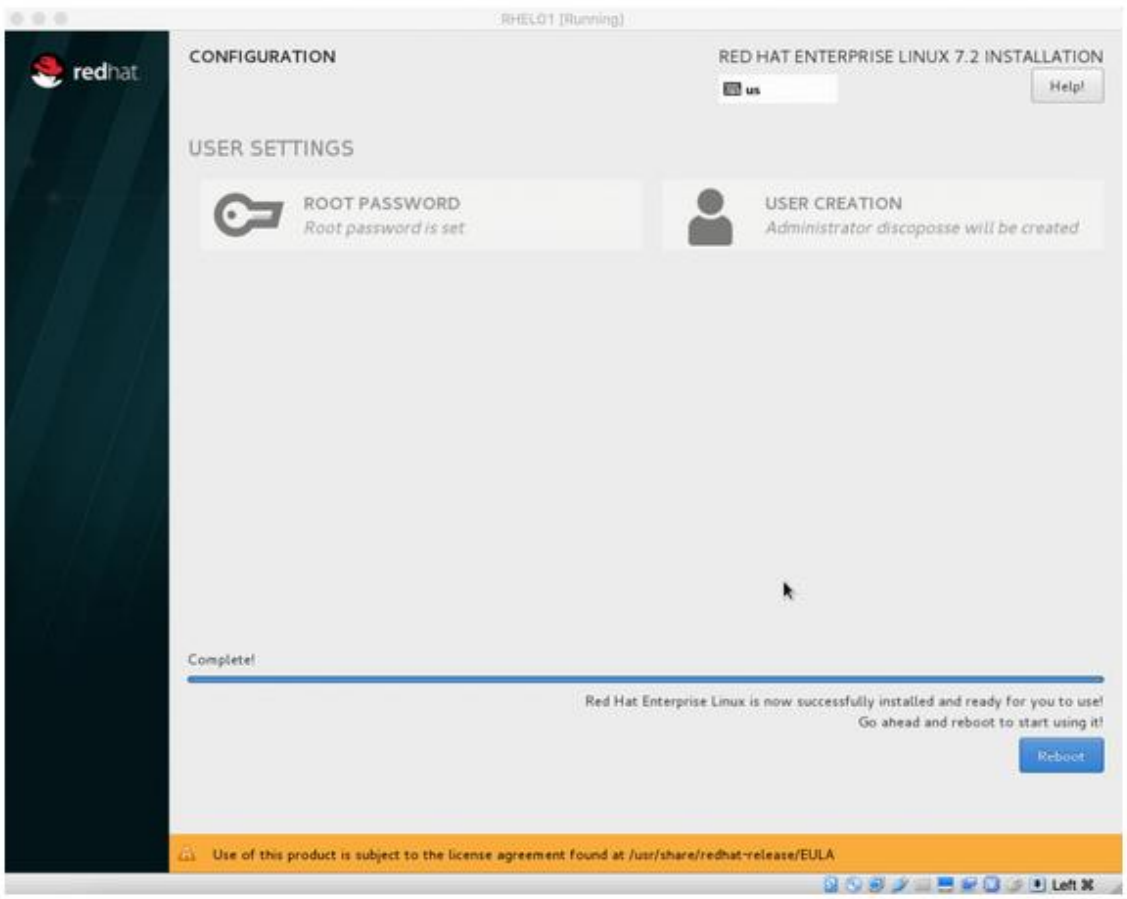


Fig. 2.21 Finishing installation

Logging in and Registering Red Hat Enterprise Linux Server

Once we reboot the server after the installation is completed, we will find ourself at the initial login prompt.

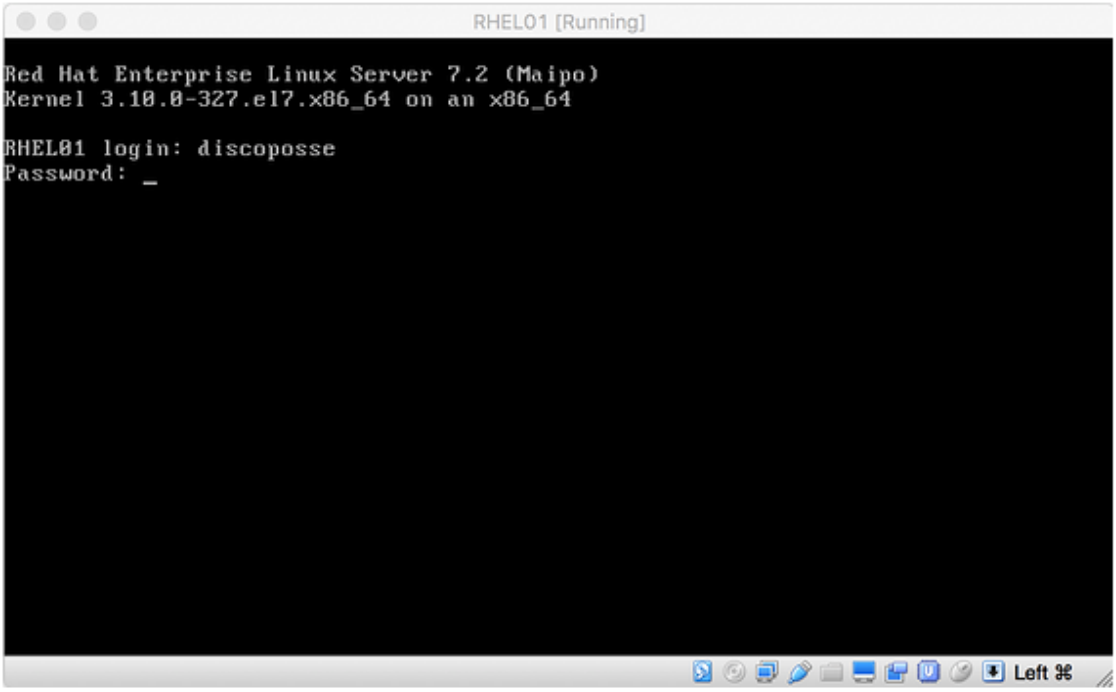


Fig. 2.22 Login Window

The Red Hat uses a subscription service to register all of their products. This is somewhat similar to the Windows Activation, and ensures that we have access to run updates and get support for our environment. It's also how we keep our server running as this is a licensed product from the company.

Using a simple single liner with our credentials will get us up and running. We can use our Red Hat subscription credentials that you used to get logged in at the start of the process. These are our Red Hat Subscription credentials, and not our local login credentials.

subscription-manager register --username --password –auto-attach

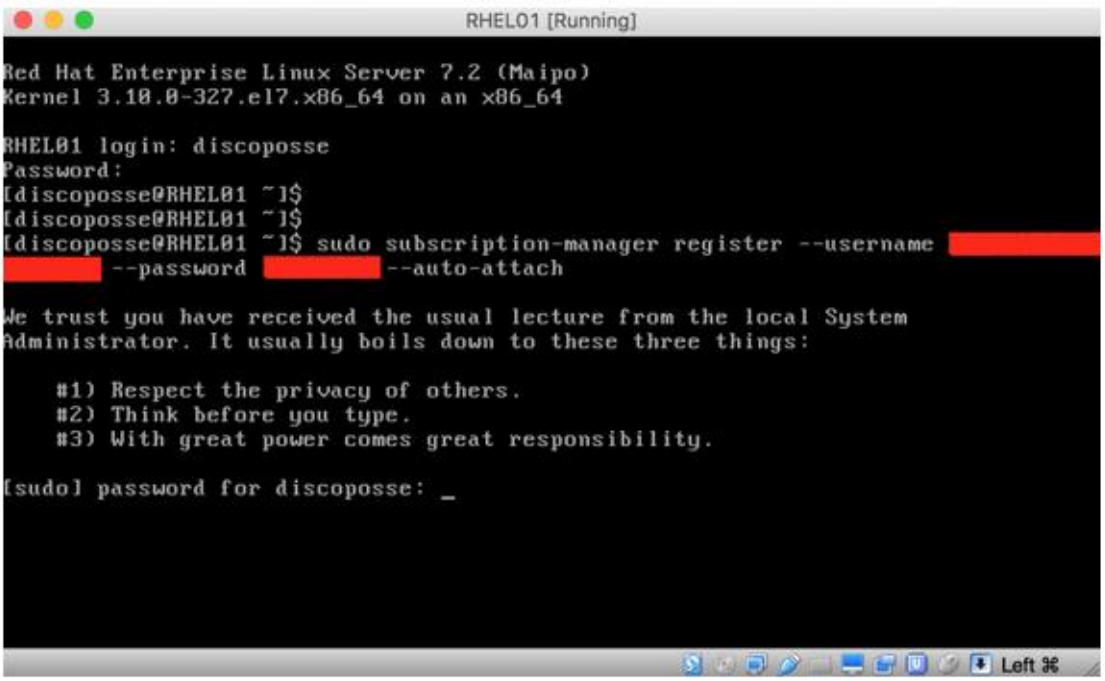


Fig. 2.23 Login Credentials

As long as we have internet access and the right username/password, we will see a message similar to this:

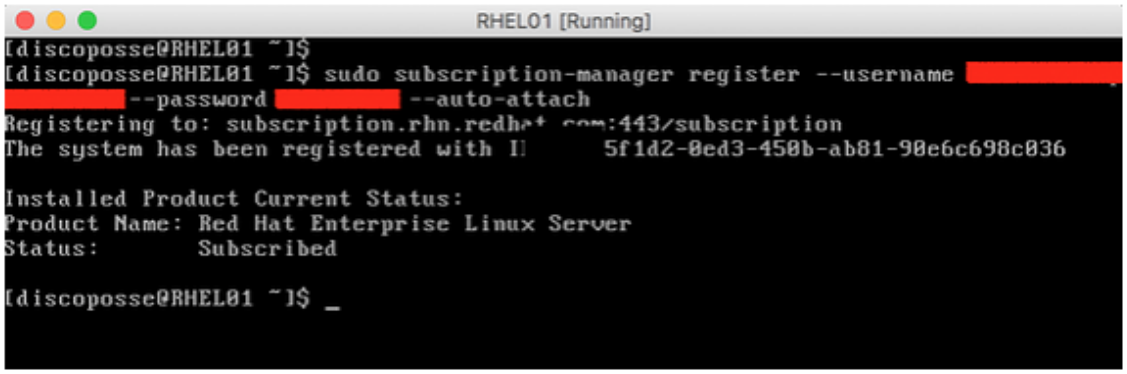


Fig. 2.24 Red Hat Linux Subscription

The last step in the install process should always be to run any kinds of updates. Because we are registered, we can now run the yum update to update the system.

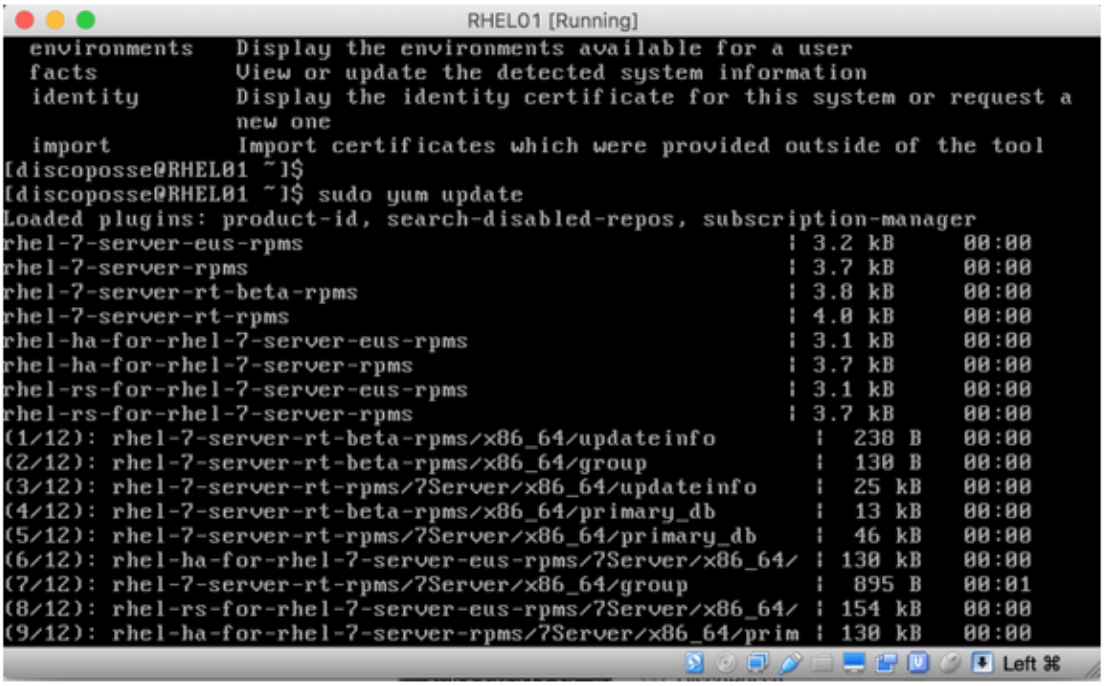


Fig. 2.25 System updation through Internet

The process will take a few minutes to pull down the necessary changes and after some confirmation, the updates will run.

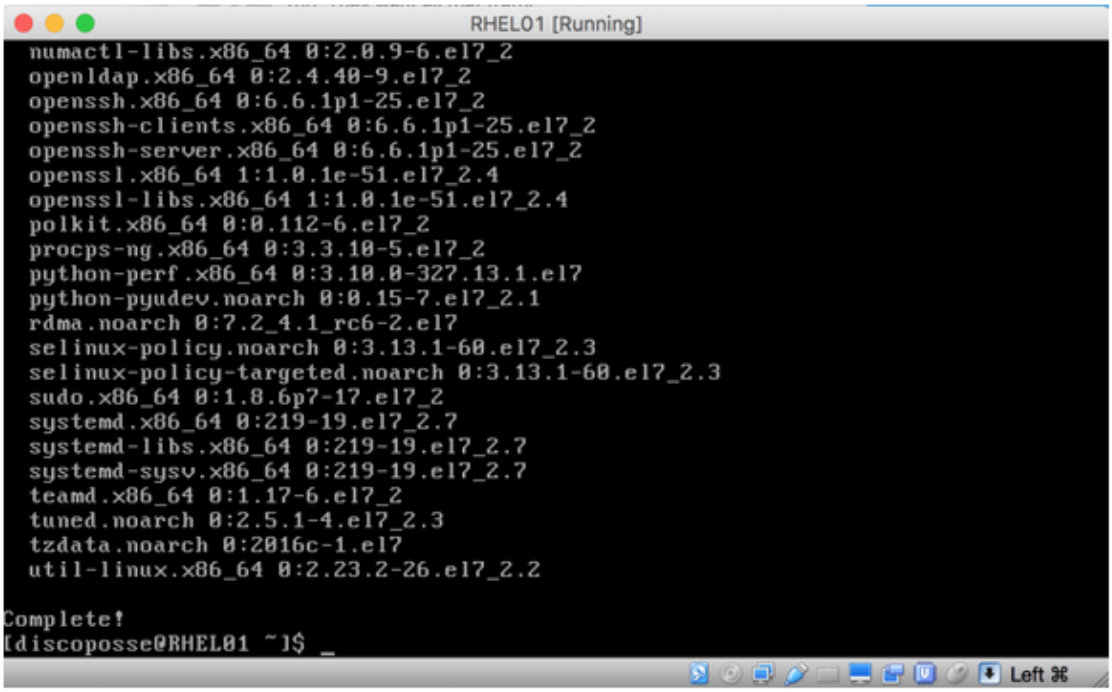


Fig. 2.26 Updation Completion

2.2 Partitions of Linux Operating System

The partition is the logically separated space on the hard drive created by operating system. Creating and deleting partitions in Linux is a regular practice because storage devices such as hard drives and USB drives must be structured in some way before they can be used. In many cases, large storage devices are divided into separate sections called partitions. The partitioning allows us to divide our hard drive into isolated sections, where each section behaves as its own hard drive. So, partitioning is particularly helpful if we run multiple operating systems on the drive.

There are many powerful tools available for creating, removing, and otherwise manipulating disk partitions in Linux. By default, Linux contains the parted, fdisk and cfdisk utilities for disk partitioning. Differences between parted and the more common fdisk and cfdisk commands include:

- GPT format: The parted command can create a Globally Unique Identifiers Partition Table GPT), while fdisk and cfdisk are limited to only DOS partition tables.
- Larger disks: The DOS partition table can format up to 2TB of disk space, although up to 16TB is possible in some case. However, a GPT partition table can address up to 8ZiB of space.
- More partitions: Using primary and extended partitions, DOS partition tables allow us only 16 partitions. With GPT, we get up to 128 partitions by default and can choose to have many more.
- Reliability: Only one copy of partition table is stored in a DOS partition. GPT keeps two copies of the partition table (at the beginning and the end of the disk). GPT also uses a CRC checksum to check the partition table integrity, which is not done with DOS partitions.

With today's larger disks and the need for more flexibility in working with them, using parted to work with disk partitions is recommended. Most of the time, disk partition tables are created as part of the operating system installation process. Direct use of the parted command is most useful when adding a storage device to an existing system.

Partitions

A hard disk can be divided into several multiple parts called as partitions. Each partition functions as if it were a separate hard disk. The idea is that if we have one hard disk, and want to have, say, two operating systems on it, we can divide the disk into two partitions. So, each operating system uses its partition as it wishes and doesn't touch the other ones. This way the two operating systems can co-exist peacefully on the same hard disk. Without partitions one would have to buy a hard disk for each operating system.

The MBR, boot sectors and partition table

The information about how a hard disk has been partitioned is stored in its very first sector (that is, the first sector of the first track on the first disk surface). The first sector is the master boot record (MBR) of the disk; this is the sector that the BIOS reads in and starts when the machine is booted for first time. The master boot record contains a tiny program that reads the partition table, checks which partition is active (marked as bootable), and reads the first sector of that partition, the partition's boot sector (the MBR is also a boot sector). This boot sector contains another tiny program that reads the first part of the operating system stored on that partition (assuming it is bootable), and then starts it.

The partitioning scheme is not built into hardware, or even into the BIOS. It is the only convention that many operating systems follow. Not all operating systems do follow it, but they are the exceptions. Some operating systems support partitions, but they occupy one partition on the hard disk, and use their internal partitioning method within that partition. The latter type exists safely with other operating systems (including Linux), and does not require any special measures, but an operating system that doesn't support partitions can’t co-exist on the same disk with any other operating system.

For a safety precaution, it is a good idea to write down the partition table on a piece of paper, so that if it ever corrupts we don't have to lose all our files. A bad partition table can be fixed with fdisk command. The relevant information is given by the fdisk -l command as shown below.

```
$ fdisk -l /dev/hda

Disk /dev/hda: 15 heads, 57 sectors, 790 cylinders
Units = cylinders of 855 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1           1       1      24   10231+  82  Linux swap
/dev/hda2          25      25      48   10260   83  Linux native
/dev/hda3          49      49     408  153900   83  Linux native
/dev/hda4         409     409     790  163305    5  Extended
/dev/hda5         409     409     744  143611+  83  Linux native
/dev/hda6         745     745     790   19636+  83  Linux native
$
```

Fig. 2.27 The fdisk command

Extended and logical partitions

The original partitioning scheme for computer hard disks allowed only four partitions. In order to overcome this design problem, extended partitions were invented. This trick allowed partitioning a primary partition into sub-partitions. The primary partition thus subdivided is the extended partition; the sub-partitions are logical partitions. They behave like primary partitions, but are created differently. There is no difference of speed between them. By using an extended partition we can now have up to 15 partitions per disk.

The partition structure of a hard disk may look like that in Figure 2.28. The disk is divided into three primary partitions, the second of which is divided into two logical partitions. Part of the disk is not partitioned at all. The disk as a whole and each primary partition has a boot sector.

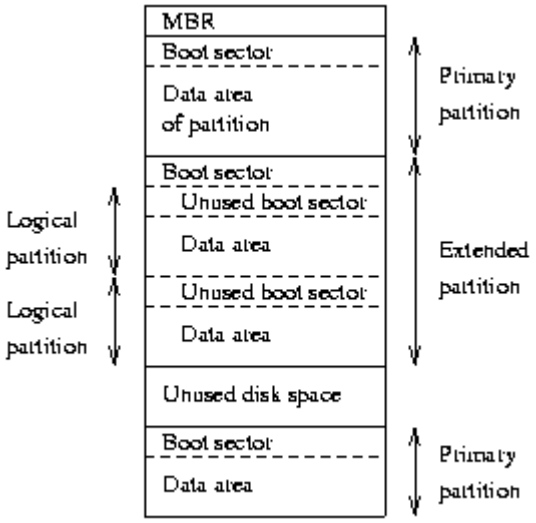


Fig. 2.28 A sample hard disk partitioning.

Partition types

The partition tables (the one in the MBR, and the ones for extended partitions) contain one byte per partition that identifies the type of that partition. This attempts to identify the operating system that uses the partition, or what it uses it for. The aim is to make it possible to avoid having multiple operating systems accidentally using the same partition. However, at actual, operating systems do not really care about the partition type byte; e.g., Linux doesn't care at all what it is.

There is no standardization office to specify what each byte value means, but as far as Linux is concerned, check the list of partition types as per the fdisk program.

```
 0  Empty           1c  Hidden Win95 FA 70  DiskSecure Mult bb  Boot Wizard hid
 1  FAT12           1e  Hidden Win95 FA 75  PC/IX           be  Solaris boot
 2  XENIX root      24  NEC DOS         80  Old Minix       c1  DRDOS/sec (FAT-
 3  XENIX usr       39  Plan 9          81  Minix / old Lin c4  DRDOS/sec (FAT-
 4  FAT16 <32M      3c  PartitionMagic  82  Linux swap      c6  DRDOS/sec (FAT-
 5  Extended        40  Venix 80286     83  Linux           c7  Syrinx
 6  FAT16           41  PPC PReP Boot   84  OS/2 hidden C:  da  Non-FS data
 7  HPFS/NTFS       42  SFS             85  Linux extended  db  CP/M / CTOS / .
 8  AIX             4d  QNX4.x          86  NTFS volume set de  Dell Utility
 9  AIX bootable    4e  QNX4.x 2nd part 87  NTFS volume set df  BootIt
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 8e  Linux LVM       e1  DOS access
 b  Win95 FAT32     50  OnTrack DM      93  Amoeba          e3  DOS R/O
 c  Win95 FAT32 (LB 51  OnTrack DM6 Aux 94  Amoeba BBT      e4  SpeedStor
 e  Win95 FAT16 (LB 52  CP/M            9f  BSD/OS          eb  BeOS fs
 f  Win95 Ext'd (LB 53  OnTrack DM6 Aux a0  IBM Thinkpad hi ee  EFI GPT
10  OPUS            54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    55  EZ-Drive        a6  OpenBSD         f0  Linux/PA-RISC b
12  Compaq diagnost 56  Golden Bow      a7  NeXTSTEP        f1  SpeedStor
14  Hidden FAT16 <3 5c  Priam Edisk     a8  Darwin UFS      f4  SpeedStor
16  Hidden FAT16    61  SpeedStor       a9  NetBSD          f2  DOS secondary
17  Hidden HPFS/NTF 63  GNU HURD or Sys ab  Darwin boot     fd  Linux raid auto
18  AST SmartSleep  64  Novell Netware  b7  BSDI fs         fe  LANstep
1b  Hidden Win95 FA 65  Novell Netware  b8  BSDI swap       ff  BBT
```

Fig. 2.29 Types of Partitions

Partitioning a hard disk

There are many programs available for creating and removing partitions from the hard disk. Many operating systems have their own, and it can be a fine idea to use each operating system's own, just in case it does something unusual that the others can't. Many of the programs are called fdisk, including the Linux one, or variations. The details on using the Linux fdisk given on its man page. The cfdisk command is similar to fdisk, but has a nicer full screen user interface.

While using IDE disks, the boot partition (that is, the partition with the bootable kernel image files) must be completely within the first 1024 cylinders. This is because the disk is used via the BIOS during boot (before the system goes into protected mode), and BIOS can't handle more than 1024 cylinders. It is sometimes possible to use a boot partition that is only partly within the first 1024 cylinders. This works as long as all the files that are read with the BIOS are within the first 1024 cylinders. Since this is difficult to arrange, it is a very bad idea to do it; we never know when a kernel update or disk defragmentation will result in an unbootable system. Therefore, make sure our boot partition is completely within the first 1024 cylinders.

This may no longer be true with newer versions of LILO (Linux Loader) that support LBA (Logical Block Addressing). Some newer versions of the BIOS and IDE disks can, in fact, handle disks with more than 1024 cylinders. If we have such a system, we can forget about the problem; if we aren't quite sure of it, put it within the first 1024 cylinders.

Each partition should have an even number of sectors, since the Linux file systems use a 1 kilobyte block size, i.e., two sectors. An odd number of sectors will result in the last sector being unused. This won't result in any problems, but it is ugly, and some versions of fdisk will warn about it.

Changing a partition's size usually requires first backing up everything we want to save from that partition (preferably the whole disk, just in case), deleting the partition, creating new partition, then restoring everything to the new partition. If the partition is growing, we may need to adjust the sizes (and backup and restore) of the adjoining partitions as well.

Since changing partition sizes is difficult, it is preferable to get the partitions right the first time, or have an effective and simple to use backup system. If we are installing from a media that does

not require much human intervention (say, from CD-ROM, of Pen drives), it is often easy to play with different configuration at first. Since we don't already have data to back up, it is not so agonized to modify partition sizes several times.

There is a program for MS-DOS, called fips, which resizes an MS-DOS partition without requiring the backup and restore, but for other file systems it is still necessary.

The fips program is included in most Linux distributions. The commercial partition manager "Partition Magic" also has a similar facility but with a nicer interface. Make sure we have a recent backup of any important data before we try changing partition sizes. The program parted can resize other types of partitions as well as MS-DOS, but sometimes in a limited manner.

2.3 Booting and Shutting down Linux

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.

During bootstrapping, the computer first loads a small piece of code called as bootstrap loader, which in turn loads and starts the operating system. The bootstrap loader is usually stored in a fixed location on a hard disk. The reason for this two step process is that the operating system is big and complicated, but the first piece of code that the computer loads must be very small (a few hundred bytes), to avoid making the firmware unnecessarily complicated.

Different computers do the bootstrapping differently. For Personal Computers, the computer (its BIOS) reads in the first sector (called the boot sector) of a hard disk. The bootstrap loader is contained within this sector. It loads the operating system from elsewhere on the disk (or from some other place).

After Linux has been loaded, it initializes the hardware and device drivers, and then runs init. The init starts other processes to allow users to log in, and do things.

In order to shut down a Linux system, first all processes are told to terminate (this makes them close any files and do other necessary things to keep things tidy), then file systems and swap areas are unmounted, and finally a message is printed to the console that the power can be turned off. If the proper procedure is not followed, terrible things can and will happen; most importantly, the file system buffer cache might not be flushed, which means that all data in it is lost and the file system on disk is inconsistent, and therefore possibly unusable.

2.4 The boot process

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 in short. It will choose a disk drive (typically the first CD/DVD drive or Pen Drive, if there is a CD/DVD or Pen Drive inserted, otherwise the first hard disk, if one is installed in the computer; the order might be configurable) 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, since a hard disk can contain several partitions, each with their own boot sectors.

The boot sector contains a tiny program whose responsibility is to read the actual operating system from the disk and start it. When booting Linux from a CD / Pen Drive, 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 Pen Drive, there is no filesystem, 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 filesystem, by using LILO, the LInux LOader, or GRUB, the GRand Unifying Bootloader.

When booting from the hard disk, the code in the master boot record (MBR) 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 code in the partition's boot sector does what a Pen Drive's boot sector does: it will read in the kernel from the partition and start it. 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 filesystem 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 bootloader 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 bootloader can be configured so that it will always ask, with an optional timeout that will cause the default kernel to be booted.

It is also possible to give a kernel command line argument, after the name of the kernel or operating system. Booting from pen drive and from hard disk have both their advantages, but generally booting from the hard disk is nicer. It is also faster. Most Linux distributions will setup the bootloader for us during the install process.

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.
- If you have a super-VGA card that Linux recognizes and that has some special text modes (such as 100 columns by 40 rows), Linux asks us which mode we want to use. During the kernel compilation, it is possible to preset a video mode, so that this is never asked. This can also be done with LILO, GRUB or rdev.
- After this, the kernel checks what other hardware there is (pen drives, hard disks, network adapters, etc), and configures some of its device drivers appropriately; while it does this, it outputs messages about its findings. For example, It will look like this:

```
LILO boot:
Loading linux.
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.94 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
lp_init: lp1 exists (0), using polling driver
Memory: 7332k/8192k available (300k kernel code, 384k reserved, 176k
data)
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
Loopback device init
Warning WD8013 board not found at i/o = 280.
Math coprocessor using irq13 error reporting.
Partition check:
  hda: hda1 hda2 hda3
VFS: Mounted root (ext filesystem).
Linux version 0.99.pl9-1 (root@haven) 05/01/93 14:12:20
```

Fig 2.30 Kernel LILO Boot

The exact texts are different on different systems, depending on the hardware, the version of Linux being used, and how it has been configured.

Then the kernel will try to mount the root filesystem. The place is configurable at compilation time, or any time with rdev or the bootloader. The filesystem type is detected automatically. If the mounting of the root filesystem fails, for example because we didn't remember to include the corresponding filesystem driver in the kernel, the kernel panics and halts the system.

The root filesystem is usually mounted read-only (this can be set in the same way as the place). This makes it possible to check the filesystem while it is mounted; it is not a good idea to check a filesystem that is mounted read-write.

After this, the kernel starts the program init (located in /sbin/init) in the background (this will always become process number 1). init does various startup chores. The exact things it does depends on how it is configured; **init** then switches to multi-user mode, 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.

More about shutdowns

It is important to follow the correct procedures when we shut down a Linux system. If we fail do so, our filesystems 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 we 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 filesystem (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, we ensure that all background processes can save their data.

The command for properly shutting down a Linux system is **shutdown**. It is usually used in one of two ways.

If we are running a system where we are the only user, the usual way of using **shutdown** is to quit all running programs, log out on all virtual consoles, log in as root on one of them (or stay logged in as root if we already are, but we should change to root's home directory or the root directory, to avoid problems with unmounting), then give the command **shutdown -h now** (substitute now with a plus sign and a number in minutes if we want a delay, though we usually don't on a single user system).

Alternatively, if our system has many users, use the command **shutdown -h +timemessage**, 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.

```
# shutdown -h +10 'We will install a new
disk.  System should
> be back on-line in three hours.'
#
```

Fig. 2.31 Shutdown message

This will warn everybody that the system will shut down in ten minutes, and that they'd better get lost or lose data. The warning is printed to every terminal on which someone is logged in, including all **xterm**s:

```
Broadcast message from root (ttyp0) Wed Aug  2 01:03:25 1995...

We will install a new disk.  System should
be back on-line in three hours.
The system is going DOWN for system halt in 10 minutes !!
```

Fig. 2.32 The Shutdown messages

The warning is automatically repeated a few times before the boot, with shorter and shorter intervals as the time runs out.

When the real shutting down starts after any delays, all filesystems (except the root one) are unmounted, user processes (if anybody is still logged in) are killed, daemons are shut down, all filesystem are unmounted, and generally everything settles down. When that is done, **init** prints out a message that we can power down the machine. Then, and only then, should we move our fingers towards the power switch.

Sometimes, although rarely on any good system, it is impossible to shut down properly. For instance, if the kernel panics and crashes and burns and generally misbehaves, it might be completely impossible to give any new commands, hence shutting down properly is somewhat difficult, and just about everything we can do is hope that nothing has been too severely damaged and turn off the power. If the troubles are a bit less severe (say, somebody hit your keyboard with an axe), and the kernel and the **update** program still run normally, it is probably a good idea to wait a couple of minutes to give **update** a chance to flush the buffer cache, and only cut the power after that.

In the old days, some people like to shut down using the command **sync** three times, waiting for the disk I/O to stop, then turn off the power. If there are no running programs, this is equivalent to using **shutdown**. However, it does not unmount any filesystems and this can lead to problems with the ext2fs “clean filesystem"” flag. The triple-sync method is not recommended.

Rebooting

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**.

Most Linux systems run **shutdown -r now** when ctrl-alt-del is pressed on the keyboard. This reboots the system. The action on ctrl-alt-del is configurable, however, and it might be better to allow for some delay before the reboot on a multiuser machine. Systems that are physically accessible to anyone might even be configured to do nothing when ctrl-alt-del is pressed.

Single user mode

The shutdown command can also be used to bring the system down to single user mode, in which no one can log in, but root can use the console. This is useful for system administration tasks that can't be done while the system is running normally.

Emergency boot pen drives

It is not always possible to boot a computer from the hard disk. For example, if we make a mistake in configuring LILO, we might make our system unbootable. For these situations, we need an alternative way of booting that will always work (as long as the hardware works). For typical PCs, this means booting from the pen drive.

Most Linux distributions allow one to create an emergency boot floppy during installation. This is referred as Live booting of Linux. It is a good idea to do this. However, some such boot disks contain only the kernel, and assume us will be using the programs on the distribution's installation disks to fix whatever problem we have. Sometimes those programs aren't enough; for example, we might have to restore some files from backups made with software not on the installation disks.

2.5 init

init is one of those programs that are absolutely essential to the operation of a Linux system, but that we still can mostly ignore. A good Linux distribution will come with a configuration for init that will work for most systems, and on these systems there is nothing we need to do about init. Usually, we only need to worry about init if we hook up serial terminals, dial-in (not dial-out) modems, or if we want to change the default run level.

When the kernel has started itself (has been loaded into memory, has started running, and has initialized all device drivers and data structures and such), it finishes its own part of the boot process by starting a user level program, init. Thus, init is always the first process (its process number is always 1).

The kernel looks for init in a few locations that have been historically used for it, but the proper location for it (on a Linux system) is /sbin/init. If the kernel can't find init, it tries to run /bin/sh, and if that also fails, the startup of the system fails.

When init starts, it finishes the boot process by doing a number of administrative tasks, such as checking filesystems, cleaning up /tmp, starting various services, and starting a getty for each terminal and virtual console where users should be able to log in.

After the system is properly up, init restarts getty for each terminal after a user has logged out (so that the next user can log in). init also adopts orphan processes: when a process starts a child process and dies before its child, the child immediately becomes a child of init. This is important for various technical reasons, but it is good to know it, since it makes it easier to understand process lists and process tree graphs. There are a few variants of init available. Most Linux distributions use sysvinit, which is based on the System V init design. The BSD versions of Unix have a different init. The primary difference is run levels: System V has them, BSD does not (at least traditionally). This difference is not essential. We'll look at sysvinit only.

2.6 Run levels

A run level is a state of init and the whole system that defines what system services are operating. Run levels are identified by numbers. Some system administrators use run levels to define which subsystems are working, e.g., whether X is running, whether the network is operational, and so on. Others have all subsystems always running or start and stop them individually, without changing run levels, since run levels are too coarse for controlling their systems. We need to decide for ourself, but it might be easiest to follow the way our Linux distribution does things.

The following table defines how most Linux Distributions define the different run levels. However, run-levels 2 through 5 can be modified to suit your own tastes.

Table 2.1 Run level numbers

0	Halt the system.
1	Single-user mode (for special administration).
2	Local Multi user with Networking but without network service (like NFS)
3	Full Multi user with Networking
4	Not Used
5	Full Multi user with Networking and GUI
6	Reboot.

Services that get started at a certain runtime are determined by the contents of the various rcN.d directories. Most distributions locate these directories either at /etc/init.d/rcN.d or /etc/rcN.d. (Replace the N with the run-level number.)

In each run-level we will find a series of if links pointing to start-up scripts located in /etc/init.d. The names of these links all start as either K or S, followed by a number. If the name of the link starts with an S, then that indicates the service will be started when you go into that run level. If the name of the link starts with a K, the service will be killed (if running).The number following the K or S indicates the order the scripts will be run. Here is a sample of what an /etc/init.d/rc3.d may look like.

```
# ls -l /etc/init.d/rc3.d
lrwxrwxrwx  1 root root 10 2004-11-29 22:09 K12nfsboot -> ../nfsboot
lrwxrwxrwx  1 root root  6 2005-03-29 13:42 K15xdm -> ../xdm
lrwxrwxrwx  1 root root  9 2004-11-29 22:08 S01pcmcia -> ../pcmcia
lrwxrwxrwx  1 root root  9 2004-11-29 22:06 S01random -> ../random
lrwxrwxrwx  1 root root 11 2005-03-01 11:56 S02firewall -> ../firewall
lrwxrwxrwx  1 root root 10 2004-11-29 22:34 S05network -> ../network
lrwxrwxrwx  1 root root  9 2004-11-29 22:07 S06syslog -> ../syslog
lrwxrwxrwx  1 root root 10 2004-11-29 22:09 S08portmap -> ../portmap
lrwxrwxrwx  1 root root  9 2004-11-29 22:07 S08resmgr -> ../resmgr
lrwxrwxrwx  1 root root  6 2004-11-29 22:09 S10nfs -> ../nfs
lrwxrwxrwx  1 root root 12 2004-11-29 22:40 S12alsasound -> ../alsasound
lrwxrwxrwx  1 root root  8 2004-11-29 22:09 S12fbset -> ../fbset
lrwxrwxrwx  1 root root  7 2004-11-29 22:10 S12sshd -> ../sshd
lrwxrwxrwx  1 root root  8 2005-02-01 09:24 S12xntpd -> ../xntpd
lrwxrwxrwx  1 root root  7 2004-12-02 20:34 S13cups -> ../cups
lrwxrwxrwx  1 root root  6 2004-11-29 22:09 S13kbd -> ../kbd
lrwxrwxrwx  1 root root 13 2004-11-29 22:10 S13powersaved -> ../powersaved
lrwxrwxrwx  1 root root  9 2004-11-29 22:09 S14hwscan -> ../hwscan
lrwxrwxrwx  1 root root  7 2004-11-29 22:10 S14nscd -> ../nscd
lrwxrwxrwx  1 root root 10 2004-11-29 22:10 S14postfix -> ../postfix
lrwxrwxrwx  1 root root  6 2005-02-04 13:27 S14smb -> ../smb
lrwxrwxrwx  1 root root  7 2004-11-29 22:10 S15cron -> ../cron
lrwxrwxrwx  1 root root  8 2004-12-22 20:35 S15smbfs -> ../smbfs
```

Fig. 2.33 Sample RC Script file

How run levels start are configured in /etc/inittab by lines like the following:

```
l2:2:wait:/etc/init.d/rc 2
```

The first field is an arbitrary label, the second one means that this applies for run level 2. The third field means that init should run the command in the fourth field once, when the run level is entered, and that init should wait for it to complete. The /etc/init.d/rc command runs whatever commands are necessary to start and stop services to enter run level 2.

The command in the fourth field does all the hard work of setting up a run level. It starts services that aren't already running, and stops services that shouldn't be running in the new run level any more. Exactly what the command is, and how run levels are configured, depends on the Linux distribution.

When **init** starts, it looks for a line in /etc/inittab that specifies the default run level:

```
id:2:initdefault
```

We can ask **init** to go to a non-default run level at startup by giving the kernel a command line argument of single or emergency. Kernel command line arguments can be given via LILO, for example. This allows us to choose the single user mode (run level 1).

While the system is running, the **telinit** command can change the run level. When the run level is changed, **init** runs the relevant command from /etc/inittab.

Special configuration in /etc/inittab

The /etc/inittab has some special features that allow **init** to react to special circumstances. These special features are marked by special keywords in the third field. Some examples:

- **powerwait**

  Allows init to shut the system down, when the power fails. This assumes the use of a UPS, and software that watches the UPS and informs init that the power is off.

- **ctrl-alt-del**

  Allows **init** to reboot the system, when the user presses ctrl-alt-del on the console keyboard. Note that the system administrator can configure the reaction to ctrl-alt-del to be something else instead, e.g., to be ignored, if the system is in a public location. (Or to start **nethack**.)

- **sysinit**

  Command to be run when the system is booted. This command usually cleans up /tmp, for example.

Booting in single user mode

An important run level is single user mode (run level 1), in which only the system administrator is using the machine and as few system services, including logins, as possible are running. Single user mode is necessary for a few administrative tasks, such as running **fsck** on a /usr partition, since this requires that the partition be unmounted, and that can't happen, unless just about all system services are killed.

A running system can be taken to single user mode by using **telinit** to request run level 1. At bootup, it can be entered by giving the word single or emergency on the kernel command line: the kernel gives the command line to **init** as well, and **init** understands from that word that it

shouldn't use the default run level. The kernel command line is entered in a way that depends on how we boot the system.

Booting into single user mode is sometimes necessary so that one can run **fsck** by hand, before anything mounts or otherwise touches a broken /usr partition any activity on a broken filesystem is likely to break it more, so **fsck** should be run as soon as possible.

The bootup scripts **init** runs will automatically enter single user mode, if the automatic **fsck** at bootup fails. This is an attempt to prevent the system from using a filesystem that is so broken that **fsck** can't fix it automatically. Such breakage is relatively rare, and usually involves a broken hard disk or an experimental kernel release, but it's good to be prepared.

As a security measure, a properly configured system will ask for the root password before starting the shell in single user mode. Otherwise, it would be simple to just enter a suitable line to LILO to get in as root. This will break if /etc/passwd has been broken by filesystem problems, of course, and in that case you'd better have a boot floppy handy.

2.7 Understanding Linux File System Structure

A filesystem 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 filesystem. Thus, one might say “I have two filesystems” meaning one has two partitions on which one stores files, or that one is using the “extended filesystem”, meaning the type of the filesystem.

The difference between a disk or partition and the filesystem it contains is important. A few programs (including, reasonably enough, programs that create filesystems) operate directly on the raw sectors of a disk or partition; if there is an existing file system there it will be destroyed or seriously corrupted. Most programs operate on a filesystem, and therefore won't work on a partition that doesn't contain one (or that contains one of the wrong type).

Before a partition or disk can be used as a filesystem, it needs to be initialized, and the bookkeeping data structures need to be written to the disk. This process is called making a filesystem.

Most UNIX filesystem types have a similar general structure, although the exact details vary quite a bit. The central concepts are superblock, inode, data block, directory block, and indirection block. The superblock contains information about the filesystem as a whole, such as its size whereas the exact information here depends on the filesystem. 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.

UNIX filesystems usually allow us to create a hole in a file (this is done with the lseek() system call), which means that the filesystem just pretends that at a particular place in the file there is just zero bytes, but no actual disk sectors are reserved for that place in the file (this means that the file will use a bit less disk space). This happens especially often for small binaries, Linux shared libraries, some databases, and a few other special cases. Holes are implemented by storing a

special value as the address of the data block in the indirect block or inode. This special address means that no data block is allocated for that part of the file, ergo, there is a hole in the file.

Linux supports several types of file systems. As of this writing the most important ones are:

MINIX

The oldest, presumed to be the most reliable, but quite limited in features some time stamps are missing, at most 30 character filenames and restricted in capabilities (at most 64 MB per filesystem).

xia

A modified version of the minix filesystem that lifts the limits on the filenames and filesystem sizes, but does not otherwise introduce new features. It is not very popular, but is reported to work very well.

ext3

The ext3 filesystem has all the features of the ext2 filesystem. The difference is, journaling has been added. This improves performance and recovery time in case of a system crash. This has become more popular than ext2.

ext2

The most featureful of the native Linux filesystems. It is designed to be easily upwards compatible, so that new versions of the filesystem code do not require re-making the existing filesystems.

ext

An older version of ext2 that wasn't upwards compatible. It is hardly ever used in new installations any more, and most people have converted to ext2.

reiserfs

A more robust filesystem. Journaling is used which makes data loss less likely. Journaling is a mechanism whereby a record is kept of transaction which are to be performed, or which have been performed. This allows the filesystem to reconstruct itself fairly easily after damage caused by, for example, improper shutdowns.

jfs

JFS is a journaled filesystem designed by IBM to to work in high performance environments.

xfs

XFS was originally designed by Silicon Graphics to work as a 64-bit journaled filesystem. XFS was also designed to maintain high performance with large files and filesystems. In addition, support for several foreign filesystems exists, to make it easier to exchange files with other operating systems. These foreign filesystems work just like native ones, except that they may be lacking in some usual UNIX features, or have curious limitations, or other oddities.

MS-DOS

Compatibility with MS-DOS (and OS/2 and Windows NT) FAT filesystems.

msdos

Extends the msdos filesystem driver under Linux to get long filenames, owners, permissions, links, and device files. This allows a normal msdos filesystem to be used as if it were a Linux one, thus removing the need for a separate partition for Linux.

vfat

This is an extension of the FAT filesystem known as FAT32. It supports larger disk sizes than FAT. Most MS Windows disks are vfat.

iso9660

The standard CD-ROM filesystem; the popular Rock Ridge extension to the CD-ROM standard that allows longer file names is supported automatically.

nfs

A networked filesystem that allows sharing a filesystem between many computers to allow easy access to the files from all of them.

smbfs

A networks filesystem which allows sharing of a filesystem with an MS Windows computer. It is compatible with the Windows file sharing protocols.

hpfs

This is the OS/2 filesystem.

sysv

SystemV/386, Coherent, and Xenix filesystems.

NTFS

The most advanced Microsoft journaled filesystem providing faster file access and stability over previous Microsoft filesystems.

The choice of filesystem to use depends on the situation. If compatibility or other reasons make one of the non-native filesystems necessary, then that one must be used. If one can choose freely, then it is probably wisest to use ext3, since it has all the features of ext2, and is a journaled filesystem.

There is also the proc filesystem, usually accessible as the /proc directory, which is not really a filesystem at all, even though it looks like one. The proc filesystem makes it easy to access certain kernel data structures, such as the process list (hence the name). It makes these data structures look like a filesystem, and that filesystem can be manipulated with all the usual file tools. For example, to get a listing of all processes one might use the command

```
$ ls -l /proc
total 0
dr-xr-xr-x   4 root     root            0 Jan 31 20:37 1
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 63
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 94
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 95
dr-xr-xr-x   4 root     users           0 Jan 31 20:37 98
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 99
-r--r--r--   1 root     root            0 Jan 31 20:37 devices
-r--r--r--   1 root     root            0 Jan 31 20:37 dma
-r--r--r--   1 root     root            0 Jan 31 20:37 filesystems
-r--r--r--   1 root     root            0 Jan 31 20:37 interrupts
-r--------   1 root     root      8654848 Jan 31 20:37 kcore
-r--r--r--   1 root     root            0 Jan 31 11:50 kmsg
-r--r--r--   1 root     root            0 Jan 31 20:37 ksyms
-r--r--r--   1 root     root            0 Jan 31 11:51 loadavg
-r--r--r--   1 root     root            0 Jan 31 20:37 meminfo
-r--r--r--   1 root     root            0 Jan 31 20:37 modules
dr-xr-xr-x   2 root     root            0 Jan 31 20:37 net
dr-xr-xr-x   4 root     root            0 Jan 31 20:37 self
-r--r--r--   1 root     root            0 Jan 31 20:37 stat
-r--r--r--   1 root     root            0 Jan 31 20:37 uptime
-r--r--r--   1 root     root            0 Jan 31 20:37
version
$
```

Fig. 2.34 The proc file system

There will be a few extra files that don't correspond to processes, though. The above example has been shortened.

Note that even though it is called a filesystem, no part of the proc filesystem touches any disk. It exists only in the kernel's imagination. Whenever anyone tries to look at any part of the proc filesystem, the kernel makes it look as if the part existed somewhere, even though it doesn't. So, even though there is a multi-megabyte /proc/kcore file, it doesn't take any disk space.

Which filesystem should be used ?

There is usually little point in using many different filesystems. Currently, ext3 is the most popular filesystem, because it is a journaled filesystem. Currently it is probably the wisest choice. Reiserfs is another popular choice because it to is journaled. Depending on the overhead for bookkeeping structures, speed, reliability, compatibility, and various other reasons, it may be advisable to use another file system. This needs to be decided on a case-by-case basis.

A filesystem that uses journaling is also called a journaled filesystem. A journaled filesystem maintains a log, or journal, of what has happened on a filesystem. In the event of a system crash, or if your two year old son hits the power button like mine loves to do, a journaled filesystem is designed to use the filesystem's logs to recreate unsaved and lost data. This makes data loss much less likely and will likely become a standard feature in Linux filesystems. However, do not get a false sense of security from this. Like everything else, errors can arise. Always make sure to back up your data in the event of an emergency.

Filesystem comparison

Table 2.2 Filesystems properties.

FS Name	Year Introduced	Original OS	Max File Size	Max FS Size	Journaling
FAT16	1983	MSDOS V2	4GB	16MB to 8GB	N
FAT32	1997	Windows 95	4GB	8GB to 2TB	N
HPFS	1988	OS/2	4GB	2TB	N
NTFS	1993	Windows NT	16EB	16EB	Y
HFS+	1998	Mac OS	8EB	?	N
UFS2	2002	FreeBSD	512GB to 32PB	1YB	N
ext2	1993	Linux	16GB to 2TB4	2TB to 32TB	N
ext3	1999	Linux	16GB to 2TB4	2TB to 32TB	Y
ReiserFS3	2001	Linux	8TB8	16TB	Y
ReiserFS4	2005	Linux	?	?	Y
XFS	1994	IRIX	9EB	9EB	Y
JFS	?	AIX	8EB	512TB to 4PB	Y
VxFS	1991	SVR4.0	16EB	?	Y
ZFS	2004	Solaris 10	1YB	16EB	N

References:

1. Red Hat Linux Networking & System Administration, by Terry Collings, Kurt Wall, 3rd Edition, 2005, Wiley Publishing
2. Red Hat RHCSA/RHCE 7 Cert Guide: Red Hat Enterprise Linux 7 (EX200 and EX300) (Certification Guide), 2015 by Sander van Vugt, Pearson IT Certification
3. https://vmware.com
4. https://opensource.com
5. https://tldp.org

Introduction

Any operating system is act as interface between hardware and user in other word it is collection of program with provide basic utilities to the user ,if you want to use hardware to do some task you need to tell to the operating system what you want to do ? For example it may be addition of two number, opening some file or software etc. Then question will come in your mind, how to talk with operating system?

The answer is you need to give command to the operating system and the operating system program which will help us to give this command to an operating system is Shell. In other word shell is interface through which we can use operating system services. A shell hides all complex details of the operating system specifically kernel, which is the lowest-level or core component of any operating systems.

Bash Shell

Bash is a UNIX shell and command language written by Brian Fox in 1989, It offers more functions over sh shell for both types of user i.e. programmer and end user. In addition, most sh scripts can be run by Bash as it is. The important features of Bash include: Command line editing, Unlimited size command history ,Job Control, Shell Functions and Aliases ,Indexed arrays of unlimited size, Integer arithmetic in any base from two to sixty-four .it has been used widely as the default login shell for many Linux distributions .

Useful Bash Key Sequences Sometimes, you will enter a command from the Bash command line and nothing, or something totally unexpected, will happen. If that occurs, it is good to know that some key sequences are available to perform basic Bash management tasks. Here is a short list of the most useful of these key sequences:

Ctrl+C Use this key sequence to quit a command that is not responding (or simply is taking too long to complete). This key sequence works in most scenarios where the command is active and producing screen output.

Ctrl+D This key sequence is used to send the end-of-file (EOF) signal to a command. Use this when the command is waiting for more input. It will indicate this by displaying the secondary prompt >.

Ctrl+R This is the reverse search feature. When used, it will open the reverse-i-search prompt. This feature helps you locate commands you have used previously. The feature is especially useful when working with longer commands. Type the first characters of the command, and you will immediately see the last command you used that started with the same characters.

Ctrl+Z Some people use Ctrl+Z to stop a command. In fact, it does stop your command, but it does not terminate it. A command that is interrupted with Ctrl+Z is just halted until it is started again with the fg command as a foreground job or with the bg command as a background job.

Ctrl+A This keystroke brings the cursor to the beginning of the current command line.

Ctrl+B This moves the cursor to the end of the current command line

Working with basic Linux command

Linux have hundreds of commands we can categorized them. Some of Categorized as follows

System Related Commands: These commands are used to view and manage Linux system-related information,

Hardware Related Commands: These commands are used to view and manage hardware-related aspects of the Linux machine,

Statistic Related Commands: These set of commands are used to view various kinds of stats of the Linux system,

User-Related Commands: These commands are used to manage Linux users,

File Related Commands: These commands are used to handle files and directories ,

Process Related Commands: These commands are used to handle Linux processes,

File Permission Related Commands: These commands are used to change permissions of the files,

Network Related Commands: These commands are used to view and edit network configurations related aspects of the system

And many more

To execute command you need to open the terminal it also known as console the short cut key to open the terminal is The combination of three keys Ctrl+Alt+T or ctrl-t depending upon the linux distort we are using. Let us learn some basic commands.

1. Pwd :- sometimes you may want to know where exactly you are. Linux pwd is a command to print the name of current/working directory. When we are “lost” into a deep directory, we can always reveal where we are.

```
sh-4.4$ pwd
/home/cg/root
```

2. ls :- If you are a Linux user, there will be not a single day that you have not used ls command. A very simple and powerful command used to list files and directories from current directory.

```
sh-4.4$ ls
README.txt  imp.txt  test.c
```

Using -l character (small L letter), will display a long list the content of current directory which contains not only the name of the file, but also owner, group owner, link count, permissions Of each file.

```
sh-4.4$ ls -l
total 4
-rw-r--r-- 1 23627 23627 978 Jul 18 09:37 README.txt
-rw-r--r-- 1 23627 23627   0 Jul 18 10:07 imp.txt
-rw-r--r-- 1 23627 23627   0 Jul 18 10:07 test.c
```

In Linux, a file begins with **"." (dot sign)** is a hidden file. To show it on ls command, we can use **-a** parameter.

```
sh-4.4$ ls -a
.  ..  .cg_conf  README.txt  imp.txt  test.c
```

3. mkdir :- To create directory on Linux, we can use mkdir command. mkdir is short name for "make directory". By default, running mkdir and name of directory you want to create without any parameter will create a directory under the current directory.

```
sh-4.4$ ls
New  README.txt  imp.txt  test.c
sh-4.4$ mkdir newDirectory
sh-4.4$ ls
New  README.txt  imp.txt  newDirectory  test.c
```

We can also create multiple directories at the same time. Let say we want to create directories named **dir1, dir2 and dir3.**

```
sh-4.4$ ls
New  README.txt  imp.txt  newDirectory  test.c
sh-4.4$ mkdir dir1 dir2 dir3
sh-4.4$ ls
New  README.txt  dir1  dir2  dir3  imp.txt  newDirectory  tes
```

If you want to create sub-directories, you will need to use **-p** parameter. This parameter will create parent directory first, if it cannot find it. And it will create sub directory in it .let us see we don't have directory with name test we try to create sub directory in it with name sub.

```
sh-4.4$ ls
New  README.txt  dir1  dir2  dir3  imp.txt  newDirectory  test.c
sh-4.4$ mkdir -p test/sub
sh-4.4$ ls
New  README.txt  dir1  dir2  dir3  imp.txt  newDirectory  test  te
sh-4.4$ ls test
sub
```

4. stat :- stat is command which will give memory storage information about file and last access information of this file following demo gives information about MyFile.txt.

```
sh-4.4$ stat MyFile.txt
  File: MyFile.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty
Device: 811h/2065d      Inode: 13895316    Links: 1
Access: (0644/-rw-r--r--)  Uid: (25230/ UNKNOWN)   Gid: (25230/ UNKNOWN)
Access: 2019-07-18 13:47:35.805960094 +0000
Modify: 2019-07-18 13:47:35.805960094 +0000
Change: 2019-07-18 13:47:35.805960094 +0000
 Birth: -
```

5. touch :- This command is used for Manipulating the timestamps of files, Create an empty file , Create a file with a particular timestamp ,to trigger a rebuild of code.

To create empty file use touch command with name

```
sh-4.4$ touch MyFile.txt
sh-4.4$ ls
MyFile.txt  README.txt
```

The other options like –a , -m are used to modify access and modify time of any file.

6. rm :- When a file no longer needed, if we want delete it to save storage space. we can use **rm** command . rm is a command to delete a file or directory.

```
sh-4.4$ ls
MyFile.txt  README.txt  demo.txt  pp.txt  prog.c  test.
sh-4.4$ rm pp.txt
sh-4.4$ ls
MyFile.txt  README.txt  demo.txt  prog.c  test.txt
sh-4.4$
```

You can remove all file with same extension as follows. In follow example we will remove all file with .c extension.

```
sh-4.4$ ls
MyFile.txt  README.txt  array.c  demo.txt  fact.c  prog.c  struct.c  tes
sh-4.4$ rm *.c
sh-4.4$ ls
MyFile.txt  README.txt  demo.txt  test.txt
sh-4.4$
```

7. **who** command is a tool print information about users who are currently logged in. On most Linux distribution, who command is already installed. to use it, just type **who** on your console.

```
sh-4.4$ who
pungki   tty7         2013-12-20 20:18 (:0)
pungki   pts/0        2013-12-21 06:02 (:0.0)
leni     pts/2        2013-12-21 06:50 (192.168.0.108)
```

8. **alias :-** The alias command lets us give our own name to a command or sequence of commands. We can then type our short name, and the shell will execute the command or sequence of commands for us. In the following example we have created alias named myCommand for ls command so both my and ls command gives us same output.

```
sh-4.4$ ls
README.txt  aone.txt  factorial.c  linux1.txt  linux2.txt  object.cpp  python.py
sh-4.4$ alias myCommand=ls
sh-4.4$ myCommand
README.txt  aone.txt  factorial.c  linux1.txt  linux2.txt  object.cpp  python.py
sh-4.4$
```

9. **Cat :-**The cat command (short for "concatenate") lists the contents of files to the terminal window. If we specify only one file name with it if we specify two file name with cat command it will show content of both file in concatenate form to do actual concatenation of two file we need to use redirection that we will see in coming section. This is faster than opening the file in an editor, and there's no chance of accidentally altering the file.

```
sh-4.4$ cat aone.txt
**** File aone.txt is concatenated****
sh-4.4$ cat linux1.txt
What is Linux?
Linux is the best-known and most-used open source operating system. As an operating system, Linux is software that
derneath all of the other software on a computer, receiving requests from those programs and relaying these reques
e computer's hardware.

For the purposes of this page, we use the term "Linux" to refer to the Linux kernel, but also the set of programs
and services that are typically bundled together with the Linux kernel to provide all of the necessary components
ly functional operating system. Some people, particularly members of the Free Software Foundation, refer to this c
n as GNU/Linux, because many of the tools included are GNU components. However, not all Linux installations use GN
ents as a part of their operating system. Android, for example, uses a Linux kernel but relies very little on GNU
-4.4$ cat linux1.txt aone.txt
What is Linux?
Linux is the best-known and most-used open source operating system. As an operating system, Linux is software that
derneath all of the other software on a computer, receiving requests from those programs and relaying these reques
e computer's hardware.

For the purposes of this page, we use the term "Linux" to refer to the Linux kernel, but also the set of programs
and services that are typically bundled together with the Linux kernel to provide all of the necessary components
ly functional operating system. Some people, particularly members of the Free Software Foundation, refer to this c
n as GNU/Linux, because many of the tools included are GNU components. However, not all Linux installations use GN
ents as a part of their operating system. Android, for example, uses a Linux kernel but relies very little on GNU
** File aone.txt is concatenated****
sh-4.4$
```

10. **Cd :-** While working Linux console we may need to get into specific directory that is we need to change our current working directory. the command used for it is cd in the following example we are changeing from root to mydir directory.

```
sh-4.4$ ls
README.txt  aone.txt  factorial.c  linux1.txt  linux2.txt  mydir  object.cpp  pytho
sh-4.4$ cd mydir
sh-4.4$ ls
sh-4.4$
```

11. **Echo :-** The echo command prints (echoes) a string of text to the terminal window.The command below will print the words "Welcome to Linux" on the terminal window.

```
sh-4.4$ echo Welcome to Linux
Welcome to Linux
sh-4.4$
```

12. Man :- The most important source information available for use of Linux commands is man, which is short for the system programmer's "manual." This command will help you to find all information about any command on linux let us see the following example , using man command we cand find the information about rm command.

man rm

Now we have seen different command and their use, so we are familiar with how to execute command on Linux. Linux command set is too big so following is the list of few commands with its use. And if you still wants to know more about any of the command you have man command with you.

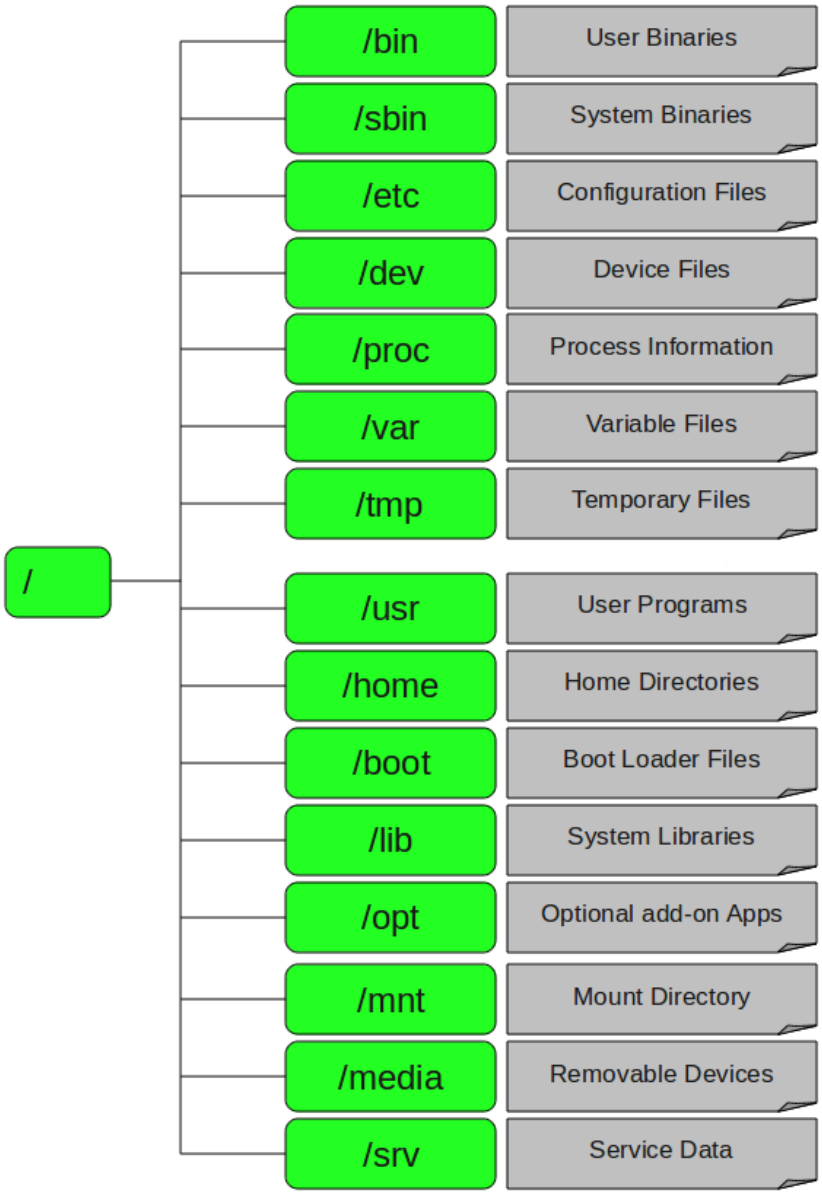
Command	use
uname	Displays linux system information. With -a switch you can view all the information, with -r switch you can view kernel release information and with -o you can view OS information
cat /etc/redhat_release	Shows which version of redhat installed
uptime	Shows how long the system has been running
hostname :	Shows system host name. With -i switch you can view the ip address of the machine and with -d you can view the domain name
last reboot	Shows system reboot history
date	Shows the current date and time. You can specify the format you want to view the date as well. As an example, by using 'date +%D' you can view the date in 'MM/DD/YY' format
cal	Shows the calendar of the current month. With -y switch you can view the calendar of the whole current year
w	Displays who is logged on and what they are doing
whoami	Shows current user id

finger user	Displays information about user
reboot	Reboots the system
shutdown	Shuts down the system
dmesg	Displays all the messages from Kernel ring buffer. With -k switch you can view kernel messages and with -u you can view userspace messages
lshw	Displays information on hardware configuration of the system. But this command must be run as super user or it will only report partial information
lsblk	Displays block device related information of the machine. With -a you can view all block devices
free -m	Shows used and free memory (-m for MB)
lsusb -tv	Shows information on USB devices
dmidecode	Shows hardware info from the BIOS (vendor details)
mpstat	Displays processors related statistics
vmstat	Displays virtual memory statistics
iostat	Displays I/O statistics
lsof	Lists all open files belonging to all active processes
lsof -u testuser	Lists files opened by a specific user

who	Shows who is logged on the system
ps	Displays your currently active processes
kill pid	Kills process with mentioned pid
killall proc	Kills all processes named proc
grep pattern files	Searches for pattern in files
grep -r pattern dir	Searches recursively for pattern in dir
top	Display all running processes and cpu/memory usage

Working with Directories

In linux every program has some pre decided location to store it. let say some programs are located under /bin, or some in /sbin, or /usr/bin etc. you may ask question why this is so? What is the different between all these directories? let us understand the Linux file system structures and understand the meaning and use of every high-level directories. linux has tree like file system hierarchy



1. / – Root

- On every file and directory starts from the root directory the top most directory in file hierarchy root directory.
- Only the root user of the system has written right under this directory.
- Please note that /root is root user's home directory, which is not same as /.

2. /bin – User Binaries

- This directory contains binary executables.
- Common linux commands you use in single-user modes are to be found under this directory.
- Commands used by all the users of the system are located in this directory.
- For example: ls, cp,ping, grep.

3. /sbin – System Binaries

- Same as /bin, /sbin also contains binary executables.
- But, the linux commands located under this directory are used usually by system administrator, for system maintenance.
- For example: iptables, reboot, fdisk, ifconfig, swapon

4. /etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains start up and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

5. /dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

6. /proc – Process Information

- Contains information about system process.
- This is a pseudo file system contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual file system with text information about system resources. For example: /proc/uptime

7. /var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

8. /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

9. /usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can’t find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp

- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. /home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

11. /boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*
- For example: ld-2.11.1.so, libncurses.so.5.7

13. /opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

14. /mnt – Mount Directory

- Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

16. /srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data

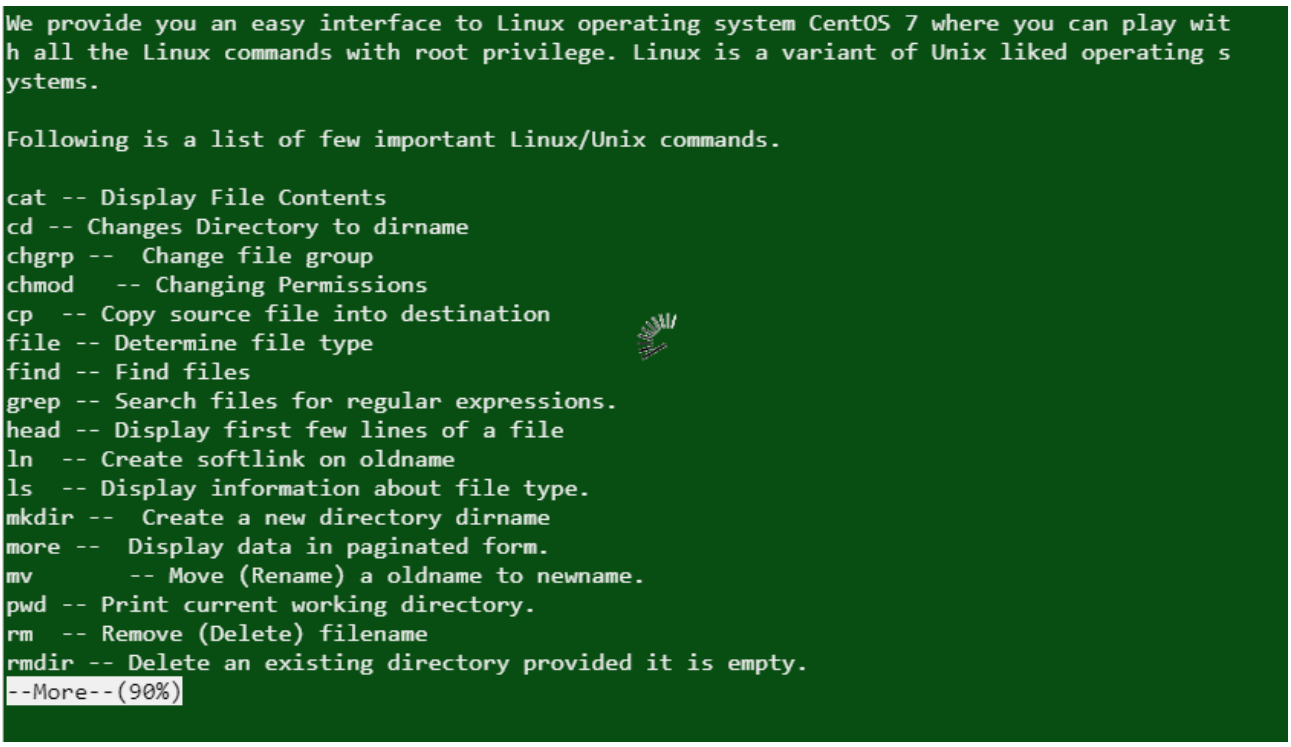
We have studied some of the file and directory handling command like mkdir,ls,touch etc in previous section. Let us now learn some more file and directory handling command.

More :- let consider that you are working with Linux, you will find a many file in text format in Linux . like Configuration files and log files are always kept as text file. But

those files usually has hough content. You can not view them all at time in one page. So we need pagination to those files. And to achieve this, we can use Linux more command. More command is a command for displaying a long text file per page at a time. Let us consider we want to see our syllabus page by page the we can use more command as follows

more README.txt

The screen will appeared as follows



In the above screen the left corner 90% indicate that it is showing you 90% content of file by pressing enter you see the next lines.

When you run more command, it will fill your screen with the content of the file you are seeing using more. You can limit it into some lines for each page. To do this you can use **-num option**.

For example, you want to **limit the lines only 6 lines** for each page. Then you can type

more -6 README.txt

it will show the output as follows

```
sh-4.4$ more -6 README.txt
We provide you an easy interface to Linux operating system CentOS 7 where you can play wit
h all the Linux commands with root privilege. Linux is a variant of Unix liked operating s
ystems.

Following is a list of few important Linux/Unix commands.

--More--(25%)
```

If your file is long , it is difficult to find a string that you want. The search string can help you. Using **+/string** we can search a string for you and put the keyword on the beginning of the line. Let say we want to search **"touch"** in **README.txt** file. Then the syntax is :

```
sh-4.4$ more +/touch README.txt

...skipping
rmdir -- Delete an existing directory provided it is empty.
tail -- Prints last few lines in a file.
touch -- Update access and modification time of a file.
sh-4.4$
```

There are many option which we can use with more command, to know this use man pages.

Following are list of file and directory relater commands with its use.

command	Use
ls -al	Displays all information about files/directories. This includes displaying all hidden files as well
mv file1 file2	Moves files from one place to another/renames file1 to file2
head file	Display the first 10 lines of file
tail file	Outputs the last 10 lines of file
gpg -c file	Encrypts file
gpg file.gpg	Decrypts file
cksum file	View the checksum of the file
diff file1 file2	View the differences between contents of file1 and file2
ln -s target-file link-file	Create a soft link named link-file to target-file
sort	orts files in alphabetical order
uniq	Compares adjacent lines in a file and removes/reports any duplicate lines
wc	Counts number of words/lines
chmod octal file-name chmod 777 /test.c	Changes the permissions of file to octal Sets rwx permission for owner , group and others
cd ..	Goes up one level of the directory tree

cd	Goes to $HOME directory
cd /test	Changes to /test directory

Piping and Redirection

One of the most powerful features of the Linux command line is The piping and redirection options. Piping is used to send the result of a one command to another command, and redirection sends the output of a command to a file may or may not be a regular file, but it can also be a device file, let us see the following examples.

This example will help you to understand how a pipe is used to add functionality to a command. Now you will execute a command where the output does not fit on the screen. Then we will see how by piping this output through less, you can see the output screen by screen.

Now execute command in give sequence. Open a shell, and use su - to become the root. Enter the root password. once you login as root type the command ps aux and hit enter this command provides you list of all the processes that are currently running on your computer. You will notice that the list is too long and it does not fit on your computer screen. now to see the output in proper manner i.e. page by page pipe will be useful., use ps aux | less. And now the output of ps is now sent to less, which outputs it so that you can browse it page by page.

Another very useful command that is often used in a pipe creation is grep. It is used as a filter to show just the information that you want to see. For example, that you want to check whether a user with the name linda exists in the user database /etc/passwd. One solution is that open the file with any viewer like cat or less and then check the contents of the file for string you are seeking is present in the file or not. This is time consuming human error may come while searching the data there is a much easier reliable solution for this , pipe the contents of the file to the filter grep, which would choose all of the lines that contain the string mention as an argument of grep command. This command would read cat /etc/passwd | grep linda.

To understand more execute following set of command in given sequence.

You will use the ps aux command to show a list of all processes on your system, but this time you will pipe the output of the command through the grep command, which will selects the information you're seeking.

Type ps aux and hit the enter key to display the list of all the processes that are running on your computer. As we know it's not easy to find the exact information you need so now use ps aux | grep blue to select only the lines that contain the text blue. You'll now see two lines, one displaying the name of the grep command you used and another one showing you the name of the Bluetooth applet. Now execute next step, in this step you are going to make sure you don't see the grep command itself. To do this, the command

grep -v grep is added to the pipe. The grep option -v excludes all lines contain a specific string. The command you'll enter to get this result is ps aux | grep blue | grep -v grep.

Redirection

Redirection sends the result of the one command to a file. While this file can be a text file, it can also be a special file like device file. The following example will help you to understand how redirection is used to redirect the standard output (STDOUT), normally it is written to the current console to a file.

First you'll use the ps aux command without redirection. The results of the command will be written to the terminal window. Now let redirect the output of the command to a file. In the final step, you'll display the contents of the file using the less utility.

Execute command with given sequence.

From a console window, use the command ps aux. You will get the output on the current console. Now execute command ps aux > ~/psoutput.txt. You will not get the output of the command because it is redirected to a file that is created in home directory, which is nominated by the ~ sign. to show the contents of the psoutput file use the command less ~/psoutput.txt.

Let us see another way of redirection instead of redirecting output of commands to the files; the opposite is also possible with redirection. For example, you may send the content of a text file to a command that will use that content of the file as input. Let us execute following commands .

Open the console and type mail root. This opens the command-line mail program to send a message to the user root when mail prompts for a subject, type Test message as the subject text, and press Enter. Then mail command displays a blank line where you can type the message body. In a real message, here is where you would type your message, however, you don't need a message body, and you want to close the input immediately. To do this, type a . (dot) and press Enter. The mail message has now been sent to the user root. Now you're going to specify the subject as a command-line option using the command mail -s test message. The mail command immediately returns a blank line, where you'll enter a . (dot) again to tell the mail client that you're done. In the third attempt, you enter everything in one command, which is useful if you want to use commands like this in automated shell scripts. Type this command: mail -s test message < As you can see, when using redirection of the STDIN, the dot is fed to the mail command immediately, and you don't have to do anything else to send the message.

When using redirection, you should be aware that it is possible not only to redirect STDOUT and STDIN. Commands can also produce error output. This error output is technically referred to as STDERR. To redirect STDERR, use the 2> construction to indicate that you are interested only in redirecting error output. This means that you won't see errors anymore on your current console, which is very helpful if your command produces error messages as well as normal output. The next exercise

demonstrates how redirecting STDERR can be useful for commands that produce a lot of error messages.

Editing files with vi

If you want use vi understand that its work-alike editors is modality. Most programs or editor has very simple UI in which it accepting input and placing it at the cursor.
But vi has different modes. When you will start vi, you will be in “Normal” mode, which is actually a command mode. When you are in Normal mode, whatever you type is considered not to be input, but commands that vi will try to execute.

This may sound a little crazy, but it is actually a very powerful way to edit documents. Even if you don’t like it, but in Linux world is one of the most popular editor so you need to learn it,but on other hand if you enjoy working at a command line, then you may end up loving vi.

Another important reason why you should become familiar with vi is that some other commands are based on it. For example, to edit quota for the end users on your server, you would use command edquota, which is a macro built on vi. If you want to set permissions for the sudo command, use visudo, which, as you can guess, is also a macro built on top of vi.

Let us start learn how to use Vi , type vi and hit the enter key if you want to open specific file from cuttent working directory type that file name with vi to open this file , After starting a vi editor , as discussed before we can start entering text there is the command mode, which is used to enter new commands. The vi offers you a lot of choices. you can choose between a number of methods to enter insert mode. Use i to insert text at the current cursor position. Use a to append text after the current position of the cursor. Use o to open a new line under the current position of the cursor , Use O to open a new line above the current position of the cursor. After entering insert mode, you can enter text, and vi will work as any other editor

To save your work, go back to command mode and use the appropriate commands. The magic key to go back to the command mode from insert mode is Esc.

Saving and Quitting

After coming back to command mode, you need to use the appropriate command to save your work. The most commonly used command is :wq! , To exit vi without saving changes, hit Escape ensure you are in Normal mode, and then type :q!

Using an ! at the end of a command is potentially dangerous; if a previous file with the same name already exists, vi will overwrite it without any further warning. Because ! Mark after command q ot wq tell to the command line interpreter to do the task without give any warning message.

Cut, Copy, and Paste

When it comes to editing any file the tree most useful operation we will perform are cut ,copy ,paste.To cut and copy the contents of a file is easy, you can use the v command, which enters visual mode. In visual mode, you can select a block of text. After selecting

the block, you can cut, copy, and paste it. Use d to cut the selection text. Cut command will remove the selection and place it in a buffer memory. Use y to copy the selection content to the selected area reserved for that purpose in your server's memory. Use p to paste the selection under the current line, or use P if you want to paste it above the current line. This will copy the selection you have just placed in the reserved area of your server's memory back into your document. For this purpose, it will always use your cursor's current position.

Deleting Text

Another action you will often do when working with vi is deleting text. There are many methods that can be used to delete text with vi. The easiest is from insert mode: just use the Delete and Backspace keys to get rid of any text you like. This works just like a word processor. Some options are available from vi command mode as well. Use x to delete a single character. This has the same effect as using the Delete key while in insert mode. Use dw to delete the rest of the word. That is, dw will delete anything from the current position of the cursor to the end of the word. Use D to delete from the current cursor position up to the end of the line.Use dd to delete a complete line.

Managing Software

Red Hat Enterprise Linux and many other Linux distributions group their software together in packages and they referred it as RPM Package Manager (RPM). The "R" in RPM originally stood for "Red Hat" but now changed to the recursive "RPM"

Understanding RPM

When Linux was first design, most of the software used in Linux systems was passed around in tar balls. A tar ball is a single archive file (created using the tar command) that can contain multiple files that need to be installed. Unfortunately, there were no rules for what needed to be in the tar ball neither there was any specifications of how the software in the tar ball was to be installed. Working with tar balls was not convenient for several reasons no standardization is one of the them. When using tar balls, there was no way to track what was installed. An updating and de-installing tar ball was much difficult. There are different issues, the tar ball contained source files that still needed to be compiled, and in some other case the tar ball had a nice installation script. Or somewhere the tar ball would just include a bunch of files including a README file explaining what to do with the software. The ability to trace software was needed to overcome the disadvantages of tar balls. The Red Hat Package Manager (RPM) is one of the standards designed to overcome the drawback of tar ball. An RPM is an archive file. It is created using command cpio. However, it's no regular archive. With RPM, there is also metadata describing what this package contains and it also contain one more important information that where those different files should be installed. This well organized design of RPM makes it is easy for Linux administrator to query exactly what is happening in it. Another benefit of using RPM is that its database is created in the /var/lib/rpm directory. This database keeps track of the exact version of files that are installed on the computer. Thus, for an administrator, it is possible to query individual RPM files to see their contents.

You can also query the database to see where a specific file comes from or what exactly is in the RPM.

Understanding Meta Package Handlers

RPM is efficient in managing software but there is still one inconvenience that must be dealt with software dependency. To standardize the software, many programs used on Linux use libraries and other common components provided by other software packages. That means before install one package, there are some other packages required to be present on the system. This is known as a software dependency. We need to take into consideration that the package which we are installing as dependency package which may be depends on other package to get installed. In the installation process if system does not find required packages it will show the "Failed dependencies" message .Though working with common components provided from other packages is a good thing even if only for the uniformity of appearance of a Linux distribution in practice doing so could lead to real problems.

The Meta Package Handler is solution for this dependency hell. Meta Package Handler in Red Hat is known as yum (Yellowdog Update Manager), this works with repositories, which are the installation sources that are consulted whenever a user wants to install a software package. In the repositories, all software packages of your distribution are typically available. While installing a software package using yum install some package, yum first checks whether there are any dependencies. If there are, yum checks the repositories to see whether the required software is available in the repositories, and if it is, the administrator will see a list of software dependencies that yum wants to install. So, this is how yum is resolving the problem of dependency hell.

Creating Your Own Repositories

If you don't have Red Hat server installed then doesn't have access to the official R H N (Red Hat Network) repositories, in this case you will need to create your own repositories. This procedure is also useful if you want to copy all of your RPM's to a directory and use that directory as a repository. Let us see how to do this

Let us preparer our system to make your own repositories, copy all of the RPM files from the Red Hat installation DVD to a directory that you will create on disk. Next you will install and run the createrepo package and its dependencies. This package is used to create the metadata that yum uses while installing the software packages. While installing the createrepo package, you will see that some dependency problems have to be handled as well.

1. Use mkdir /repo to create a directory that you can use as a repository in the root of your server's fi le system.

2. Insert the Red Hat installation DVD in the optical drive of your server. Assuming that you run the server in graphical mode, the DVD will be mounted automatically.

3. Use the cd /media/RHEL[Tab] command to go into the mounted DVD. Next use cd Packages, which brings you to the directory where all RPMs are by default. Now use cp * /repo to copy all of them to the /repo directory you just created. Once this is finished, you don't need the DVD anymore.

4. Now use cd /repo to go to the /repo directory. From this directory, type rpm -ivh createrepo. This doesn't work, and it gives you a " Failed dependencies" error. To install createrepo, you fi rst need to install the deltarpm and python-deltarpm packages. Use rpm -ivh deltarpm python-deltarpm to install both of them. Next, use rpm -ivh createrepo again to install the createrepo package.

5. Once the createrepo package has been installed, use createrepo /repo, which creates the metadata that allows you to use the /repo directory as a repository. This will take a few minutes. When this procedure is fi nished, your repository is ready for use.

Managing Repositories

In the preceding section, you learned how to turn a directory that contains RPM s into a repository, just marking a directory as a repository is not sufficient. To use your newly created repository, you need to tell your server where it can find this repository and for this, you need to create a repository file in the directory /etc/yum.repos.d. You will probably already have some repository files in this directory. You can see the content of the rhel-source. repo file that is created by default.

```
[root@hnl ~]# cat /etc/yum.repos.d/rhel-source.repo
[rhel-source]
name=Red Hat Enterprise Linux $releasever - $basearch - Source
baseurl=ftp://ftp.redhat.com/pub/redhat/linux/enterprise/$releasever/en/os/SRPMS/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
[rhel-source-beta]
name=Red Hat Enterprise Linux $releasever Beta - $basearch - Source
baseurl=ftp://ftp.redhat.com/pub/redhat/linux/beta/$releasever/en/os/SRPMS/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-beta,file:///etc/pki/rpm
-gpg/RPM-GPG-KEY-redhat-release
[root@hnl ~]#
```

See the above execution. You will find all elements that a repository file should contain. First, between square brackets there is an identifier for the repository. It is immaterial what you use here; the identifier helps you to identify the repository. The same goes for the name parameter it gives a name to the repository. The important parameter is baseurl. It will help where the repository can be found in URL format. As you can see in this

example, an FTP server at Red Hat is specified. Alternatively, you can also use URLs that refer to a website or to a directory that is local on your server's hard drive. In the latter case, the repository format looks like file:///yourrepository. Some people are confused about the third slash in the URL, but it really has to be there. The file:// part is the URI, which tells yum that it has to look at a fi le, and after that, you need a complete path to the file or directory, which in this case is /yourrepository. Next the parameter enabled specifies whether this repository is enabled. A 0 indicates that it is not, and if you really want to use this repository, this parameter should have 1 as its value. The last part of the repository specifies if a GPG file is available. Because RPM packages are installed as root and can contain scripts that will be executed as root without any warning, it really is important that you are confident that the RPM. you are installing can be trusted. GPG helps in guaranteeing the integrity of software packages you are installing. To check whether packages have been tampered with, a GPG check is done on each package that you'll install.

To do this check, you need the GPG files installed locally on your computer. As you can see, some GPG files that are used by Red H at are installed on your computer by default. Their location is specified using the gpgkey option. The other option gpgcheck=1 tells yum that it has to perform the GPG integrity check. If you're having a hard time confi guring the GPG check, you can change this parameter to gpgcheck=0, which completely disables the GPG check for RPM s that are found in this repository.

Working with yum

Yellowdog Updater Modified in short called as YUM is a package management tool for RPM (RedHat Package Manager). It allows users and Linux system administrator to simply install, remove update, or search software packages on systems. It was developed under GPL (General Public License) i.e it is open source. YUM uses numerous third party repositories to install packages manually by resolving their dependencies issues.

Install a Package with YUM

Let us now want to install a package called Firefox 14, just run the command it will automatically find and install all required dependencies for Firefox

yum install Firefox 14

In the above command execution it will ask you confirmation before installing package on your system. If you want to install packages automatically without asking any confirmation, use option -y as follows

. # yum -y install Firefox 14

Removing a Package with YUM

in some case if you want to remove a package completely with their all dependencies, run the following command .

yum remove Firefox 14

Same as installation command it will ask you conformation to removing package if you what to remove packages without asking any kind of conformation execute following.

yum –y remove Firefox 14

Updating a Package using YUM

Let's say you have outdated version of MySQL package and you want to update it to the latest stable version. The how yum will help us update it ? , run the following command it will automatically resolves all dependencies issues and install them.

yum update mysql

List a Package using YUM

Use the list function to search for the specific package with name. For example to search for a package called openssh, use the command.

yum list openssh

Search for a Package using YUM

If you do not remember the exact name of the package you are looking for, then use search function with yum to search all the available packages which are match the name of the package you have give. For example we want to search all the packages that match the word. Then execute following command

yum search vsft

Get Information of a Package using YUM

Say you would like to know information of a package before installing it. to know information about any package use following command. Let us say we want to know information about Firefox.

yum info firebox

List all Available Packages using YUM

to list all the available packages in the Yum database, execute following command.

yum list | less

Check for Available Updates using Yum

To find how many of installed packages on your system have updates available, to check execute following command

yum check-update

Update System using Yum

To keep your system up-to-date with all security and binary package updates, run the following command. It will install all latest patches and security updates to your system.

yum update

List all available Group Packages

In Linux, numbers of packages are bundled to particular group. Instead of installing individual packages with yum, you can install particular group that will install all the related packages that belongs to the group. For example to list all the available groups, just issue following command.

yum grouplist

#yum groupinstall Install all packages in a package group

To learn more option which can use with yum, use man pages.

Querying Software

Once software installed, it can be quite helpful to query software. This is a generic way to get additional information about software installed on your system. In addition, querying RPM packages also helps you solve specific problems with packages,

There are many ways to query software packages. Before finding out more about your currently installed software, be aware that there are two ways to perform a query. You can query packages that are currently installed on your system, and it's also possible to install package files that haven't installed. To query an installed package, you can use one of the rpm -q options. To get information about a package that hasn't yet been installed, you need to add the -p option. To request a list of files that are in the samba-common RPM file, for example, you can use the rpm -ql samba-common command, if this package is installed. In case it hasn't yet been installed, you need to use rpm -qpl samba-common-[version-number].rpm, where you also need to refer to the exact location of the samba-common file. If you omit it, you'll get an error message stating that the samba-common package hasn't yet been installed

A very common way to query RPM packages is by using rpm -qa. This command generates a list of all RPM packages that are installed on your server and thus provides a useful means for finding out whether some software has been installed. Let us consider, if you want to check whether the media-player package is installed or not, you can use rpm -qa | grep media player. A useful modification to rpm -qa is the -V option, which shows you if a package have been modified from its original version. Using rpm -qVa thus allows you to perform a basic integrity check on the software you have on your server. Every file that is shown in the output of this command has been modified since it was originally installed. Note that this command will take a long time to complete. Also note that it is not the best way, nor the only one, to perform an integrity check on your server. Tripwire offers better and more advanced options. Listing 4.8 displays the output of rpm –qVa

Query options for installed packages

Query command	Result
rpm -ql packagename	Lists all files in packagename
rpm -qc packagename	Lists all configuration files in packagename
rpm -qd packagename	Lists all documentation files in packagename

To query packages that you haven't installed yet, you need to add the option –p. Finally, there is one more useful query option: rpm -qf. You can use this option to find out from which file a package originated.

Extracting Files from RPM Packages

It may happened that Software on your computer may get damaged . If this caes, we can extract files from the packages and copy them to the original place of the file on our system. The RPM package consists of two important parts the one contains the metadata which describes what this package contain and a cpio archive which contains the actual files in the package. Let us consider that our one file is damaged, we cto find out from what package the file originate using the rpm -qf query command. Now use rpm2cpio | cpio -idmv to extract that files from the package and store it at some temporary location. And then add this copy it at appropriate location.

In this chapter, you learned how to install, query, and manage software on your Linux server. You also learned how you can use the RPM tool to get extensive information about the software installed on your server.

Users and Groups

Authorization in Linux is provided by *users* and *groups*. Each user is associated with a unique positive integer called the *user ID* (uid). The unique id identifies the user running the process, and is called the process' *real uid*. Users refer to themselves and other users through *usernames*, and not with the numerical uid values. Usernames and their corresponding uids are stored in

/etc/passwd, and library routines map user-supplied usernames to the corresponding uids.

E.g. During login, the user provides a username and password to the login program. If given a valid username and the correct password, the login program spawns the user's login shell, which is also specified in /etc/passwd, and makes the shell's uid equal to that of the user. Child processes inherit the uids of their parents.

The uid 0 is associated with a special user known as *root*. The root user has special privileges, and can do almost anything on the system. For example, only the root user can change a process' uid. Consequently, the *login* program runs as root.

Each user may belong to one or more groups, including a primary or login group, listed in /etc/passwd, and possibly a number of supplemental groups, listed in /etc/group. Each process is therefore also associated with a corresponding group ID (gid), and has a real gid, an effective gid, a saved gid, and a file system gid. Processes are generally associated with a user's login group, not any of the supplemental groups.

Certain security checks allow processes to perform certain operations only if they meet specific criteria. Historically, UNIX has made this decision very black-and-white: processes with uid 0 had access, while no others did. Recently, Linux has replaced this security system with a more general capabilities system. Instead of a simple binary check, capabilities allow the kernel to base access on much more fine-grained settings.

Commands for User Management

If you want to add users from the command line, useradd is the command to use. Some other commands are available as well. Here are the most important commands for managing the user environment:

useradd - This command is used for adding users to the local authentication system.

usermod - This command is used to modify properties for existing users.

Userdel - This command is used to delete users properly from a system.

Using useradd is simple. In its easiest form, it just takes the name of a user as its argument, so user add parag will create a user called parag on your server. The useradd command has a few options. If an option is not specified, useradd will read its configuration file in

/etc/default/useradd. In this configuration file, useradd finds some default values. These specify the groups the user will become a member of, where to create the user's home directory, and more.

Ex: Setting default values in /etc/default/useradd [root@hnl ~]# cat /etc/default/useradd

useradd defaults
file GROUP=100

HOME=/
home
INACTIV
E=-1
EXPIRE=

SHELL=/bin/bash SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes

You can set different properties to manage users. To set up an effi cient server, it's important to know the purpose of the settings. For every user, the group membership, UID, and shell default properties are set.

Permissions

The standard file permission and security mechanism in Linux is the same as that of UNIX. Each file is associated with an owning user, an owning group, and a set of

permission bits. The bits describe the ability of the owning user, the owning group, and everybody else to read, write, and execute the file; there are three bits for each of the three classes, making nine bits in total. The owners and the permissions are stored in the file's inode.

For regular files, the permissions are rather obvious they specify the ability to open a file for reading, open a file for writing, or execute a file. Read and write permissions are the same for special files as for regular files, although what exactly is read or written is up to the special file in question. Execute permissions are ignored on special files. For directories, read permission allows the contents of the directory to be listed, write permission allows new links to be added inside the directory, and execute permission allows the directory to be entered and used in a pathname. The following table lists each of the nine permission bits, their octal values (a popular way of representing the nine bits), their text values (as ls might show them), and their corresponding meanings.

Bit	Octal value	Text value	Corresponding permission
8	400	r--------	Owner may read
7	200	-w-------	Owner may write
6	100	--x------	Owner may execute
5	040	---r-----	Group may read
4	020	----w----	Group may write
3	010	-----x---	Group may execute
2	004	------r--	Everyone else may read
1	002	-------w-	Everyone else may write
0	001	--------x	Everyone else may execute

In addition to historic UNIX permissions, Linux also supports access control lists (ACLs). ACLs allow for much more detailed and exacting permission and security controls, at the cost of increased complexity and on-disk storage.

Managing Passwords

To access the system, a user needs a password. By default, login is denied for the users you create, and passwords are not assigned automatically. Thus, your newly created users can't do anything on the server. To enable these users, assign passwords using the passwd command.

The passwd command is easy to use. A user can use it to change his password. If that happens, the passwd command will first prompt for the old password and then for the new one. Some complexity requirements, however, have to be met. This means, in essence, that the password cannot be a word that is also in the dictionary. The root user can change passwords as well. To set the password for a user, root can use passwd followed by the name of the user whose password needs to be changed. For example, passwd parag would change the password for user parag. The user root can use the passwd command in three generic ways. First, you can use it for password maintenance—to change a password, for example. Second, it can also be used to set password expiry information, which dictates that a password will expire at a particular date. Lastly, the passwd command can be used for account maintenance. For example, an administrator can use passwd to lock an account so that login is disabled temporarily.

Performing Account Maintenance with passwd

In an environment where many users are using the same server, it is important to perform some basic account maintenance tasks. These include locking accounts when they are unneeded for a long time, unlocking an account, and reporting the password status. An administrator can also force a user to change their password on first use.

To perform these tasks, the passwd command has some options available.

-l Enables an administrator to lock an account. For example, passwd -l rima will lock the account for user rima.

-u Unlocks an account that has been locked before.

-S Reports the status of the password for a given account.

-e Forces the user to change their password on next login.

Managing Password Expiry

In a server environment, it makes sense to change passwords occasionally. The passwd command has some options to manage account expiry.

-n min This rarely used option is applied to set the minimum number of days that a user must use their password. If this option is not used, a user can change their password at any time.

-x max This option is used to set the maximum number of days a user can use a password without changing it.

-c warn When a password is about to expire, you can use this option to send a warning to the user. The argument for this option specifies the number of days before expiry of the password that the user will receive the warning.

-i inact Use this option to expire an account automatically when it hasn't been used for a given period of time. The argument for this option is used to specify the exact duration of this period. Apart from the passwd command, you can also use change to manage account expiry. Consult the man page for more details on its usage.

Modifying and Deleting User Accounts

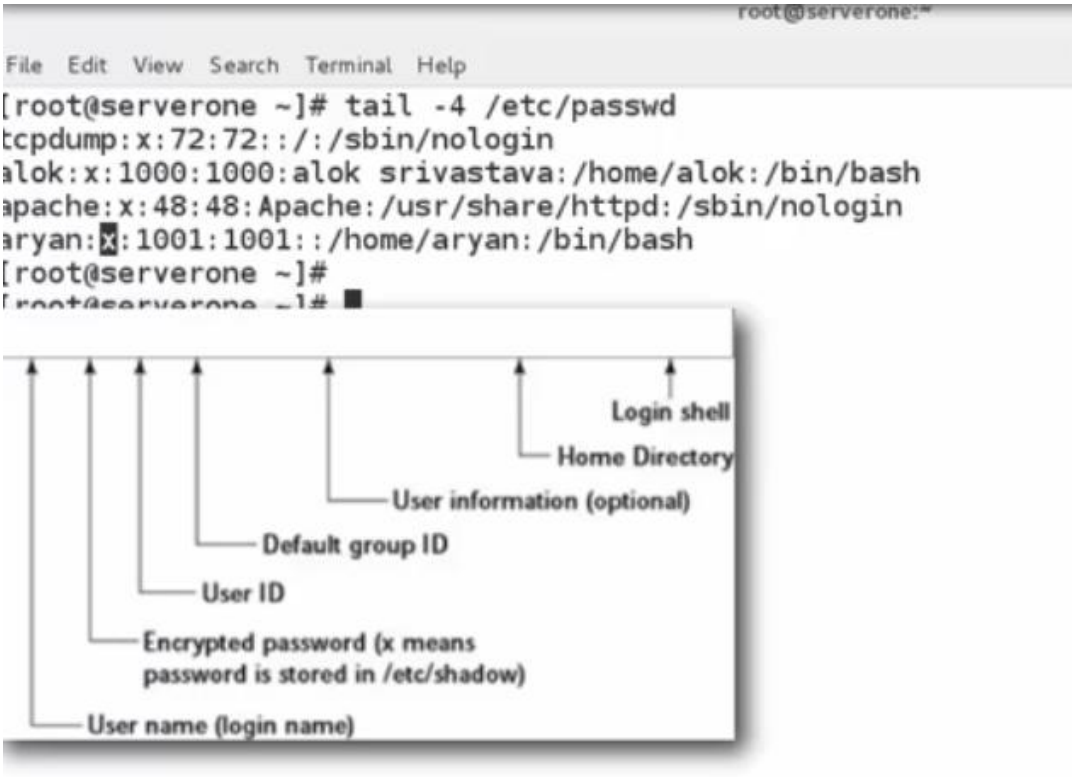
If you already know how to create a user, modifying an existing user account is no big deal. The usermod command is used for this purpose. It employs many of the same options that are used with useradd. For example, use usermod -g 101 linda to set the new primary group of user linda to a group with the unique ID 101. The usermod command has many other options. For a complete overview, consult its man page. Another command that you will occasionally need is userdel. Use this command to delete accounts from your server. userdel is a very simple command: userdel linda deletes user linda from your system, for example. However, if used this way, userdel will leave the home directory of your user untouched. This may be necessary to ensure that your company still has access to the work of a user

Configuration Files:

It is the Configuration files (or config files), which are used for user applications, server processes and operating system settings. For managing the user environment , a configuration file is also used which sets the default settings. In a operating system like Unix, many different configuration-file formats does exist. System- software often uses configuration files stored in the folder /etc, while user applications often use a "dot file" – a file or directory in the home directory prefixed with a period. Unix hides such files or directory from casual listing.

- **/etc/passwd**

It is the most important configuration file. /etc/passwd file is the primary database where user information is stored. That is, the most important user properties are stored in this file.



The following are the important fields from the configuration:

User name: The user's login name is stored in the first field in /etc/passwd. In modern Linux distributions, there is no limitation on the length of the login name. One can have user name of any length.

Password: The passwords are stored in the Encrypted format. And the passwords are always stored in the configuration file /etc/shadow.

User ID: Every user is given with a unique user ID. For the Red Hat Enterprise Linux, the starting local user IDs is 500, and the highest user ID to be used is 60000 (the highest numbers are reserved for special-purpose accounts).

Group ID: this field is used to reflect the Id of the primary group every user is member of. On Red Hat Enterprise Linux, every user is also a member of a private group that has the name of the user.

User Information: This field is used to include some additional information about the user. The field can contain any personal information, such as name of user's department, her phone number, or anything else related to the user. This makes identifying a user easier for an administrator. This is an optional field.

Home Directory: This field points to the directory of the user's home directory.

Login Shell: This is last field in /etc/passwd and used to refer to the program that starts automatically when a user logs in. Most often, this will be /bin/bash.

- **/etc/shadow**

The encrypted user passwords are stored in /etc/shadow. Other information relating to password like when the password will expire etc is also kept in this file.

```
[root@nn1 ~]# cat /etc/shadow
root:$6$4U.GRa4hziUW5lnk$gAbbcEBNFThzAc.GaQTuZUXYR/dJbhsoVWzzexkN
 AIYviyp5QlgUWTdf3tQot8jMYkUagI.rP3WtapObyFyIS1:15368:0:99999:7:::
bin:*:15155:0:99999:7:::
daemon:*:15155:0:99999:7:::
adm:*:15155:0:99999:7:::
lp:*:15155:0:99999:7:::
sync:*:15155:0:99999:7:::
shutdown:*:15155:0:99999:7:::
halt:*:15155:0:99999:7:::
mail:*:15155:0:99999:7:::
uucp:*:15155:0:99999:7:::
```

The folder /etc/shadow are also organized in different fields. The first two fields are the important fields. The first field is used to store the name of the user, and the second field is used to store the encrypted password. In the encrypted password field, an ! and an * can be used. If an! is used, login is currently disabled. If an * is used, it is a system account that can be used to start services, but that is not allowed for interactive shell login.

- **/etc/login.defs**

  The configuration file that relates to the user environment is /etc/login.defs. This file is used completely in the background. The generic settings are defined in this configuration file which is responsible for all kinds of information relating to the creation of users. The variables defined in login.defs will specify the default values used at time of users creation.

```
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local
#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is overridden with the -m flag on
# useradd command line.
#
CREATE_HOME     yes
# The permission mask is initialized to this value. If not specified,
# the permission mask will be initialized to 022.
UMASK           077
# This enables userdel to remove user groups if no members exist.
#
USERGROUPS_ENAB yes
# Use SHA512 to encrypt password.
ENCRYPT_METHOD SHA512
```

  login.defs contains variables that are used when users are created.

Creating Groups:

There are 2 types of groups - 1. Primary group
2. Secondary group (other group)

- **/ etc/ group**

  All groups on your system are defined in the configuration file names as /etc/group.

```
[root@hnl ~]# cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
vcsa:x:69:
rpc:x:32:
rtkit:x:497:
abrt:x:173:
cdrom:x:11:
tape:x:33:
sshd:x:74:
tcpdump:x:72:
slocate:x:21:
linda:x:500:
kvm:x:36:qemu
qemu:x:107:
cgred:x:490:
```

The first field in /etc/group is reserved for the name of the group whereas the second field indicates the password of the group. Every group is having a unique group ID which is provided in the third field of /etc/group. And finally, the last field contains the names of the members of the group. These names are required only for Non primary group members. Primary group membership itself is managed from the /etc/passwd configuration file.

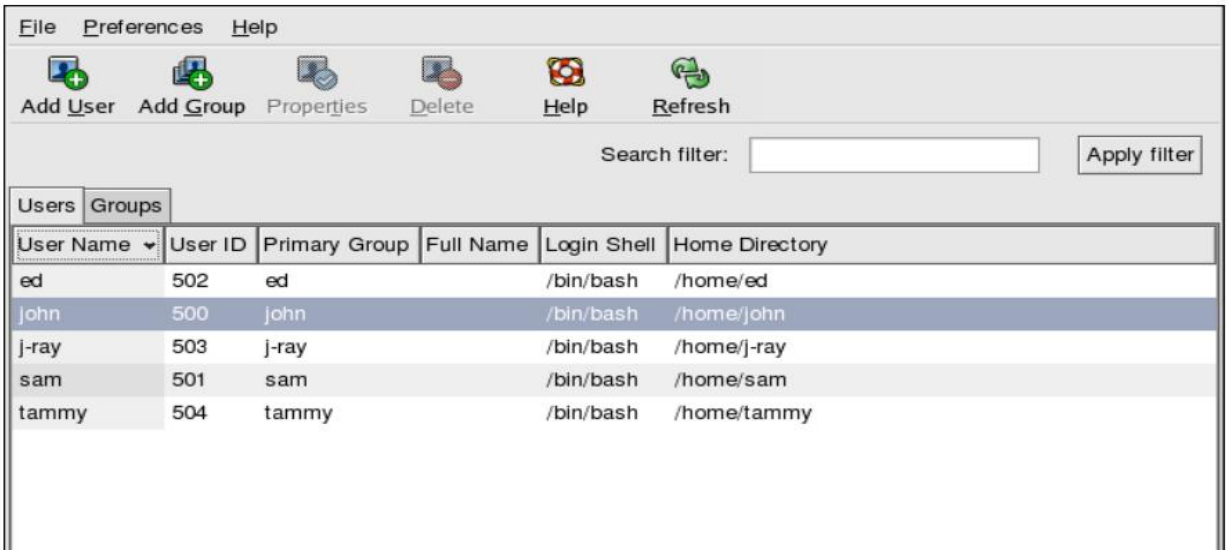
There are three commands to manage the groups in your environment:

- **Groupadd**-To add a user in the group.
- **Groupdel** -To delete a group.
- **Groupmod**- To modify a group.

1. **Graphical Tools for User and Group Management:**

   Linux provides the system-config tools that offer a graphical solution as an alternative to the command- line tools. For user and group management, the name of the tool is system-config-users.

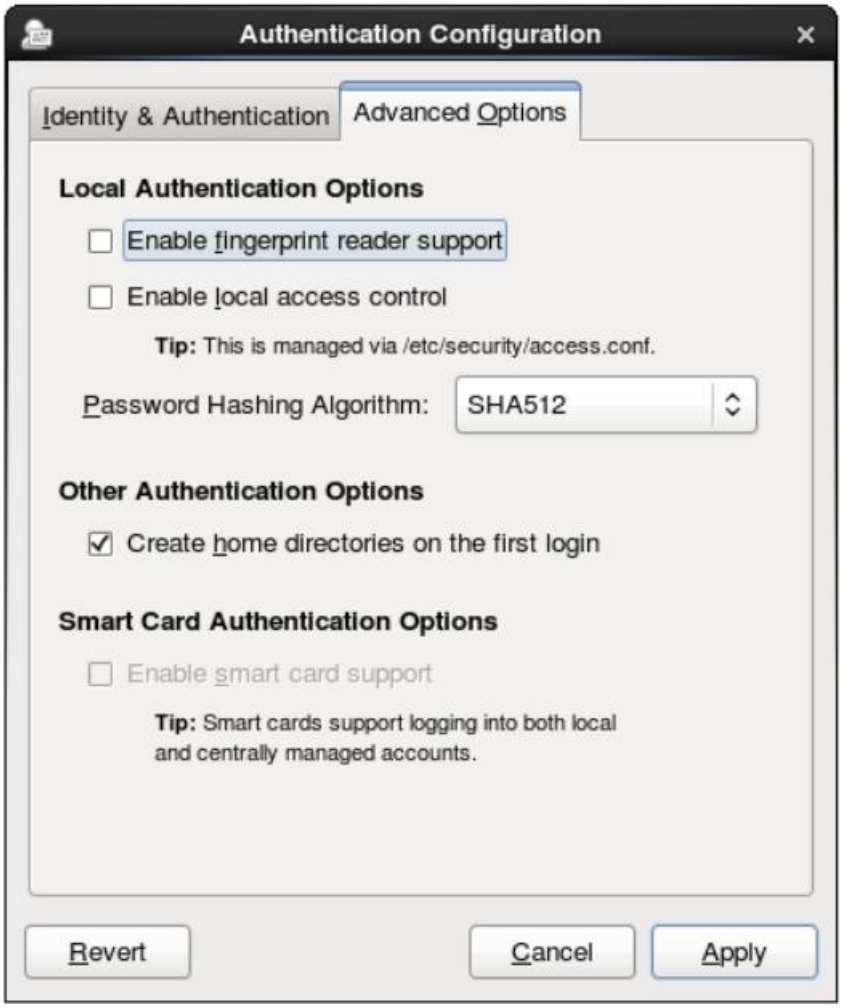
System -config-users provides a convenient interface for user and group management



The system-config-users tool was developed to simplify managing users and groups. To create a new user, click Add User. This opens the Add New User window in which you can specify all of the properties you want when creating a new user. It is also easy to add new groups. Just click Add Group, and you'll see a window prompting you for all of the properties that are needed to add a new group.

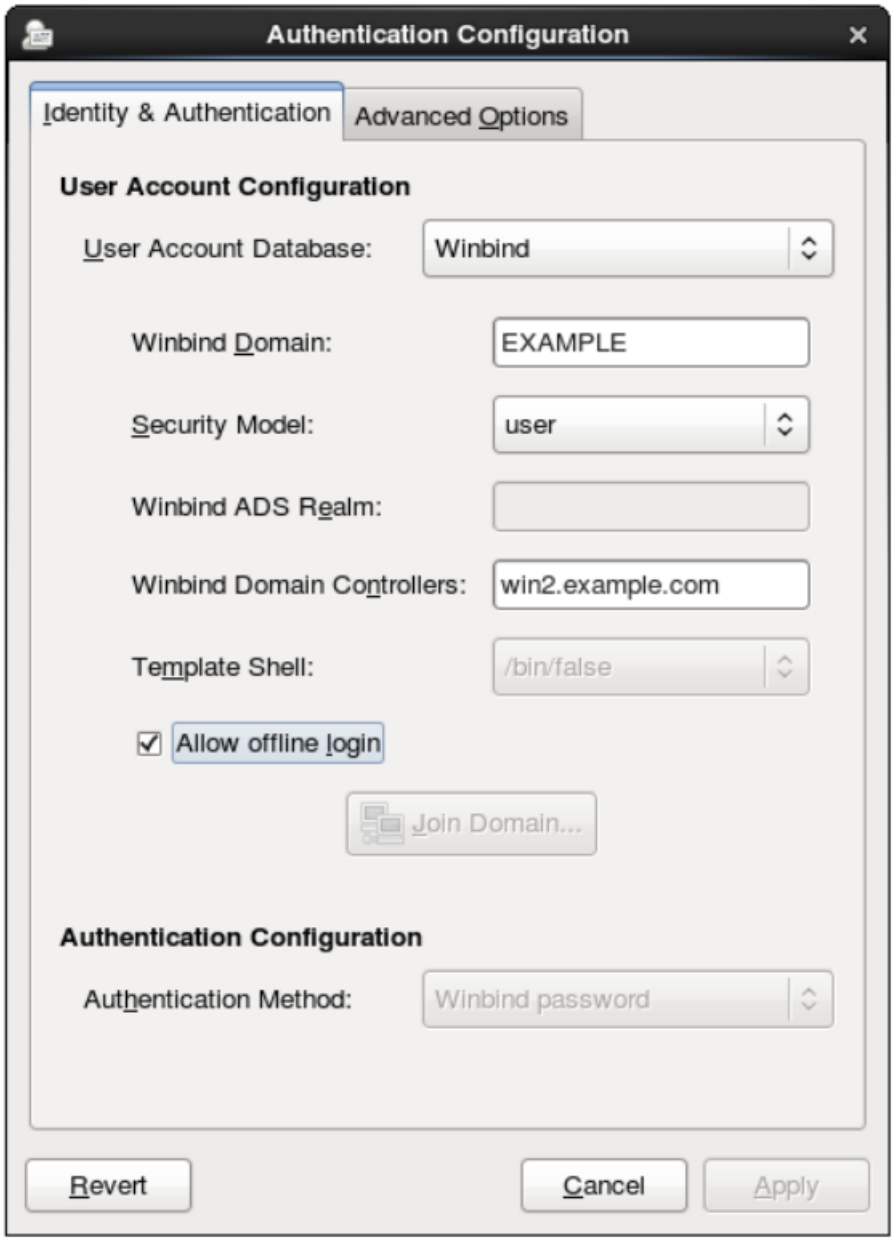
2. **External Authentication:**

An external source of authentication can be an LDAP directory server or an Active Directory service offered by Windows servers. To use these sources, you have to configure the server with the system-config-authentication tool or authconfig. After starting the system-config-authentication tool, you'll see two tabs. On the Identity & Authentication tab, you can specify how authentication should happen. By default, the tool is set to use local accounts only as the user account database. On the Advanced Options tab, you can enable advanced authentication methods, such as the use of a fingerprint reader.



Logging in Using an LDAP Directory Server:

Connecting to an Active Directory Server:



Authentication in process.

- Authentication is process used by a server in which the server grants the access to their information or site.
- When a user authenticates to your server, the local user database as defined in the files /etc/passwd and /etc/shadow is used on a default configuration.
- passwd is the file where the user information (like username, user ID, group ID, location of home directory, login shell, ...) is stored when a new user is created.
- Whenever a new user is created, a file called shadow is maintained, where important information about the password of the user is stored like an encrypted password of a user, password expiry date, whether or not the passwd has to be changed, the minimum and maximum time between password changes etc.
- To configure authentication against external authentication server, the sssd service is involved also PAM /etc/nsswitch.conf is used.
- For executing this command we must enables authentication server. Of typing commands like the following, which enables secure LDAPauthentication where Kerberos is used (all is one command!):

  authconfig --enableldap --enableldapauth --ldapserver=ldap.example.com

  --ldapbasedn=dc=example,dc=com --enabletls

  --ldaploadcert=http://ldap.example.com/certificate --enablekrb5

  --krb5kdc=krb.example.com --krb5realm=examplecom --update

1.SSSD:

- The System Security Services Daemon works in Ubuntu to allow authentication on directory-style backends, including OpenLDAP, Kerberos, RedHat's FreeIPA, Microsoft's Active Directory, and Samba4 Active Directory. It provides a cross-domain compatible method for users to sign in with configurable UID, GID, extended groups, home directory and login shell.It also provide information about authentication sources and also providing offline authentication.
- The configuration parameters you've specified are written to the configuration file /etc/sssd/sssd.conf.

- Command is" /etc/sssd/sssd.conf"
- LDAP authentication parameters in /etc/sssd/sssd.conf

ldap_user_object_class = user

ldap_group_object_class = group

ldap_user_home_directory = unixHomeDirectory

ldap_user_principal = userPrincipalName

ldap_account_expire_policy = ad

ldap_force_upper_case_realm = true

Understanding nsswitch **209**

#

krb5_server = your.ad.example.com

krb5_realm = EXAMPLE.COM

[domain/default]

ldap_id_use_start_tls = False

krb5_realm = EXAMPLE.COM

ldap_search_base = dc=example,dc=com

id_provider = ldap

auth_provider = krb5

chpass_provider = krb5

ldap_uri = ldap://127.0.0.1/

krb5_kpasswd = kerberos.example.com

krb5_kdcip = kerberos.example.com

cache_credentials = True

ldap_tls_cacertdir = /etc/openldap/cacerts

2. nsswitch

- It stands for Name Service Switch. The /etc/nsswitch file is used to determine where different services on a computer are looking for configuration information. The different sources of information are specified in this file.
- The nsswitch.conf file has different fields to maintain. The first field has service entry consisting of a database name, terminated by a colon, the second field has list of possible source databases mechanisms. A typical file might look like:
- Specifying sources of information in /etc/nsswitch.conf

    passwd: files sssd

    shadow: files sssd

    group: files sssd

    bootparams: nisplus [NOTFOUND=return] files

    ethers: files

    netmasks: files

    networks: files

    protocols: files

    rpc: files

    services: files

    netgroup: files

    publickey: nisplus

    automount: files

    aliases: files nisplus

3. Pluggable Authentication Modules

- PAM basically used for authentication is pluggable. Every modern service that needs to handle authentication passes through PAM. Every service has its own configuration file in the directory /etc/pam.d. For instance, the login service uses the configuration file /etc/pam.d/login.
- [root@hnl ~]# cat /etc/pam.d/login

  #%PAM-1.0

  auth [user_unknown=ignore success=ok ignore=ignore default=bad] pam_securetty.so

  auth include system-auth

  account required pam_nologin.so

  account include system-auth

  password include system-auth

  # pam_selinux.so close should be the first session rule

  session required pam_selinux.so close

  session required pam_loginuid.so

  session optional pam_console.so

  # pam_selinux.so open should only be followed by sessions to be

  executed in the user context

  session required pam_selinux.so open

  session required pam_namespace.so

  session optional pam_keyinit.so force revoke

  session include system-auth

  -session optional pam_ck_connector.so

1) Advanced Permissions:

- Sticky bit on directory: Files can be protected in directory from getting removed by other users who do not own it with the help of sticky bit. It is represented by‘t’ (Permission to execute file i.e. x is there for other user) or ‘T’ (x is not there for others). Sticky bit can also be set with the octal permissions it is binary 1 in first 4 triplets. In octal permission sticky bit is represented by ‘1’.

  Original permission of directory is:

```
drwxrwsr-x  2 kajal kajal 4096 Jul 23 09:29 dir1/
```

  After executing ‘chmod’ command to set sticky bit (symbolic representation):

```
kajal@kajal-desktop:~$ chmod +t dir1/
kajal@kajal-desktop:~$ ll
drwxrwsr-t  2 kajal kajal 4096 Jul 23 09:29 dir1/
```

  After executing ‘chmod’ command to set sticky bit (octal representation):

```
kajal@kajal-desktop:~$ chmod 1775 dir1/
kajal@kajal-desktop:~$ ll
drwxrwxr-t  2 kajal kajal 4096 Jul 23 09:29 dir1/
```

  To remove sticky bit just replace ‘+’ with ‘–‘

```
kajal@kajal-desktop:~$ chmod -t dir1/
```

- Setgid bit on regular directory : Setgid used to check whether all files in directory belongs to group of user. It’s location is same as location of ‘x’ for group user. It is represented by ‘s’ (Permission to execute file i.e. x is there for group user) or ‘S’ (x is not there for group user). In octal permission setgid is represented by ‘2’.

  After executing ‘chmod’ command to set setgid (symbolic representation):

```
kajal@kajal-desktop:~$ chmod g+s dir1/
kajal@kajal-desktop:~$ ll
drwxrwsr-x  2 kajal kajal 4096 Jul 23 09:29 dir1/
```

  Octal command for setgid is same as sticky bit only replacing 1 by 2.

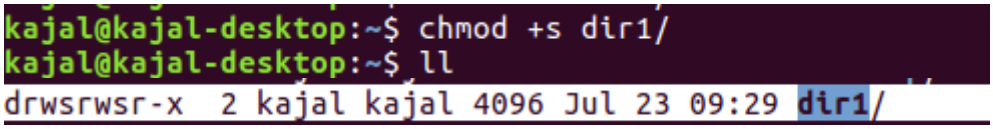
  To remove setgid just replace ‘+’ with ‘-‘.

- Setuid bit on regular directory : With the help of these permissions an executable file is accessed with permission of file owner instead of executing owner i.e. program runs as root if any use extends program and setuid bit is set for that program . It’s location is same as location of ‘x’ for owner user. It is represented by ‘s’ (Permission to execute file i.e. x is there for owner user) or ‘S’ (x is not

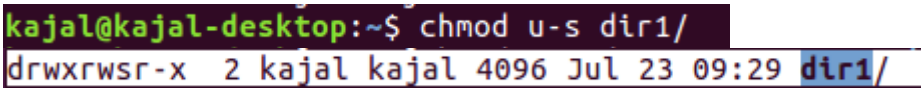
there for owner user). To set ‘setuid’ use ‘u+s’ in above command. In octal permission ‘setuid’ is represented by ‘4’.

```
chmod u+s myfile
```

Command to set setgid and setuid is:



To remove setuid just replace ‘+’ with ‘–‘



2) **Working with Access control list:**Access control list (ACL) provides an additional, more flexible permission mechanism for file systems. It is designed to assist with UNIX file permissions. ACL allows you to give permissions for any user or group to any disc resource.

Access ACL

The user and group access permissions for all kinds of file system objects (files and directories) are determined by means of access ACLs.

Default ACL

Default ACLs can only be applied to directories. They determine the permissions a file system object inherits from its parent directory when it is created.

ACL entry

Each ACL consists of a set of ACL entries. An ACL entry contains a type, a qualifier for the user or group to which the entry refers, and a set of permissions. For some entry types, the qualifier for the group or users is undefined.

Use of ACL :

Think of a scenario in which a particular user is not a member of group created by you but still you want to give some read or write access, how can you do it without making user a member of group, here comes in picture Access Control Lists, ACL helps us to do this trick.

From Linux man pages, ACLs are used to define more fine-grained discretionary access rights for files and directories.

```
kajal@kajal-desktop:~$ getfacl dir1
# file: dir1
# owner: kajal
# group: kajal
user::rwx
group::rwx
other::r-x

kajal@kajal-desktop:~$ setfacl -m d:o:6 dir1/
kajal@kajal-desktop:~$ getfacl dir1
# file: dir1
# owner: kajal
# group: kajal
user::rwx
group::rwx
other::r-x
default:user::rwx
default:group::rwx
default:other::rw-
```

3)Setting Default Permissions with umask :

When user creates a file or directory under Linux or UNIX, the file gets created but the set of permissions to that file is default set of permission. In most cases the system defaults may be open or relaxed for file sharing purpose. For example, if a text file has 666 permissions, it grants read and write permission to everyone. Similarly a directory with 777 permissions, grants read, writes, and executes permission to everyone.

When we create a new file or directory, shell automatically assigns the default permission to it.

```
Default permission = pre-defined initial permission – umask permission
```

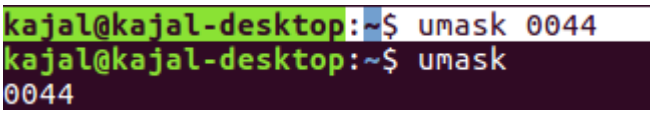
- The pre-defined initial permissions for files and directories are 666 and 777 respectively.
- The default umask permissions for root user and remaining users are 0022 and 0002 respectively.
- The pre-defined initial permissions are fixed and cannot be changed. But the default umask permissions are flexible and can be updated as per requirement.
- Umask permissions are also known as umask values or umask setting. All these words (umask permissions, umask values and umask setting) are used to represent the four numeric variables which are used to calculate the default permissions.

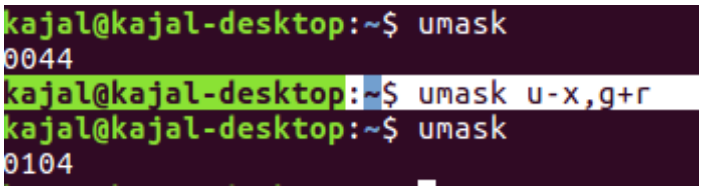
Umask can be set or expressed using:

- Symbolic values
- Octal values

umask Octal Value	**File Permissions**	**Directory Permissions**
0	rw-	Rwx
1	rw-	rw-
2	r--	r-x
3	r--	r--
4	-w-	-wx
5	-w-	-w-
6	--x	--x
7	--- (none)	--- (none)

To set umask following command is used:





4) Working with Attributes:

Attributes define properties of files. The file can have basic attribute such as File Index, Owner, File Size, File Time Stamps etc.

The files and directories can have following attributes:

- **a - append only:** this attribute allows a file to be added to, but not to be removed. It prevents accidental or malicious changes to files that record data, such as log files.
- **c - Compressed:** it causes the kernel to compress data written to the file automatically and uncompress it when it's read back.
- **i - Immutable:** it makes a file immutable. It not only restricts the write access to the file but also put few more restrictions like, the file can't be deleted, links to it can't be created, and the file can't be renamed.
- **j - Data journaling:** it ensures that on an Ext3 file system the file is first written to the journal and only after that to the data blocks on the hard disk.
- **s - Secure deletion:** it makes sure that recovery of a file is not possible after it has been deleted.
- **t - No tail-merging:** Tail-merging is a process in which small data pieces at a file's end that don't fill a complete block are merged with similar pieces of data from other files.
- **u - Undeletable:** When a file is deleted, its contents are saved which allows a utility to be developed that works with that information to salvage deleted files.
- **A - No atime updates:** Linux won't update the access time stamp when you access a file.

- **D - Synchronous directory updates:** it makes sure that changes to files are written to disk immediately and not to cache first.
- **S - Synchronous updates:** the changes on a file are written synchronously on the disk.
- **T - top of directory hierarchy:** A directory will be deemed to be the top of directory hierarchies for the purposes of the Orlov block allocator.

- The‘chattr’ is the command that allows a user to set certain attributes of a file.
- The ‘lsattr’ is the command that displays the attributes of a file.
- Among other things, the chattr command is useful to make files immutable so that password files and certain system files cannot be erased during software upgrades.

```
chattr +i test.txt
```

```
kajal@kajal-desktop:~$ lsattr test1
----------e-- test1
```

Using ‘–d’ it gives list of attributes itself instead of files in that directory.
Using ‘–R’ it gives list of attributes recursively, shows the subdirectories as well.

```
kajal@kajal-desktop:~$ lsattr -d dir1
----------e-- dir1
kajal@kajal-desktop:~$ lsattr -R dir1
----------e-- dir1/newfolder

dir1/newfolder:

----------e-- dir1/newfile
```

Unit 5: TCP/IP Networking and Network File System

5.1.1

5.1 Learning Objectives

After successful completion of this unit, you will be able to:

- Analyze the requirements for a given organizational structure to select the most appropriate class address.
- Configure DHCP and NFS.

5.2 Introduction

TCP/IP stands 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.

5.3 TCP/IP Networking:

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.

5.4 Understanding Network Classes

An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing. Internet Protocol version 4 (IPv4) defines an IP address as a 32-bit number. However, because of the growth of the Internet and the depletion of available IPv4 addresses, a new version of IP (IPv6), using 128 bits for the IP address, was developed. IP addresses are usually written and displayed in human-readable notations, such as 172.16.254.1 in IPv4, and 2001:db8:0:1234:0:567:8:1 in IPv6.
Every host and router on the Internet has an IP address, which encodes its network number and host number. The combination is unique: in principle, no two machines on the Internet have the same IP address.
The class A, B, C, and D formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 64K hosts, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special). Also supported is multicast, in which a datagram is directed to multiple hosts. Addresses beginning with 1111 are reserved for future use. Over 500,000 networks are now connected to the Internet, and the number grows every year. Network numbers are managed by a nonprofit corporation called ICANN (Internet Corporation for Assigned Names and Numbers) to avoid conflicts.

Class	**First Byte**
A	1 – 127
B	128 – 191

C	192 – 223
D	224 – 239

Table 1: Network Class range

Class A Address:
The first bit of the first octet is always set to 0 (zero). Thus the first octet ranges from 1 – 127, i.e.

00000001 – 01111111
1 – 127

Class A addresses only include IP starting from 1.x.x.x to 126.x.x.x only. The IP range 127.x.x.x is reserved for loopback IP addresses.
The default subnet mask for Class A IP address is 255.0.0.0.

Class B Address
An IP address which belongs to class B has the first two bits in the first octet set to 10, i.e.

10000000 – 10111111
128 – 191

Class B IP Addresses range from 128.0.x.x to 191.255.x.x. The default subnet mask for Class B is 255.255.x.x.

Class C Address
The first octet of Class C IP address has its first 3 bits set to 110, that is:

11000000 – 11011111
192 – 223

Class C IP addresses range from 192.0.0.x to 223.255.255.x. The default subnet mask for Class C is 255.255.255.x.

Class D Address
Very first four bits of the first octet in Class D IP addresses are set to 1110, giving a range of:

11100000 – 11101111
224 – 239

Class D has IP address range from 224.0.0.0 to 239.255.255.255. Class D is reserved for Multicasting. In multicasting data is not destined for a particular host, that is why there is no need to extract host address from the IP address, and Class D does not have any subnet mask.

Class E Address
This IP Class is reserved for experimental purposes only for R&D or Study. IP addresses in this class ranges from 240.0.0.0 to 255.255.255.254. Like Class D, this class too is not equipped with any subnet mask.

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 number of hosts to be connected. Although this is possible, in the real world this approach would not allow for connecting to the Internet.

5.5 Setting Up a Network Interface Card (NIC)

Red Hat Linux distribution includes networking support and tools that can be used to configure your network. Even if the computer is not connected to outside networks, an internal network functionality is required for some applications. This address is known as the loopback 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.

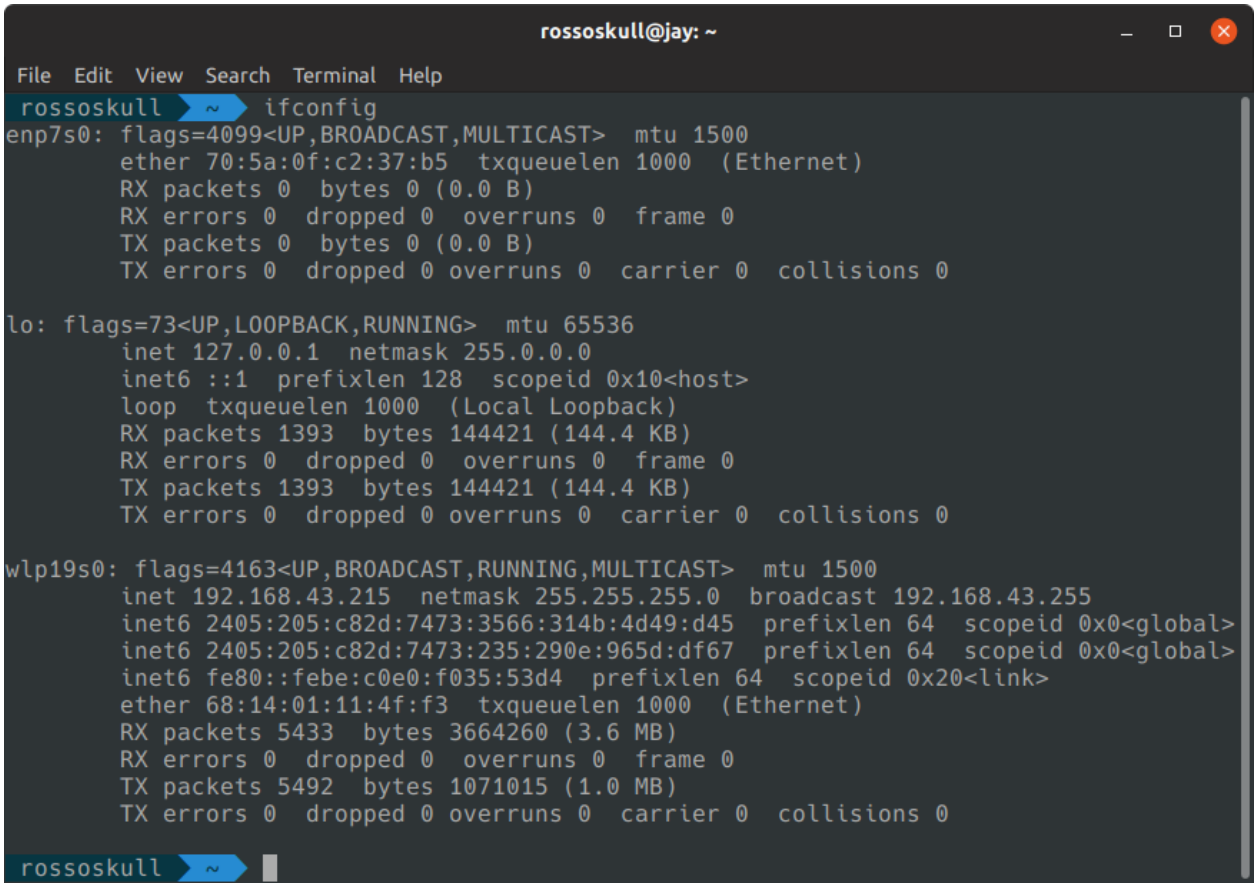


Figure 1: ifconfig output

To configure a network card 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 netmask, and the broadcast addresses. These numbers vary depending on the type of network being built.

In this example, you configure an Ethernet interface for an internal network. You need to issue the command:

ifconfig eth0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255

Another way to configure network is using GUI. In setting you will find network configuration. Figure 2 shows network configuration.

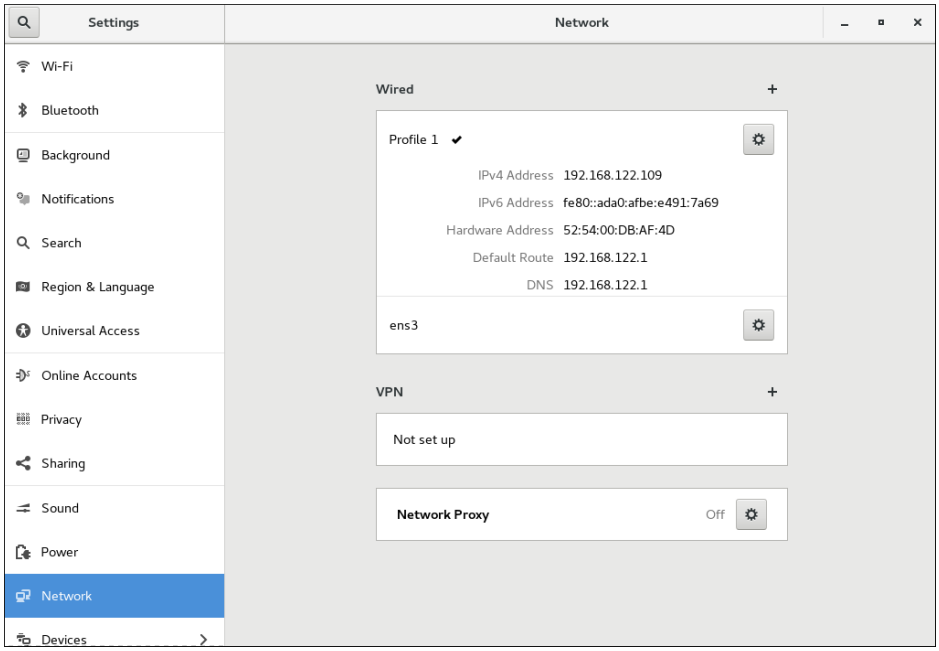


Figure 2: Network configuration

5.6 Understanding Subnetting

In previous section we have seen how to build an internal network, but to connect to the outside world few more steps are needed including configuring a router, obtaining an IP address, and actually making the connection. IP numbers are not assigned to hosts, they are assigned to network interfaces onhosts. Even though many computers on an IP network have a single network interfaceand a single IP number, a single computer can have more than one networkinterface. In this case, each interface would have its own IP number.Even though this is true, most people refer to host addresses when referring to anIP number. Just remember, this is simply shorthand for the IP number of this particularinterface on this host. Many devices on the Internet have only a single interfaceand thus a single IP number.In the current (IPv4) implementation, IP numbers consist of 4 (8-bit) bytes for atotal of 32 bits of available information. This system results in large numbers, evenwhen they are represented in decimal notation. To make them easier to read andorganize, they are written in what is called dotted quad format. The numbers yousaw earlier in this chapter were expressed in this format, such as the internal networkIP address 192.168.1.1. Each of the four groups of numbers can range from 0to 255. The following shows the IP number in binary notation with its decimalequivalent. If the bit is set to 1 it is counted, and if set to zero it is not counted.

$$1 + 1 + 1 + 1 + 1 + 1 + 1 + 1$$

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$$

The binary notation for 192.168.1.1 would be:

$$11000000.10101000.00000001.00000001$$

The dotted quad notation from this binary is:

(128+64) = 192.(128+32+8) = 168.(1)=1.(1) = 1

The leftmost bits of the IP number of a host identify the network on which the host resides; the remaining bits of the IP number identify the network interface. Exactly how many bits are used by the network ID and how many are available to identify interfaces on that network is determined by the network class.

Class A IP network numbers use the left quad to identify the network, leaving3 quads to identify host interfaces on that network. Class A addresses always have the farthest left bit of the farthest left byte a zero, so there are a maximum of 128class A network numbers available, with each one containing up to 33,554,430possible interfaces.

The network numbers 0.0.0.0, known as the default route, and 127.0.0.0, the loopback network, have special meanings and cannot be used to identify networks. You saw the loopback interface when you set up your internal network. You'll look at the default route when you set up your connection to the Internet. So if you take these two network numbers out, there are only 126 available class A network numbers.

Class B IP network numbers use the two left dotted quads to identify the network, leaving two dotted quads to identify host interfaces. Class B addresses always have the farthest left bits of the left byte set to 10. This leaves 14 bits left to specify the network address giving 32,767 available B class networks. Class B networks have a range of 128 to 191 for the first of the dotted quads, with each network containing up to 32,766 possible interfaces. Class C IP network numbers use the left three quads to identify the network, leaving the right quad to identify host interfaces.

Class C addresses always start with the farthest left 3 bits set to 1 1 0 or a range of 192 to 255 for the farthest left dotted quad. This means that there are 4,194,303 available Class C network numbers, each containing 254 interfaces.

As we have seen, all the hosts in a network must have the same network number. This property of IP addressing can cause problems as networks grow. For example, consider an organization that started with class B network for accounts Dept. After one year the marketing Dept. wanted to get on the Internet, so they used a repeater to extend the accounts Ethernet to their building. As time went on, many other departments acquired computers and the limit of four repeaters per Ethernet was quickly reached. Due to this a different organization was required. Solution for this is to divide network in several parts which will act as a single network to outside world.

A subnetwork or subnet is a logical subdivision of an IP network. The practice of dividing a network into two or more networks is called subnetting. Computers that belong to a subnet are addressed with an identical most-significant bit-group in their IP addresses. This results in the logical division of an IP address into two fields, the network number or routing prefix and the rest field or host identifier. The rest field is an identifier for a specific host or network interface.

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 which need multiple connectivity, but eliminate 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 simply based on the number of times a given number can be divided.

Subnetworking takes one or more of the available host bits and makes them appear as network bits to the local interfaces. If you wanted to divide your Class Cnetwork into two subnetworks, you would change the first host bit to one, and you would get a netmask of 11111111.11111111.11111111.10000000 or 255.255.255.128.This would give you 126 possible IP numbers for each of our subnets. Remember that you lose two IP addresses for each subnet. If you want to have four subnetworks, you need to change the first two host bits to ones, and this would give you a netmask of 255.255.255.192. You would have 62 IP addresses available on each subnetwork. Table 5.6.1 shows the subnets, the subnet masks, and the available hosts for your Class C network.

Number of Bits	Number of Subnets	Subnet Mask	Number of Hosts
1	2	255.255.255.128	126
2	4	255.255.255.192	62
3	8	255.255.255.224	30
4	16	255.255.255.240	14
5	32	255.255.255.248	6
6	64	255.255.255.252	2

Table 5.6.1: CLASS C SUBNETS AND SUBNET MASKS

Now all you need to do is assign the appropriate numbers for the network, the broadcast address, and the IP addresses for each of the interfaces and you're nearly done. Table 5.6.2 shows these numbers for subnetting your Class C network into two subnets.

Network	Netmask	Broadcast	First IP	Last IP
192.168.1.0	255.255.255.128	192.168.1.127	192.168.1.1	192.168.1.126
192.168.1.128	255.255.255.128	192.168.1.255	192.168.1.129	192.168.1.254

Table 5.6.2: CREATING TWO SUBNETS FOR A CLASS C NETWORK ADDRESS

Classless Inter Domain Routing (CIDR): 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. For example, if you needed about a thousand addresses, you could supernet 4 Class C networks together:

192.60.128.0 (11000000.00111100.10000000.00000000) Class C subnet address

192.60.129.0 (11000000.00111100.10000001.00000000) Class C subnet address

192.60.130.0 (11000000.00111100.10000010.00000000) Class C subnet address

192.60.131.0 (11000000.00111100.10000011.00000000) Class C subnet address

---

192.60.128.0 (11000000.00111100.10000000.00000000) Supernetted Subnet address

255.255.252.0 (11111111.11111111.11111100.00000000) Subnet Mask

192.60.131.255 (11000000.00111100.10000011.11111111) Broadcast address

In this example, the subnet 192.60.128.0 includes all the addresses from192.60.128.0 to 192.60.131.255. As you can see in the binary representation of the subnet mask, the network portion of the address is 22 bits long, and the host portion is 10 bits long. Under CIDR, the subnet mask notation is reduced to simplified shorthand. Instead of spelling out the bits of the subnet mask, the number of 1s 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 starting address of the network, and number of 1s 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).

It is currently almost impossible for you, as an individual or company, to be allocated your own IP address blocks. You will be told simply to get them from your ISP. The reason for this is the ever-growing size of the Internet routing table. Just five years ago, there were less than 5,000 network routes in the entire Internet. Today, there are over 100,000. Using CIDR, the biggest ISPs are allocated large chunks of address space, usually with a subnet mask of /19 or even smaller. The ISP's customers, often other, smaller ISPs, are then allocated networks from the big ISP's pool. That way, all the big ISP's customers, and their customers, are accessible via one network route on the Internet.

CIDR will probably keep the Internet happily in IP addresses for the next fewyears at least. After that, IPv6, with 128 bit addresses, will be needed. Under IPv6,even careless address allocation would comfortably enable a billion unique IP addresses for every person on earth! The complete details of CIDR are documented in RFC1519, which was released in September of 1993.

5.7 Working with Gateways and Routers

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 that is connected to each network is called its gateway. In Linux, computer with two network interfaces, can use to route between two or more subnets. For this we need to 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. Use below command to check:

cat /proc/sys/net.ipv4/ip_forward.

If forwarding is not enabled then it returns number 0, and if enabled then number 0. Type the following command to enable IP forwarding if it is not already enabled:

echo "0" > /proc/sys/net/ipv4/ip_forward

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
route add default gw 192.168.1.129

5.8 Configuring Dynamic Host Configuration Protocol

Every device on a TCP/IP-based network must have a unique unicast IP address to access the network and its resources. Without DHCP, IP addresses for new computers or computers that are moved from one subnet to another must be configured manually; IP addresses for computers that are removed from the network must be manually reclaimed.

With DHCP, this entire process is automated and managed centrally. The DHCP server maintains a pool of IP addresses and leases an address to any DHCP-enabled client when it starts up on the network. Because the IP addresses are dynamic rather than static addresses no longer in use are automatically returned to the pool for reallocation. 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. You should have a DHCP server configured for each of your subnets. Each host on the subnet needs to be configured as a DHCP client. You may also need to configure the server that connects to your ISP as a DHCP client if your ISP dynamically assigns your IP address.

The network administrator establishes DHCP servers that maintain TCP/IP configuration information and provide address configuration to DHCP-enabled clients in the form of a lease offer. The DHCP server stores the configuration information in a database that includes:

- Valid IP addresses, maintained in a pool for assignment to clients, as well as excluded addresses.
- Reserved IP addresses associated with particular DHCP clients. This allows consistent assignment of a single IP address to a single DHCP client.

- The lease duration, or the length of time for which the IP address can be used before a lease renewal is required.

The program which runs on the server is dhcpd and is included as an RPM on Red Hat 7.2. Look for the file dhcp-2.0pl5-1.i386.rpm and use the Gnome-RPM (the graphical RPM tool) from the desktop, or use the rpm command from a command prompt to install it. In Red Hat Linux the DHCP server is controlled by the text file /etc/ dhcpd.conf. Here is a sample of a typical setup file. Shown in parentheses is an explanation of the line.

default-lease-time 36000; (The amount of time in seconds that the host can keep the IP address.)

max-lease-time 100000; (The maximum time the host can keep the IP address.)

#domain name

option domain-name "tactechnology.com"; (The domain of the DHCPserver.)

#nameserver

option domain-name-servers 192.168.1.1; (The IP address of the DNSservers.)

#gateway/routers, can pass more than one:

option routers 1.2.3.4,1.2.3.5;

option routers 192.168.1.1; (IP address of routers.)

#netmask

option subnet-mask 255.255.255.0; (The subnet mask of the network.)

#broadcast address

option broadcast-address 192.168.1.255; (The broadcast address ofthe network.)

#specify the subnet number gets assigned in

subnet 192.168.1.0 netmask 255.255.255.0 (The subnet that uses thedhcp server.)

#define which addresses can be used/assigned

range 192.168.1.1 192.168.1.126; (The range of IP addresses that canbe used.)

To start the server, run the command dhcpd. To ensure that the dhcpd program runs whenever the system is booted, you should put the command in one of your init scripts.

First you need to check if the dhcp client is installed on your system. You can check for it by issuing the following command:

whichdhcpcd

If the client is on your system, you will see the location of the file. If the file is not installed, find it on Red Hat Installation CD 1. Use the rpm command to install the client. After you install the client software, start it by running the command dhcpcd. Each of your clients will now receive its IP address, subnet mask, gateway, and broadcast address from your dhcp server. Since you want this program to run every time the computer boots, you need to place it in the /etc/rc.local file. Now whenever the system starts, this daemon will be loaded.

5.9 Configuring the Network Using the Network

Now that you know how to work with services in Red Hat Enterprise Linux, it's time to get familiar with Network Manager. The easiest way to configure the network is by clicking the Network Manager icon on the graphical desktop of your server. In this section, you'll learn how to set network parameters using the graphical tool. You can find the Network Manager icon in the upper-right corner of the graphical desktop. If you click it, it provides an overview of all currently available network connections, including Wi-Fi networks to which your server is not connected. This interface is convenient if you're using Linux on a laptop that roams from one Wi-Fi network to another, but it's not as useful for servers. If you right-click the Network Manager icon, you can select Edit Connections to set the properties for your server's network connections. You'll find all of the wired network connections on the Wired tab. The name of the connection you're using depends on the physical location of the device. Whereas in older versions of RHEL names like eth0 andeth1 were used, Red Hat Enterprise Linux 6.2 and newer uses device-dependent names likep6p1. On servers with many network cards, it can be hard to find the specific device you need. However, if your server has only one network card installed, it is not that hard. Just select the network card that is listed on the Wired tab (as shown in below figure).

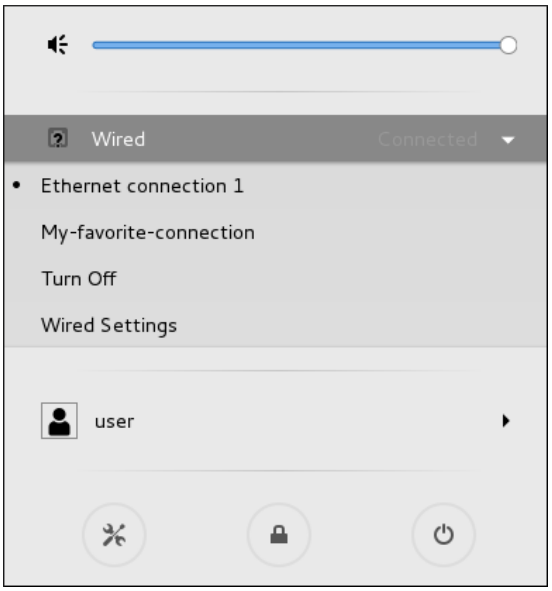


Figure 3: Configure Networks Using the Network Settings Window

When you click on the GNOME Shell network connection icon, you are presented with:

- a list of categorized networks you are currently connected to (such as Wired and Wi-Fi);
- a list of all Available Networks that Network Manager has detected;
- options for connecting to any configured Virtual Private Networks (VPNs); and,
- an option for selecting the Network Settings menu entry.

If you are connected to a network, this is indicated by a black bullet on the left of the connection name.
Click Network Settings. The Network settings tool appears.

5.10 The Network File System:

A Network File System (NFS) allows remote hosts to mount file systems over a network and interact with those file systems as though they are mounted locally. This enables system administrators to consolidate resources onto centralized servers on the network.

5.11 NFS Overview

NFS, the Network File System, is the most common method for providing file sharing services on Linux and Unix networks. It is a distributed file system that enables local access to remote disks and file systems. In a properly designed and implemented NFS environment, NFS's operation is totally transparent to clients using remote file systems. NFS is also a popular file sharing protocol, so NFS clients are available for many non-Unix operating systems, including the various Windows versions, MacOS, VAX/VMS, and MVS.

NFS Server Configuration:

There are three ways to configure an NFS server under Red Hat Enterprise Linux: using the NFS Server Configuration Tool (system-config-nfs), manually editing its configuration file (/etc/exports), or using the /usr/sbin/exportfs command.

To use the NFS Server Configuration Tool, you must be running X Windows, have root privileges, and have the system-config-nfs RPM package installed. To start the application, click on System => Administration => Server Settings => NFS. You can also type the command system-config-nfs in a terminal. The NFS Server Configuration tool window is illustrated below.

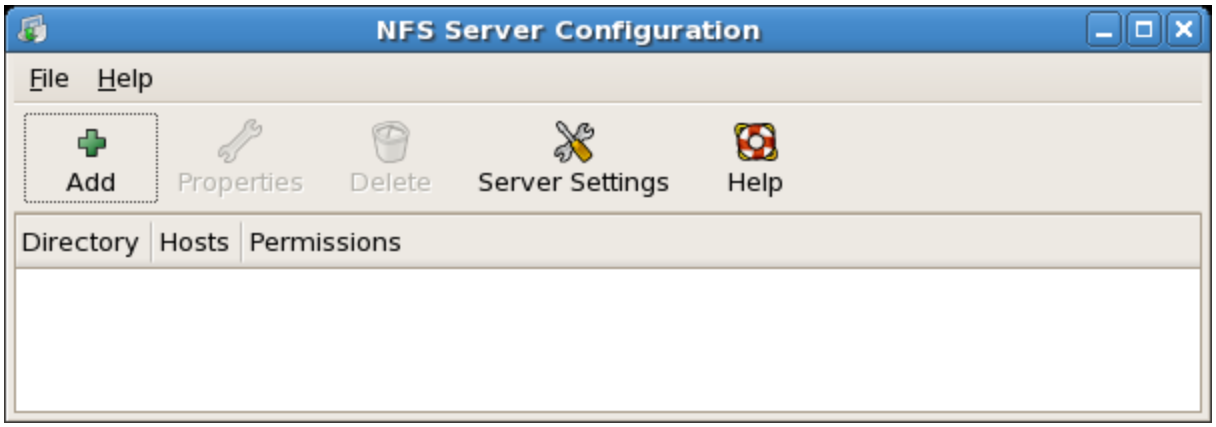


Figure: NFS Server Configuration Tool

Based on certain firewall settings, you may need to configure the NFS daemon processes to use specific networking ports. The NFS server settings allows you to specify the ports for each process instead of using the random ports assigned by the portmapper. You can set the NFS Server settings by clicking on the Server Settings button. The figure below illustrates the NFS Server Settings window.

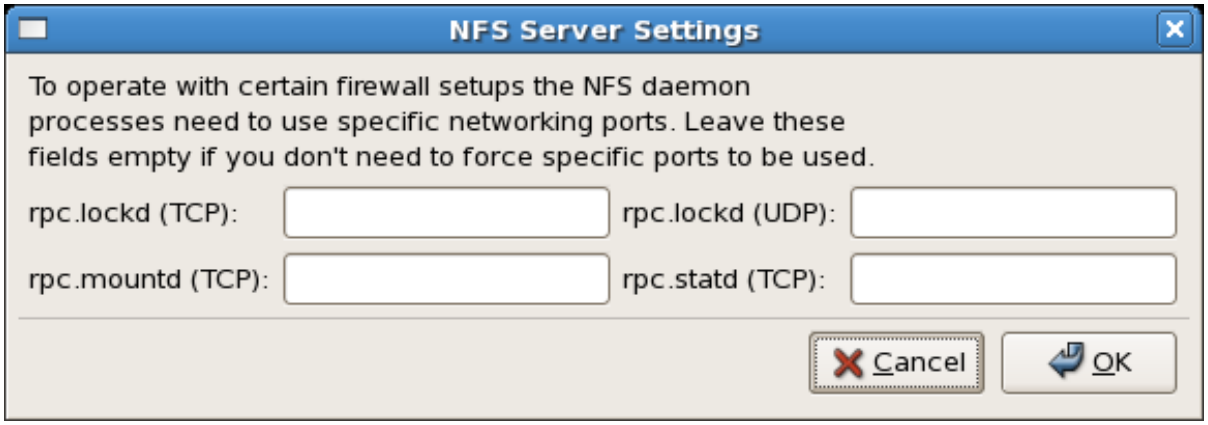


Figure: NFS Server Settings

Exporting or Sharing NFS File Systems:

Sharing or serving files from an NFS server is known as exporting the directories. The NFS Server Configuration Tool can be used to configure a system as an NFS server.

To add an NFS share, click the Add button. The dialog box shown in Figure 18.3, "Add Share" appears.

The Basic tab requires the following information:

- Directory — Specify the directory to share, such as /tmp.
- Host(s) — Specify the host(s) with which to share the directory. Refer to Section 18.6.3, "Hostname Formats" for an explanation of possible formats.
- Basic permissions — Specify whether the directory should have read-only or read/write permissions.

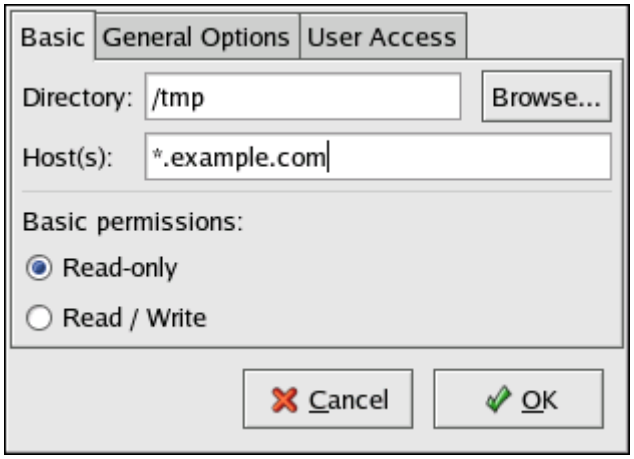


Figure: Add Share

5.12 Planning an NFS Installation,

The possible uses of NFS are quite varied. For example, 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. Of course, in this case, the exported directories must be mounted as /home/username on the local (client) systems, but the export itself can be stored anywhere on the NFS server, say, /exports/users/ username. Another common scheme is to export public data or project-specific files from an NFS server and to enable clients to mount these remote file systems anywhere they see fit on the local system.

NFS advantages: Clearly, the biggest advantage NFS provides is centralized administration. It is much easier, for example, to back up a file system stored on a server (such as the /home file system) than it is to back up /home directories scattered throughout the network, on systems that are geographically dispersed, and that might or might not be accessible when the backup is made.
NFS disadvantages: 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 both cases, NFS clients seem to be running slowly because disk reads and writes take longer. If an exported file system is not available when a client attempts to mount it, the client system hangs, although this can be mitigated using a specific mount. An exported file system also represents a single point of failure. If the disk or system exporting vitaldata or application becomes unavailable for any reason (say, due to a disk crash or server failure), no one can access that resource.

5.13 Configuring an NFS Server

Configuring an NFS server divided into design and implementation. Of these two steps, design is the most important because it ensures that the implementation is transparent to end users and trivial to administer. The implementation is remarkably straightforward. This section highlights the server configuration process, discusses the key design issues to keep in mind, identifies the key files and commands you use to implement, maintain, and monitor the NFS server, and illustrates the process using a typical NFS configuration. Server configuration stapes are as follows.

- Design
- Implementation
- Testing
- Monitoring

In Server designing we need to decide what file systems to export to which users and selecting a naming convention and mounting scheme that maintains network transparency. Implementation is nothing but how to configure the exports and starting the appropriate daemons. After implementation is important to test the naming convention and mounting scheme to check whether it works as designed and identifies potential performance bottlenecks. Monitoring, finally, extends the testing process: you need to ensure that exported file systems continue to be available and that heavy usage or a poorly conceived export scheme does not adversely affect overall performance.
Designing an NFS server involves

- Selecting the file systems to export
- Choosing which users (or hosts) are permitted to mount the exported file Systems
- Selecting a naming convention and mounting scheme that maintains network transparency and ease of use
- Configuring the server and client systems to follow the convention

Tips and suggestions for designing an NFS server are as follows:

- file system that is shared among a large number of users, such as /home, workgroup project directories, shared directories etc.,
- Use /home/username to mount home directories

- Use the same path names on the server and on clients
- Few networks are static, particularly network file systems, so design NFS servers with growth in mind.

Configuration and status files are:

- /etc/exports
- /var/lib/nfs/rmtab
- /var/lib/nfs/xtab
- /etc/hosts.allow
- /etc/hosts.deny

Scripts and commands

- /etc/rc.d/init.d/nfs
- Nfstat
- Showmount
- Rpcinfo
- Exportfs

5.14 Configuring an NFS Client

Configuring client systems to mount NFS exports is simpler than configuring the NFS server itself. Here we will study key files and commands involved in configuring a client to access the NFS exports configured.

For NFS client, NFS exported file systems are functionally equivalent to local file systems. Thus, as you might expect, you use the mount command at the command line to mount NFS exports on the fly, just as you would a local file system.

NFS shares are mounted on the client side using the *mount* command. The format of the command is as follows:

```
mount -t <nfs-type> -o
<options><host>:</remote/export></local/directory>
```

An alternate way to mount an NFS share from another machine is to add a line to the /etc/fstab file. At boot time the /etc/fstab file is referenced by the netfs service, so lines referencing NFS shares have the same effect as manually typing the mount command during the boot process. Each line in this file must state the hostname of the NFS server, the directory on the server being exported, and the directory on the local machine where the NFS share is to be mounted. To modify the /etc/fstab file you must be root.

The most commonly used and useful NFS-specific mount options arersize=8192, wsize=8192, hard, intr, and no lock. Increasing the default size of the NFS read and write buffers improves NFS's performance. The suggested value is8192 bytes, but you might find that you get better performance with larger or smaller values. The no lock option can also improve performance because it eliminates the overhead of file locking calls, but not all servers support file locking over NFS.

NFS client requires the portmap daemon to process and route RPC calls and returns from the server to the appropriate port and programs. It is important that the portmapper is running on the client system using the portmap initialization script, /etc/rc.d/init.d/portmap. If you want to use NFS file locking, an NFS server and any NFS clients need to run statd and lockd. For this use the initialization script, /etc/rc.d/init.d/nfslock.After configuring the mount table and starting the

requisite daemons, last step is to mount the file systems. To mount /home from the server configured at the end of the previous section, execute the following command as root:

mount -t nfsluther:/home /home

When we talk about NFS performance, we should check network performance. Because heavy network traffic degrades NFS performance. To diagnose specific NFS performance problems, use the nfsstat command, which prints the kernel's NFS statistics. Its syntax is:

nfsstat [-acnrsz] [-o facility]

Apart from performance degradation, you might encounter other problems with NFS that require resolution like, attempt to access a file to which NFS client does not have access, timing out etc.
Now we will see the issues in details. When the NFS setattr call fails because an NFS clients attempting to access a file to which it does not have access. This message is harmless, but we can conclude that many such log entries might indicate a systematic attempt to compromise the system. The most common message occurs when older NFS startup scripts try to start newer versions of rpc.lockdmanually ,is the rpc.lockd startup failure message. To avoid this failure message edit the startup scripts and remove statements that attempt to start lockd manually.
If you transfer very large files via NFS, and NFS consumes all of the available CPU cycles, causing the server to respond at a glacial pace, you are probably running an older version of the kernel that has problems with the fsync call that accumulates disk syncs before flushing the buffers. This issue is reportedly fixed in the 2.4 kernel series, so upgrading the kernel may solve the problem.

Example NFS client

The NFS server configured in the previous section exported /home and /usr/local,so I will demonstrate configuring an NFS client that mounts those directories.

- Clients that want to use both exports need to have the following entries in

/etc/fstab:
/usr/local nfs
rsize=8192,wsize=8192,hard,intr,nolock 0 0
luther:/home /home nfs
rsize=8192,wsize=8192,hard,intr,nolock 0 0

- Start the portmapper using the following command:

/etc/rc.d/init.d/portmap start
Starting portmapper: [OK]

- Mount the exports using one of the following commands:

mount –a –t nfs
or
mount /home /usr/local

The first command mounts all (-a) directories of type nfs (-t nfs). The second command mounts only the file systems /home and /usr/local. Verify that the mounts completed successfully by attempting to access files on each file system. If everything works as designed, you are ready to go. Otherwise, read the section titled "Troubleshooting NFS" for tips and suggestions for solving common NFS problems.

5.15 Using Automount Services

In some cases, putting your NFS mounts in /etc/fstab works just fine. In other cases, it doesn't work well, and you'll need a better way to mount NFS shares. An example of such a scenario is a network where users are using OpenLDAP to authenticate, after which they get access to their home directories. To make sure users can log in on different workstations and still get access to their home directory, you can't just put an NFS share in /etc/fstab for each user. Automount is a service that mounts NFS shares automatically. To configure it, you'll need to take care of three different steps:

- Start the autofs service.
- Edit the /etc/auto.master file.
- Create an indirect file to specify what you want Automount to do.
- The central configuration file in Automount is /etc/auto.master.

Sample /etc/auto.master

```
[root@hnl ~]# cat /etc/auto.master
#
# Sample auto.master file
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# For details of the format look at autofs(5)
#
/misc /etc/auto.misc
#
# NOTE: mounts done from a hosts map will be mounted with the
# "nosuid" and "nodev" options unless the "suid" and "dev"
# options are explicitly given.
#
/net -hosts
#
# Include central master map if it can be found using
# nsswitch sources.
#
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
+auto.master
[root@hnl ~]#
```

In /etc/auto.masterdirectories are specified that Automount should monitor. On every interesting directory is /net, which is monitored by the automount -hosts mechanism. This means that in /net, mounts to hosts that have NFS shares available will be mounted automatically.

5.16 Examining NFS Security

As explained, NFS has 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. Before warned, however, that no list of security tips, however comprehensive, makes your site completely secure and that plugging NFS security holes does not address other potential exploits.

General NFS security issues:

One NFS weakness, in general terms, is the /etc/exports file. If a cracker is capable of spoofing or taking over a trusted address, one listed in /etc/exports, then your mount points 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 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 as described earlier in the chapter to limit access to services on your system, particularly the portmapper, which has long been a target of exploit attempts. Similarly, you should put in entries for lockd, statd, mountd, and rquotad. More generally, wise application 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. Although netfilter is described in detail in Chapter 26, this section gives you a few tips on how to configure a netfilter firewall that plays nicely with NFS.

Server security considerations

On the server, always use the root squash option in /etc/exports. Actually, 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. 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 even access, much less change, files that only the server's root account can access orchange. The implication of root squashing may not be clear, so permit me to make it explicit: all critical binaries and files should be owned by root, not bin, wheel, admor another nonroot account. The only account that an NFS client's root user can not access is the server's root account, so critical files owned by root are much less exposed than if they are owned by other accounts.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.

Client security considerations

On the client, disable SUID (set UID) programs on NFS mounts using the nosuidoption. The nosuid option prevents a server's root account from creating an SUIDroot 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 that contain scripts or programs that need to be executed.

NFS version 3, the version available with Red Hat Linux (well, with version 2.4of the Linux kernel) supports NFS file locking. Accordingly, NFS clients must runstatd 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 forportmap, statd, and lockd on NFS clients that were suggested for the NFS server.

5.17 Self-Test (Multiple Choice Questions)

1. Your router has the following IP address on Ethernet0: 172.16.2.1/23. Which of the following can be valid host IDs on the LAN interface attached to the router?

   1. 172.16.1.100, 2. 172.16.1.198, 3. 172.16.2.255, 4. 172.16.3.0

   A. 1 only, B. 2 and 3 only, **C.** 3 and 4 only, None of above

2. Which two statements describe the IP address 10.16.3.65/23?
   1. The subnet address is 10.16.3.0 255.255.254.0.
   2. The lowest host address in the subnet is 10.16.2.1 255.255.254.0.
   3. The last valid host address in the subnet is 10.16.2.254 255.255.254.0.
   4. The broadcast address of the subnet is 10.16.3.255 255.255.254.0.

A. 1 and 3, **B.** 2 and 4, C. 1, 2 and 4, D. 2, 3 and 4

3. How long is an IPv6 address?

A. 32 bits, B. 128 bytes, C. 64 bits, **D.** 128 bits

5.18 Summary

In this chapter we have discussed various topics related to TCP/IP networking like Network Classes, subnetting, gateways, routers and Configuring Dynamic Host Configuration Protocol
In second part of this chapter we introduced the Network File System, installation process, configuration of NFS server and client. In continuation with this we have examined NFS security.

5.19 Exercise (short answer questions)

5.20 References

1. Red Hat® Linux® Networking and System Administration, Terry Collings and Kurt Wall
2. Red Hat ® Enterprise Linux® 6 Administration, Sander van Vugt
3. https://access.redhat.com/documentation/en-us/

Unit 6: Configuring DNS and DHCP

Unit 6: Configuring DNS and DHCP

6.1 Learning Objectives

After successful completion of this unit, you will be able to:

- Configure DNS
- Configure DHCP

6.2 Introduction

DNS associates hostnames with their respective IP addresses, so that when users want to connect to other machines on the network, they can refer to them by name, without having to remember IP addresses.

Use of DNS and FQDNs also has advantages for system administrators, allowing the flexibility to change the IP address for a host without affecting name-based queries to the machine. Conversely, administrators can shuffle which machines handle a name-based query.

DNS is normally implemented using centralized servers that are authoritative for some domains and refer to other DNS servers for other domains.

When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the FQDN based on its resolver library, which may contain authoritative information about the host requested or cached data from an earlier query. If the nameserver does not already have the answer in its resolver library, it queries other nameservers,

called root nameservers, to determine which nameservers are authoritative for the FQDN in question. Then, with that information, it queries the authoritative nameservers to determine the IP address of the requested host. If a reverse lookup is performed, the same procedure is used, except that the query is made with an unknown IP address rather than a name.

6.3 Introduction to DNS:

DNS is usually implemented using one or more centralized servers that are authoritative for certain domains. When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the name requested. If it does not have an authoritative answer, or does not already have the answer cached from an earlier query, it queries other nameservers, called root nameservers, to determine which nameservers are authoritative for the name in question, and then queries them to get the requested name.

6.4 The DNS Hierarchy

DNS uses a hierarchy to manage its distributed database system. The DNS hierarchy, also called the domain name space, is an inverted tree structure; much like eDirectory.The DNS tree has a single domain at the top of the structure called the root domain. A period or dot (.) is the designation for the root domain. Below the root domain are the top-level domains that divide the DNS hierarchy into segments.

Listed below are the top-level DNS domains and the types of organizations that use them. Below the top-level domains, the domain name space is further divided into subdomains representing individual organizations.

Domain	Used by
.com	Commercial organizations, as in novell.com
.edu	Educational organizations, as in ucla.edu
.gov	Governmental agencies, as in whitehouse.gov
.mil	Military organizations, as in army.mil
.org	Nonprofit organizations, as in redcross.org
.net	Networking entities, as in nsf.net
.int	International organizations, as in nato.int

Table 6.5.1: Top-Level DNS Domains

Additional top-level domains organize domain name space geographically. For example, the top-level domain for France is fr. DNS Hierarchy illustrates the DNS hierarchy.

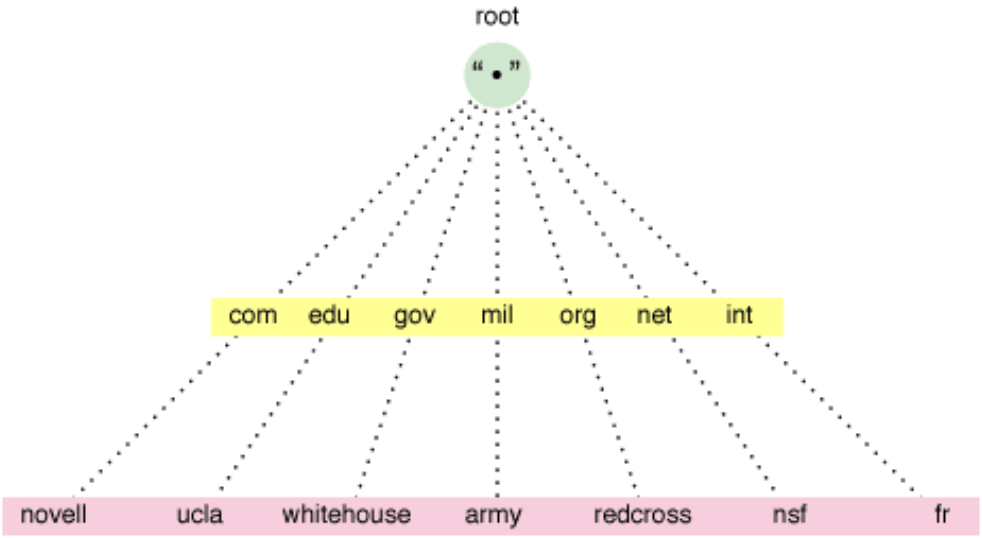


Figure 6.5.2: DNS Hierarchy

Domains and Subdomains

A domain is a label of the DNS tree. Each node on the DNS tree represents a domain. Domains under the top-level domains represent individual organizations or entities. These domains can be further divided into subdomains to ease administration of an organization's host computers.

For example, Company A creates a domain called companya.com under the .com top-level domain. Company A has separate LANs for its locations in Chicago, Washington, and Providence. Therefore, the network administrator for Company A decides to create a separate subdomain for each division, as shown in Domains and Subdomains

Any domain in a subtree is considered part of all domains above it. Therefore, chicago.companya.com is part of the companya.com domain, and both are part of the .com domain.

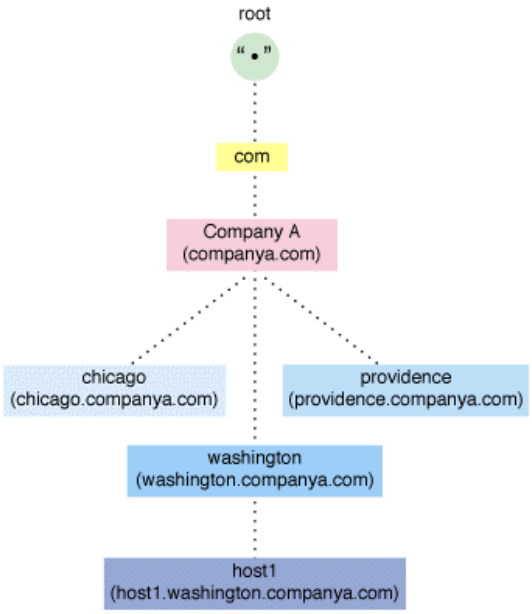


Figure 6.5.3: Domains and Subdomains

Domain Names

The domain name represents an entity's position within the structure of the DNS hierarchy. A domain name is simply a list of all domains in the path from the local domain to the root. Each label in the domain name is delimited by a period. For example, the domain name for the Providence domain within Company A is providence.companya.com, as shown in Domains and Subdomains and the list below.

Note that the domain names in the figure end in a period, representing the root domain. Domain names that end in a period for root are called fully qualified domain names (FQDNs).Each computer that uses DNS is given a DNS hostname that represents the computer's position within the DNS hierarchy. Therefore, the hostname for host1 in Figure 6.4.2 is host1.washington.companya.com.

6.5 DNS Server Types:

The DNS hierarchy is built by connecting name servers to one another. You can imagine that it is useful to have more than one name server per domain. Every zone has at least primary name

server, also referred to as the master name server. This is the server that is responsible for a zone and the one on which modifications can be made. To increase redundancy in case the master name server goes down, zones are also often configured with a secondary or slave name server. One DNS server can fulfill the role of both name server types. This means that an administrator can configure a server to be the primary nameserver for one domain and the secondary name server for another domain.

To keep the primary and secondary name servers synchronized, a process known as zone transfer is used. In a zone transfer, a primary server can push its database to the secondary name server, or the secondary name server can request updates from the primary name server. How this occurs depends on the way that the administrator of the name server configures it. In DNS traffic, both primary and secondary name servers are considered to be authoritative name servers. This means that if a client gets an answer from the secondary name server about a resource record within the zone of that name server, it is considered to be an authoritative reply. This is because the answer comes from a name server that has direct knowledge of the resource records in that zone. Apart from authoritative name servers, there are also recursive name servers. These are
name servers that are capable of giving an answer, but they don't get the answer from their own database. This is possible because, by default, every DNS name server caches its most recent request. How this works is explained in the following section.

6.6 The DNS Lookup Process

To get information from a DNS server, a client computer is configured with a DNS resolver. This is the configuration that tells the client which DNS server to use. If the client computer is a Linux machine, the DNS resolver is in the configuration file /etc/resolv.conf. When a client needs to get information from DNS, it will always contact the name server that is configured in the DNS resolver to request that information. Because each DNSserver is part of the worldwide DNS hierarchy, each DNS server should be able to handle client requests. In the DNS resolver, more than one name server is often configured to handle cases where the first DNS server in the list is not available. Let's assume that a client is in the example.com domain and wants to get the resource record for www.sander.fr. The following will occur:

a) When the request arrives at the name server of example.com, this name server will check its cache. If it has recently found the requested resource record, the name server will issue a recursive answer from cache, and nothing else needs to be done.
b) If the name server cannot answer the request from cache, it will first check whether a forwarder has been configured. A forwarder is a DNS name server to which requests are forwarded that cannot be answered by the local DNS server. For example, this can be the name server of a provider that serves many zones and that has a large DNS cache.
c) If no forwarder has been configured, the DNS server will resolve the name step-by step. In the first step, it will contact the name servers of the DNS root domain to find out how to reach the name servers of the .fr domain.
d) After finding out which name servers are responsible for the .fr domain, the local DNS server, which still acts on behalf of the client that issued the original request, contacts a name server of the .fr domain to find out which name server to contact to obtain information about the sander domain.

e) After finding the name server that is authoritative for the sander.fr domain, the name server can then request the resource record it needs. It will cache this resource record and send the answer back to the client.

6.7 DNS Zone Types:

Most DNS servers are configured to service at least two zone types. First there is the regular zone type that is used to find an IP address for a hostname. This is the most common use of DNS. In some cases, however, it is needed to find the name for a specific IP address. This type of request is handled by the in-addr.arpa zones.

Most DNS servers are configured to service at least two zone types. First there is the regular zone type that is used to find an IP address for a hostname. This is the most common use of DNS. In some cases, however, it is needed to find the name for a specific IP address. This type of request is handled by the in-addr.arpa zones.In in-addr.arpa zones, PTR resource records are configured. The name of the in-addr.arpa zone is the reversed network part of the IP address followed by in-addr.arpa. For example, if the IP address is 193.173.10.87, the in-addr.arpa zone would be 87.10.173.in-addr.arpa. The name server for this zone would be configured to know the names off all IP addresses within that zone. Although in-addr.arpa zones are useful, they are not always configured. The main reason is that DNS name resolving also works without in-addr.arpa zones; reverse name resolution is required in specific cases only.

6.8 Setting Up a DNS Server,

This section covers BIND (Berkeley Internet Name Domain), the DNS server included in Red Hat Enterprise Linux. It focuses on the structure of its configuration files, and describes how to administer it both locally and remotely.

6.8.1 Empty Zones

BIND configures a number of “empty zones” to prevent recursive servers from sending unnecessary queries to Internet servers that cannot handle them (thus creating delays and SERVFAIL responses to clients who query for them). These empty zones ensure that immediate and authoritative NXDOMAIN responses are returned instead. The configuration option empty-zones-enable controls whether or not empty zones are created, whilst the option disable-empty-zone can be used in addition to disable one or more empty zones from the list of default prefixes that would be used.

The number of empty zones created for RFC 1918 prefixes has been increased, and users of BIND 9.9 and above will see the RFC 1918 empty zones both when empty-zones-enable is unspecified (defaults to yes), and when it is explicitly set to yes.

Configuring the named Service

When the named service is started, it reads the configuration from the files as described in Table 6.8.1, “The named Service Configuration Files”.

Path	Description
/etc/named.conf	The main configuration file.
/etc/named/	An auxiliary directory for configuration files that are included in the main configuration file.

Table 6.8.1: “The named Service Configuration Files”.

The configuration file consists of a collection of statements with nested options surrounded by opening and closing curly brackets ({ and }). Note that when editing the file, you have to be careful not to make any syntax error, otherwise the named service will not start. A typical /etc/named.conf file is organized as follows:

```
statement-1 ["statement-1-name"] [statement-1-class] {
option-1;
option-2;
option-N;
};
statement-2 ["statement-2-name"] [statement-2-class] {
option-1;
option-2;
option-N;
};
statement-N ["statement-N-name"] [statement-N-class] {
option-1;
option-2;
option-N;
};
```

If you have installed the bind-chroot package, the BIND service will run in the chroot environment. In that case, the initialization script will mount the above configuration files using the mount --bind command, so that you can manage the configuration outside this environment. There is no need to copy anything into the /var/named/chroot/ directory because it is mounted automatically. This simplifies maintenance since you do not need to take any special care of BIND configuration files if it is run in a chroot environment. You can organize everything as you would with BIND not running in a chroot environment.

The following directories are automatically mounted into the /var/named/chroot/ directory if the corresponding mount point directories underneath /var/named/chroot/ are empty:

/etc/named
/etc/pki/dnssec-keys
/run/named
/var/named
/usr/lib64/bind or /usr/lib/bind (architecture dependent).

The following files are also mounted if the target file does not exist in /var/named/chroot/:

/etc/named.conf
/etc/rndc.conf
/etc/rndc.key
/etc/named.rfc1912.zones
/etc/named.dnssec.keys
/etc/named.iscdlv.key
/etc/named.root.key

Configuring a cache-only name server isn't difficult. You just need to install the BIND service and make sure that it allows incoming traffic. For cache-only name servers, it also makes sense to configure a forwarder.

6.9 Setting Up a Cache-Only Name Server

In this exercise, you'll install BIND and set it up as a cache-only name server. You'll also configure a forwarder to optimize speed in the DNS traffic on your network. To complete this exercise, you need to have a working Internet connection on your RHEL server.

1. Open a terminal, log in as root, and run yum -y install bind-chroot on the host computer to install the bind package.
2. With an editor, open the configuration file /etc/named.conf. Listing 14.1 shows a portion of this configuration file. You need to change some parameters in the configuration file to have BIND offer its services to external hosts.

```
[root@hnl ~]# vi /etc/named
named/ named.iscdlv.keynamed.root.key
named.conf named.rfc1912.zones
[root@hnl ~]# vi /etc/named.conf
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//
options {
listen-on port 53 { 127.0.0.1; };
listen-on-v6 port 53 { ::1; };
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
allow-query { localhost; };
recursion yes;
dnssec-enable yes;
dnssec-validation yes;
dnssec-lookaside auto;
/* Path to ISC DLV key */
bindkeys-file "/etc/named.iscdlv.key";
};
logging {
channeldefault_debug {
```

3. Change the file to include the following parameters: listen-on port 53 { any; }; and allow-query { any; };. This opens your DNS server to accept queries on any network inter face from any client.
4. Still in /etc/named.conf, change the parameter dnssec-validation; to dnsserver-validation no;.

5. Finally, insert the line forwarders x.x.x.x in the same configuration file, and give it the value of the IP address of the DNS server you normally use for your Internet connection. This ensures that the DNS server of your Internet provider is used for DNS recursion and that requests are not sent directly to the name servers of the root domain.
6. Use the service named restart command to restart the DNS server.
7. From the RHEL host, use dig redhat.com. You should get an answer, which is sent by your DNS server. You can see this in the SERVER line in the dig response. Congratulations, your cache-only name server is operational!

6.10 Setting Up a Primary Name Server

In the previous section, you learned how to create a cache-only name server. In fact, this is a basic DNS server that doesn't serve any resource records by itself. In this section, you'll learn how to set up your DNS server to serve its own zone.

To set up a primary name server, you'll need to define a zone. This consists of two parts. First you'll need to tell the DNS server which zones it has to service, and next you'll need to create a configuration file for the zone in question.

To tell the DNS server which zones it has to service, you need to include a few lines in/etc/named.conf. In these lines, you'll tell the server which zones to service and where the configuration files for that zone are stored. The first line is important. It is the directory line that tells named.conf in which directory on the Linux file system it can find its configuration. All filenames to which you refer later in named.conf are relative to that directory. By default, it is set to /var/named. The second relevant part tells the named process the zones it services. On Red Hat Enterprise Linux, this is done by including another file with the name /etc/named.rfc192.conf. Below Listing shows a named.conf for a name server that services the example.com domain. All relevant parameters have been set correctly in this example file.

Example named.conf:

```
[root@rhev ~]# cat /etc/named.conf
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//
options {
listen-on port 53 { any; };
listen-on-v6 port 53 { ::1; };
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
allow-query { any; };
forwarders { 8.8.8.8; };
recursion yes;
```

```
dnssec-enable yes;
dnssec-validation no;
dnssec-lookaside auto;
/* Path to ISC DLV key */
bindkeys-file "/etc/named.iscdlv.key";
managed-keys-directory "/var/named/dynamic";
};
logging {
channeldefault_debug {
file "data/named.run";
severity dynamic;
};
};
zone "." IN {
type hint;
file "named.ca";
};
include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

6.11 Setting Up a Secondary Name Server:

After setting up a primary name server, you should add at least one secondary name server. A secondary server is one that synchronizes with the primary. Thus, to enable this, you must first allow the primary to transfer data. You do this by setting the allow-transfer parameter for the zone as you previously defined it in the /etc/named.rfc1912.conf file. It's also a good idea to set the notify yes parameter in the definition of the master zone. This means that the master server automatically sends an update to the slaves if something has changed. After adding these lines, the definition for the example.com zone should appear as shown In below Listing.

Listing 6.11.1: Adding parameters for master-slave communication

```
zone "example.com" IN {
type master;
file "example.com";
notify yes;
allow-update { 192.168.1.70; };
};
```

Once you have allowed updates on the primary server, you need to configure the slave.This means that in the /etc/named.rfc1912.conf file on the Red Hat server, which you'regoing to use as DNS slave, you also need to define the zone. The example configuration inlisting 6.9.2 will do that for you.

Listing 6.11.2 : Creating a DNS slave configuration

```
zone "example.com" IN {
type slave;
masters {
192.168.1.220;
```

};
file "example.com.slave";
};
After creating the slave configuration, make sure to restart the named service to get itworking.

6.12 Understanding DHCP:

The Dynamic Host Configuration Protocol (DHCP) is a network management protocol networks where a DHCP server dynamically assigns an IP address and other network configuration parameters to each device on a network so they can communicate with other IP networks. A DHCP server enables computers to request IP addresses and networking parameters automatically from the Internet service provider (ISP), reducing the need for a network administrator or a user to manually assign IP addresses to all network devices. In the absence of a DHCP server, a computer or other device on the network needs to be manually assigned an IP address, which will not enable it to communicate outside its local subnet.

DHCP can be implemented on networks ranging in size from home networks to large campus networks and regional Internet service provider networks. A router or a residential gateway can be enabled to act as a DHCP server.
The DHCP operates based on the client–server model. When a computer or other device connects to a network, the DHCP client software sends a DHCP broadcast query requesting the necessary information. Any DHCP server on the network may service the request. The DHCP server manages a pool of IP addresses and information about client configuration parameters such as default gateway, domain name, the name servers, and time servers. On receiving a DHCP request, the DHCP server may respond with specific information for each client, as previously configured by an administrator, or with a specific address and any other information valid for the entire network and for the time period for which the allocation (lease) is valid. A DHCP client typically queries for this information immediately after booting, and periodically thereafter before the expiration of the information. When a DHCP client refreshes an assignment, it initially requests the same parameter values, but the DHCP server may assign a new address based on the assignment policies set by administrators.
DHCP clients obtain a DHCP lease for an IP address, a subnet mask, and various DHCP options from DHCP servers in a four-step process:

DHCPDISCOVER:
The client broadcasts a request for a DHCP server.

DHCPOFFER:
DHCP servers on the network offer an address to the client.

DHCPREQUEST:
The client broadcasts a request to lease an address from one of the offering DHCP servers.

DHCPACK:
The DHCP server that the client responds to acknowledges the client, assigns it any configured DHCP options, and updates its DHCP database. The client then initializes and binds its TCP/IP protocol stack and can begin network communication.

6.13 Setting Up a DHCP Server

To set up a DHCP server, after installing the dhcp package, you need to change common DHCP settings in the main configuration file: /etc/dhcp/dhcpd.conf. After installing the dhcp package, the file is empty, but there is a good annotated example file in /usr/share/doc/dhcp-<version>/dhcpd.conf.sample. You can see the default parameters from this file in listing 6.13.1.

Listing 6.13.1: Example dhcpd.conf file

```
[root@hnl dhcp-4.1.1]# catdhcpd
dhcpd6.conf.sample dhcpd.conf.sampledhcpd-conf-to-ldap
[root@hnl dhcp-4.1.1]# catdhcpd.conf.sample
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#
# option definitions common to all supported networks...
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;
default-lease-time 600;
max-lease-time 7200;
# Use this to enble / disable dynamic dns updates globally.
#ddns-update-style none;
# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;
# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;
# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.
subnet 10.152.187.0 netmask 255.255.255.0 {
}
# This is a very basic subnet declaration.
subnet 10.254.239.0 netmask 255.255.255.224 {
range 10.254.239.10 10.254.239.20;
option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;
}
# This declaration allows BOOTP clients to get dynamic addresses,
# which we don't really recommend.
subnet 10.254.239.32 netmask 255.255.255.224 {
range dynamic-bootp 10.254.239.40 10.254.239.60;
option broadcast-address 10.254.239.31;
option routers rtr-239-32-1.example.org;
}
# A slightly different configuration for an internal subnet.
subnet 10.5.5.0 netmask 255.255.255.224 {
range 10.5.5.26 10.5.5.30;
option domain-name-servers ns1.internal.example.org;
option domain-name "internal.example.org";
option routers 10.5.5.1;
option broadcast-address 10.5.5.31;
default-lease-time 600;
max-lease-time 7200;
```

```
}
# Hosts which require special configuration options can be listed in
# host statements. If no address is specified, the address will be
# allocated dynamically (if possible), but the host-specific information
# will still come from the host declaration.
host passacaglia {
hardwareethernet 0:0:c0:5d:bd:95;
filename "vmunix.passacaglia";
server-name "toccata.fugue.com";
}
# Fixed IP addresses can also be specified for hosts. These addresses
# should not also be listed as being available for dynamic assignment.
# Hosts for which fixed IP addresses have been specified can boot using
# BOOTP or DHCP. Hosts for which no fixed address is specified can only
# be booted with DHCP, unless there is an address range on the subnet
# to which a BOOTP client is connected which has the dynamic-bootp flag
# set.
host fantasia {
hardwareethernet 08:00:07:26:c0:a5;
fixed-address fantasia.fugue.com;
}
# You can declare a class of clients and then do address allocation
# based on that. The example below shows a case where all clients
# in a certain class get addresses on the 10.17.224/24 subnet, and all
# other clients get addresses on the 10.0.29/24 subnet.
class "foo" {
match if substring (option vendor-class-identifier, 0, 4) = "SUNW";
}
shared-network 224-29 {
subnet 10.17.224.0 netmask 255.255.255.0 {
option routers rtr-224.example.org;
}
subnet 10.0.29.0 netmask 255.255.255.0 {
option routers rtr-29.example.org;
}
pool {
allow members of "foo";
range 10.17.224.10 10.17.224.250;
}
pool {
deny members of "foo";
range 10.0.29.10 10.0.29.230;
}
}
```

Here are the most relevant parameters from the dhcpd.conffile and a short explanation of each: option domain-name Use this to set the DNS domain name for the DHCP clients.

option domain-name-servers: This specifies the DNS name servers that should be used.

default-lease-time :This is the default time in seconds that a client can use the IPaddress that it has received from the DHCP server.

max-lease-time :This is the maximum time that a client can keep on using its assigned IP address. If within the max-lease-time timeout it hasn't been able to contact the DHCPserver for renewal, the IP address will expire, and the client can't use it anymore.

log-facility :This specifies which syslog facility the DHCP server uses.

Subnet: This is the essence of the work of a DHCP server. The subnet definition specifies the network on which the DHCP server should assign IP addresses. A DHCP server can serve multiple subnets, but it is common for the DHCP server to be directly connected to the subnet it serves.
range :This is the range of IP addresses within the subnet that the DHCP server can assign to clients.

option routers :This is the router that should be set as the default gateway.

As you see from the sample DHCP configuration file, there are many options that an administrator can use to specify different kinds of information that should be handed out. Some options can be set globally and also in the subnet, while other options are set in specific subnets. As an administrator, you need to determine where you want to set specific options.
Apart from the subnet declarations that you make on the DHCP server, you can also define the configuration for specific hosts. In the example file in Listing6.13.1, you can see this in the host declarations for host passacaglia and host fantasia. Host declarations will work based on the specification of the hardware Ethernet address of the host; this is the MAC address of the network card where the DHCP request comes in.
At the end of the example configuration file, you can also see that a class is defined, as well as a shared network in which different subnets and pools are used. The idea is that you can use the class to identify a specific host. This works on the basis of the vendor class identifier, which is capable of identifying the type of host that sends a DHCP request. Once a specific kind of host is identified, you can match it to a class and, based on class membership, assign specific configuration that makes sense for that class type only.
At the end of the example dhcpd.conf configuration file, you can see that, on a shared network, two different subnets are declared where all members of the class for are assigned to one of the subnets and all others are assigned to the other class.

Starting and Stopping the Server:
To start the DHCP service, use the command /sbin/service dhcpd start. To stop the DHCP server, use the command /sbin/service dhcpd stop.
By default, the DHCP service does not start at boot time.
If more than one network interface is attached to the system, but the DHCP server should only be started on one of the interfaces, configure the DHCP server to start only on that device. In /etc/sysconfig/dhcpd, add the name of the interface to the list of DHCPDARGS:

```
# Command line options here
DHCPDARGS=eth0
```

Configuring a DHCP Client:

To configure a DHCP client manually, modify the /etc/sysconfig/network file to enable networking and the configuration file for each network device in the /etc/sysconfig/network-scripts directory. In this directory, each device should have a configuration file named ifcfg-eth0, where eth0 is the network device name.

The /etc/sysconfig/network file should contain the following line:

```
NETWORKING=yes
```

The NETWORKING variable must be set to yes if you want networking to start at boot time.
The /etc/sysconfig/network-scripts/ifcfg-eth0 file should contain the following lines:

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

A configuration file is needed for each device to be configured to use DHCP.

6.14 Self Test (Multiple Choice Questions)

1. DHCP (dynamic host configuration protocol) provides _____ to the client.
   a. IP Address, b, MAC Address, c, url, d. none of these
2. DHCP is used for _____
   a. IPv6, b. IPv4, **c.** Both IPv6 and IPv4, d. None of the mentioned
3. The DHCP server _____

a) maintains a database of available IP addresses, b) maintains the information about client configuration parameters, c) grants a IP address when receives a request from a client, **d)** all of the mentioned

6.15 Summary

In this chapter we have seen basics of DNS and DHCP. We have studied configuration and other details of both DNS and DHCP.

6.16 Exercise (short answer questions)

6.17 References

1. Red Hat® Linux® Networking and System Administration, Terry Collings and Kurt Wall
2. Red Hat ® Enterprise Linux® 6 Administration, Sander van Vugt
3. https://access.redhat.com/documentation/en-us/

Unit 7
Connecting to Microsoft Networks and Setting up a Mail Server

Samba

Samba is the standard Windows interoperability suite of programs for Linux and Unix.
Samba is Free Software licensed under the GNU General Public License, the Samba project is a member of the Software Freedom Conservancy.
Since 1992, Samba has provided secure, stable and fast file and print services for all clients using the SMB/CIFS protocol, such as all versions of DOS and Windows, OS/2, Linux and many others.
Samba is an important component to seamlessly integrate Linux/Unix Servers and Desktops into Active Directory environments. It can function both as a domain controller or as a regular domain member.
Samba is a software package that gives network administrators flexibility and freedom in terms of setup, configuration, and choice of systems and equipment. Because of all that it offers, Samba has grown in popularity, and continues to do so, every year since its release in 1992.

What Samba is All About
The commercialization of the Internet over the past few years has created something of a modern melting pot. It has brought business-folk and technologists closer together than was previously thought possible. As a side effect, Windows and Unix systems have been invading each others' turf, and people expect that they will not only play together nicely, but that they will share.

A lot of emphasis has been placed on peaceful coexistence between Unix and Windows. The Usenix Association has even created an annual conference (LISA/NT--July 14-17, 1999) around this theme. Unfortunately, the two systems come from very different cultures and they have difficulty getting along without mediation. ...and that, of course, is Samba's job. Samba runs on Unix platforms, but speaks to Windows clients like a native. It allows a Unix system to move into a Windows "Network Neighborhood" without causing a stir. Windows users can happily access file and print services without knowing or caring that those services are being offered by a Unix host.

All of this is managed through a protocol suite which is currently known as the "Common Internet File System", or CIFS. This name was introduced by Microsoft, and provides some insight into their hopes for the future. At the heart of CIFS is the latest incarnation of the Server Message Block (SMB) protocol, which has a long and tedious history. Samba is an open source CIFS implementation, and is available for free from the http://samba.org/ mirror sites.

Samba and Windows are not the only ones to provide CIFS networking. OS/2 supports SMB file and print sharing, and there are commercial CIFS products for Macintosh and other platforms (including several others for Unix). Samba has been ported to a variety of non-Unix operating systems, including VMS, AmigaOS, & NetWare. CIFS is also supported on dedicated file server platforms from a variety of vendors. In other words, this stuff is all over the place.

History

It started a long time ago, in the early days of the PC, when IBM and Sytec co-developed a simple networking system designed for building small LANs. The system included something called NetBIOS, or **Net**work **B**asic **I**nput **O**utput **S**ystem. NetBIOS was a chunk of software that was loaded into memory to provide an interface between programs and the network hardware. It included an addressing scheme that used 16-byte names to identify workstations and network-enabled applications. Next, Microsoft added features to DOS that allowed disk I/O to be redirected to the NetBIOS interface, which made disk space sharable over the LAN. The file-sharing protocol that they used eventually became known as SMB, and now CIFS.

Lots of other software was also written to use the NetBIOS API (**A**pplication **P**rogrammer's **I**nterface), which meant that it would never, ever, ever go away. Instead, the workings beneath the API were cleverly gutted and replaced. NetBEUI (NetBIOS Enhanced User Interface), introduced by IBM, provided a mechanism for passing NetBIOS packets over Token Ring and Ethernet. Others developed NetBIOS LAN emulation over higher-level protocols including DECnet, IPX/SPX and, of course, TCP/IP.

NetBIOS and TCP/IP made an interesting team. The latter could be routed between interconnected networks (internetworks), but NetBIOS was designed for isolated LANs. The trick was to map the 16-byte NetBIOS names to IP addresses so that messages could actually find their way through a routed IP network. A mechanism for doing just that was described in the Internet RFC1001 and RFC1002 documents. As Windows evolved, Microsoft added two additional pieces to the SMB package. These were service announcement, which is called "browsing", and a central authentication and authorization service known as Windows NT Domain Control.

More Systems

Andrew Tridgell, who is Australian, had a bit of a problem. He needed to mount disk space from a Unix server on his DOS PC. Actually, this wasn't the problem at all because he had an NFS (**N**etwork **F**ile **S**ystem) client for DOS and it worked just fine.

Unfortunately, he also had an application that required the NetBIOS interface. Anyone who has ever tried to run multiple protocols under DOS knows that it can be...er...quirky.

So Andrew chose the obvious solution. He wrote a packet sniffer, reverse engineered the SMB protocol, and implemented it on the Unix box. Thus, he made the Unix system appear to be a PC file server, which allowed him to mount shared filesystems from the Unix server while concurrently running NetBIOS applications. Andrew published his code in early 1992. There was a quick, but short succession of bug-fix releases, and then he put the project aside. Occasionally he would get E'mail about it, but he otherwise ignored it. Then one day, almost two years later, he decided to link his wife's Windows PC with his own Linux system. Lacking any better options, he used his own server code. He was actually surprised when it worked.

Through his E'mail contacts, Andrew discovered that NetBIOS and SMB were actually (though nominally) documented. With this new information at his fingertips he set to work again, but soon ran into another problem. He was contacted by a company claiming trademark on the name that he had chosen for his server software. Rather than cause a fuss, Andrew did a quick scan against a spell-checker dictionary, looking for words containing the letters "smb". "Samba" was in the list. Curiously, that same word is not in the dictionary file that he uses today. (Perhaps they know it's been taken.)

The Samba project has grown mightily since then. Andrew now has a whole team of programmers, scattered around the world, to help with Samba development. When a new release is announced, thousands of copies are downloaded within days. Commercial systems vendors, including Silicon Graphics, bundle Samba with their products. There are even Samba T-shirts available. Perhaps one of the best measures of the success of Samba is that it was listed in the "Halloween Documents", a pair of internal Microsoft memos that were leaked to the Open Source community. These memos list Open Source products which Microsoft considers to be competitive threats. The absolutely best measure of success, though, is that Andrew can still share the printer with his wife.

What Samba Does ?

Samba consists of two key programs, plus a bunch of other stuff that we'll get to later. The two key programs are smbd and nmbd. Their job is to implement the four basic modern-day CIFS services, which are:
File & print services
Authentication and Authorization
Name resolution
Service announcement (browsing)

File and print services are, of course, the cornerstone of the CIFS suite. These are provided by smbd, the SMB Daemon. Smbd also handles "share mode" and "user mode"

authentication and authorization. That is, you can protect shared file and print services by requiring passwords. In share mode, the simplest and least recommended scheme, a password can be assigned to a shared directory or printer (simply called a "share"). This single password is then given to everyone who is allowed to use the share. With user mode authentication, each user has their own username and password and the System Administrator can grant or deny access on an individual basis.

The Windows NT Domain system provides a further level of authentication refinement for CIFS. The basic idea is that a user should only have to log in once to have access to all of the authorized services on the network. The NT Domain system handles this with an authentication server, called a Domain Controller. An NT Domain (which should not be confused with a Domain Name System (DNS) Domain) is basically a group of machines which share the same Domain Controller.

The NT Domain system deserves special mention because, until the release of Samba version 2, only Microsoft owned code to implement the NT Domain authentication protocols. With version 2, Samba introduced the first non-Microsoft-derived NT Domain authentication code. The eventual goal, of course, it to completely mimic a Windows NT Domain Controller.

The other two CIFS pieces, name resolution and browsing, are handled by nmbd. These two services basically involve the management and distribution of lists of NetBIOS names.

Name resolution takes two forms: broadcast and point-to-point. A machine may use either or both of these methods, depending upon its configuration. Broadcast resolution is the closest to the original NetBIOS mechanism. Basically, a client looking for a service named Trillian will call out "Yo! Trillian! Where are you?", and wait for the machine with that name to answer with an IP address. This can generate a bit of broadcast traffic (a lot of shouting in the streets), but it is restricted to the local LAN so it doesn't cause too much trouble.

The other type of name resolution involves the use of an NBNS (NetBIOS Name Service) server. (Microsoft called their NBNS implementation WINS, for Windows Internet Name Service, and that acronym is more commonly used today.) The NBNS works something like the wall of an old fashioned telephone booth. (Remember those?) Machines can leave their name and number (IP address) for others to see.

> Hi, I'm node Voomba. Call me for a good time! 192.168.100.101

It works like this: The clients send their NetBIOS names & IP addresses to the NBNS server, which keeps the information in a simple database. When a client wants to talk to

another client, it sends the other client's name to the NBNS server. If the name is on the list, the NBNS hands back an IP address. You've got the name, look up the number.

Clients on different subnets can all share the same NBNS server so, unlike broadcast, the point-to-point mechanism is not limited to the local LAN. In many ways the NBNS is similar to the DNS, but the NBNS name list is almost completely dynamic and there are few controls to ensure that only authorized clients can register names. Conflicts can, and do, occur fairly easily.

Finally, there's browsing. This is a whole 'nother kettle of worms, but Samba's nmbd handles it anyway. This is not the web browsing we know and love, but a browsable list of services (file and print shares) offered by the computers on a network.

On a LAN, the participating computers hold an election to decide which of them will become the Local Master Browser (LMB). The "winner" then identifies itself by claiming a special NetBIOS name (in addition to any other names it may have). The LMBs job is to keep a list of available services, and it is this list that appears when you click on the Windows "Network Neighborhood" icon.

In addition to LMBs, there are Domain Master Browsers (DMBs). DMBs coordinate browse lists across NT Domains, even on routed networks. Using the NBNS, an LMB will locate its DMB to exchange and combine browse lists. Thus, the browse list is propagated to all hosts in the NT Domain. Unfortunately, the synchronization times are spread apart a bit. It can take more than an hour for a change on a remote subnet to appear in the Network Neighborhood.

Samba is a powerful suite of applications for allowing UNIX-based systems (such as Linux) to interoperate with Windows-based and other operating systems. It is an open source implementation of the Server Message Block/Common Internet File System (SMB/CIFS) protocol suite.

Samba transparently provides file and print sharing services to Windows clients. It is able to do this through the use of the native Microsoft networking protocols SMB/CIFS. From a system administrator’s point of view, this means being able to deploy a UNIX-based server without having to install Network File System (NFS), and some kind of UNIX-compatible authentication support on all the Windows clients in the network. Instead, the clients can use their native tongue to talk to the server which means fewer hassles for you and seamless integration for your users.

This chapter covers the procedure for downloading, compiling, and installing Samba. Thankfully, Samba’s default configuration requires little modification, so we’ll concentrate on how to perform customary tasks with it and how to avoid some common pitfalls. In terms of administration, you’ll get a short course on using Samba’s Web Administration Tool (SWAT) and on the **smbclien**t command-line utility.

No matter what task you've chosen for Samba to handle, be sure to take the time to read the program's documentation. It is well written, complete, and thorough. For the short afternoon it takes to get through most of it, you'll gain a substantial amount of knowledge.

THE MECHANICS OF SMB

To fully understand the Linux/Samba/Windows relationship, you need to understand the relationships of the operating systems to their files, printers, users, and networks. To better see how these relationships compare, let's examine some of the fundamental issues of working with both Linux-based systems and Windows in the same environment.

Usernames and Passwords

The Linux/UNIX login/password mechanism is radically different from the Windows PDC (Primary Domain Controller) model and the Windows Active Directory model. Thus, it's important for the system administrator to maintain consistency in the logins and passwords across both platforms. Users may need to work in heterogeneous environments and may need access to the different platforms for various reasons. It is thus useful to make working in such environments as seamless as possible without having to worry about users needing to re-authenticate separately on the different platforms or worry about cached passwords that don't match between servers, etc.

Relative to Samba, there are several options for handling username and password issues in heterogeneous environments. Some of these are:

- **The Linux Pluggable Authentication Modules (PAM) -** Allows you to authenticate users against a PDC. This means you still have two user lists—one local and one on the PDC—but your users need only keep track of their passwords on the Windows system.
- **Samba as a PDC -** Allows you to keep all your logins and passwords on the Linux system, while all your Windows boxes authenticate with Samba. When Samba is used with a Lightweight Directory Access Protocol (LDAP) back-end for this, you will have a scalable and extensible solution.
- **Roll your own solution using Perl -** Allows you to use your own custom script. For sites with a well-established system for maintaining logins and passwords, it isn't unreasonable to come up with a custom script. This can be done using WinPerl and Perl modules that allow changes to the Security Access Manager (SAM) to update the PDC's password list. A Perl script on the Linux side can communicate with the WinPerl script to keep accounts synchronized.

In the worst-case situation, you can always maintain the username and password databases of the different platforms by hand (which some early system admins did indeed have to do!), but this method is error-prone and not much fun to manage.

Encrypted Passwords

Starting with Windows NT 4/Service Pack 3, Windows 98, and Windows 95 OSR2, Windows uses encrypted passwords when communicating with the PDC and any server requiring authentication (including Linux and Samba). The encryption algorithm used by Windows is different from UNIX's, however, and, therefore, is not compatible.
Here are your choices for handling this conflict:

- Edit the Registry on Windows clients to disable the use of encrypted passwords. The Registry entries that need to be changed are listed in the **docs** directory in the Samba package. As of version 3 of Samba, this option is no longer necessary.
- Configure Samba to use Windows-style encrypted passwords.

The first solution has the benefit of not pushing you to a more complex password scheme. On the other hand, you may have to apply the Registry fix on all your clients. The second option, of course, has the opposite effect: For a little more complexity on the server side, you don't have to modify any of your clients.

Samba Daemons

The Samba code is actually composed of several components and daemons. We will examine three of the main daemons here, namely, smbd,nmbd, and winbindd
The **smbd** daemon handles the actual sharing of file systems and printer services for clients. It is also responsible for user authentication and resource-locking issues. It Starts by binding to port 139 or port 445 and then listens for requests. Every time a client authenticates itself, **smbd** makes a copy of itself; the original goes back to listening to its primary port for new requests, and the copy handles the connection for the client. This new copy also changes its effective user ID from root to the authenticated user. (For example, if the user yyang authenticated against **smbd**, the new copy would run with the permissions of yyang, not the permissions of root.) The copy stays in memory as long as there is a connection from the client.
The **nmbd** daemon is responsible for handling NetBIOS name service requests. **Nmbd**can also be used as a drop-in replacement for a Windows Internet Name Server (WINS). It begins by binding itself to port 137; unlike **smbd**, however, **nmbd** does not create a new instance of itself to handle every query. In addition to name service requests, **nmbd** handles requests from master browsers, domain browsers, and WINS servers—and as such, it participates in the browsing protocols that make up the popular Windows Network Neighborhood of systems. The services provided by the **smbd** and **nmbd** daemons complement each other.
Finally, the service provided by **winbindd** can be used to query native Windows servers for user and group information, which can then be used on purely Linux/UNIX platforms. It does this by using Microsoft Remote Procedure Call (RPC) calls, PAM, and the name service switch (NSS) capabilities found in modern C libraries. Its use can be extended through the use of a PAM module (**pam_winbind**) to provide

authentication services. This service is controlled separately from the main **smb** service and can run independantly.

Installing Samba via RPM

Precompiled binaries for Samba exist for most Linux distributions. This section will show how to install Samba via Red Hat Package Manager (RPM) on a Fedora distribution. To provide the server-side services of Samba, three packages are needed on Fedora and RedHat Enterprise Linux (RHEL)–type systems. They are

- **samba*.rpm -** This package provides an SMB server that can be used to provide network services to SMB/CIFS clients.
- **samba-common*.rpm -**This package provides files necessary for both the server and client packages of Samba—files such as configuration files, log files, man pages, PAM modules, and other libraries.
- **samba-client*.rpm -** It provides the SMB client utilities that allow access to SMB shares and printing services on Linux and non-Linux-type system. The package is used on Fedora, OpenSuSE, and other RHEL-type systems.

Assuming you have a working connection to the Internet, installing Samba can be as simple as issuing this command:

[root@serverA ~]# **yum -y install samba**
You can similarly install the samba-client package like so:
[root@serverA ~]# **yum -y install samba-client**
You may also choose to install the RPM package from the distribution's install media's **/mount_point/Packages/**directory using the usual RPM commands, e.g.,
[root@serverA ~]# **rpm -ivh /media/dvdrom/Packages/samba-*.rpm**

Installing Samba via APT

The essential components of the Samba software on Debian-like distros, such as Ubuntu, are split into **samba*.deb** and **samba-common*.deb** packages. Getting the client and server components of Samba installed in Ubuntu is easy as running the following **apt-get** command:
yyang@ubuntu-serverA:~$ **sudo apt-get -y install samba**
As with installing most other services under Ubuntu, the installer will automatically start the Samba daemons after installation.

Compiling and Installing Samba from Source

Samba comes prepackaged in binary format on most Linux distributions. Since its inception, Samba has had users across many different UNIX/Linux platforms and so has

been designed to be compatible with the many variants. There is rarely a problem during the compilation process.
As of this writing, the latest version of Samba was 3.2.0. You should therefore remember to change all references to the version number (3.2.0) in the following steps to suit the version you are using.
Begin by downloading the Samba source code from www.samba.org into the directory where you want to compile it. For this example, we'll assume this directory is **/usr/local/src**. You can download the latest version directly from http://us4.samba.org/samba/ftp/samba-latest.tar.gz.

1. Unpack Samba using the **tar** command.
[root@serverA src]# **tar xvzf samba-latest.tar.gz**
2. Step 1 creates a subdirectory called **samba-3.2.0** for the source code. Change into that directory. Type
[root@serverA src]# **cd samba-3.2.0/**
3. Within the samba-3.2.0 directory, there will be another subdirectory called **source**. Changeinto that directory like so:
[root@serverA samba-3.2.0]# cd source/
4. We'll run Samba's configure script and enable support for **smbmount**. Here we'll enable
only the **smbmount** option and accept the other defaults.
Type [root@serverA source]# **./configure –with-smbmount**
5. Begin compiling Samba by running the **make** command.
[root@serverA source]# **make**
6. Next, run **make install.**
[root@serverA source]# **make install**
7. We are done. You will find all the Samba binaries and configuration files installed under the **/usr/local/samba/** directory. You can now carry on using them as you would if you had installed Samba via RPM. Of course, you should watch out for the paths!

SAMBA ADMINISTRATION

This section describes some typical Samba administrative functions. We'll see how to start and stop Samba, how to do common administrative tasks with SWAT, and how to use smbclient. Finally, we'll examine the process of using encrypted passwords.

Samba Configuration (./configure) Option	**Description**
--prefix=PREFIX	Install architecture-independent files in PREFIX.
--with-smbmount	Include support for the **smbmount** command. The **smbmount**

	command allows you to attach shares off of NT servers (or other Samba servers), much as you mount NFS partitions.
--with-pam	Include PAM support (default=no).
--with-ldapsam	Include LDAP SAM 2.2–compatible configuration (default=no).
--with-ads	Active Directory support (default=auto).
--with-ldap	LDAP support (default=yes).
--with-pam_smbpass	Build PAM module for authenticating against **passdb** back-ends.
--with-krb5=base-dir	Locate Kerberos 5 support (default=/usr).
--enable-cups	Turn on Common UNIX Printing System (CUPS) support (default=auto).

Table 7.1 - Common Samba Configuration (./configure) Options

Starting and Stopping Samba

Most distributions of Linux have scripts and programs that will start and stop Samba without your needing to do anything special. They take care of startup at boot time and stopping at shutdown. On our sample system running Fedora with Samba installed via RPM, the **service** command and the **chkconfig** utility can be used to manage Samba's startup and shutdown.

For example, to start the **smbd** daemon, you can execute this command:
[root@serverA ~]# **service smb status**
And to stop the service, type
[root@serverA ~]# **service smb stop**
After making any configuration changes to Samba, you can restart it with this command to make the changes go into effect:
[root@serverA ~]# **service smb restart**
The **smb** service on Fedora will not automatically start up with the next system reboot. You can configure it to start up automatically using the **chkconfig** utility, like so:
[root@serverA ~]# **chkconfig smb on**

Starting the Samba that we installed from source earlier can be done from the command line with this command:

[root@serverA ~]# **/usr/local/samba/sbin/smbd -D**

The only command-line parameter used here (**-D**) tells **smbd** to run as a daemon. The **nmbd** daemon can be started in the same manner with

[root@serverA ~]# **/usr/local/samba/sbin/nmbd -D**

Stopping Samba without the use of proper scripts is a little trickier. But in general, you may have to use the **ps** command to list all of the Samba processes. From this list, find the instance of **smbd** that is owned by root and kill this process. This will also kill all of the other Samba connections.

USING SWAT

As mentioned, SWAT is the Samba Web Administration Tool, with which you can manage Samba through a browser interface. It's an excellent alternative to editing the Samba configuration files (**smb.conf** and the like) by hand.

Prior to version 2.0 of Samba, the official way to configure it was by editing the **smb.conf** file. Though verbose in nature and easy to understand, this file was rather cumbersome to deal with because of its numerous options and directives. Having to edit text files by hand also meant that setting up shares under Microsoft Windows was still easier than setting up shares with Samba. Some individuals developed graphical front-ends to the editing process. Many of these tools are still being maintained and enhanced. As of version 2.0, however, the source for Samba ships with SWAT.

The SWAT software is packaged separately on Fedora and RHEL systems. The binary RPM that provides SWAT is named **samba-swat.** In this section, we'll install the RPM for SWAT using the Yum program.

Setting Up SWAT

What makes SWAT a little different from other browser-based administration tools is that it does not rely on a separate web server (like Apache). Instead, SWAT performs all the needed web server functions without implementing a full web server.

Setting up SWAT is pretty straightforward. Here are the steps:

1. Use Yum to download and install SWAT. Type

[root@serverA ~]# **yum -y install samba-swat**

2. Confirm that you have the **samba-swat** package installed. Type

[root@serverA ~]# **rpm -q samba-swat**

samba-swat-3.*

3. SWAT runs under the control of the superdaemon, xinetd. It is disabled by default. Check its status by typing

[root@serverA ~]# **chkconfig --list swat**

swat off

4. Enable it by typing

[root@serverA ~]# **chkconfig swat on**

5. Restart xinetd to make your changes take effect. Type

[root@serverA ~]# **service xinetd restart**

6. Finally, you can connect to SWAT's web interface using a web browser on the system where

it is installed. Point the web browser to SWAT's Uniform Resource Locator (URL): http://localhost:901/

Upon entering this URL, you will be prompted for a username and password with which to log into SWAT. Type **root** as the username and type root's password. Upon successfully logging in, you will be presented with a web page similar to the one in Figure 7.1.

And that is pretty much all there is to installing and enabling SWAT on a Fedora system.

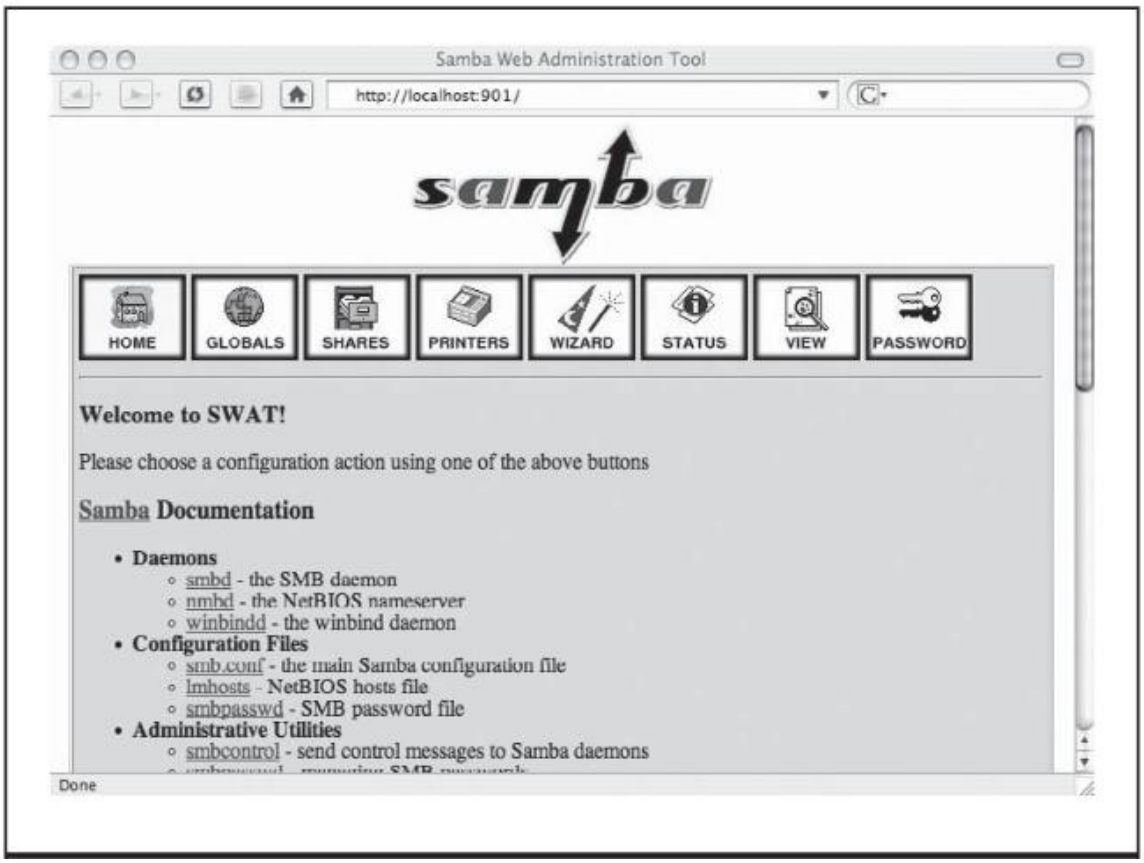


Fig.7.1 - Samba Web Administration Tool

THE SWAT MENUS

When you connect to SWAT and log in as root, you'll see the main menu shown in Figure 7.1. From here, you can find almost all the documentation you'll need for Samba's configuration files, daemons, and related programs. None of the links point to external web sites, so you can read them at your leisure without connecting to the Net. At the top of SWAT's main page are buttons for the following menu choices:

- Home The main menu page
- Globals Configuration options that affect all operational aspects of Samba
- Shares For setting up disk shares and their respective options
- Printers For setting up printers
- Wizard This will initiate a Samba configuration wizard that will walk you through setting up the Samba server
- Status The status of the smbd and nmbd processes, including a list of all clients connected to these processes and what they are doing (the same information that's listed in the smbstatus command-line program)
- View The resulting smb.conf file
- Password Password settings

Globals

The Globals page lists the settings that affect all aspects of Samba's operation. These settings are divided into five groups: base, security, logging, browse, and WINS. To the left of each option is a link to the relevant documentation for the setting and its values.

Shares

In Microsoft Windows, setting up a share can be as simple as selecting a folder (or creating a new one), right-clicking it, and allowing it to be shared. Additional controls can be established by right-clicking the folder and selecting Properties.Using SWAT, these same actions are accomplished by creating a new share. You can then select the share and click Choose Share. This brings up all the configurable parameters for the share.

Printers

The Printers page for SWAT lets you configure Samba-related setting for printers that are currently available on the system. Through a series of menus, you can add printer shares, delete them, modify them, etc. The one thing you cannot do here is add printers to the main system—you must do that by some other means.

Status

The Status page shows the current status of the smbd and nmbd daemons. This information includes what clients are connected and their actions. The page automatically updates every 30 seconds by default, but you can change this rate if you like (it's an option on the page itself). Along with status information, you can turn Samba on and off or ask it to reload its configuration file. This is necessary if you make any changes to the configuration.

View

As you change your Samba configuration, SWAT keeps track of the changes and figures out what information it needs to put into the smb.conf file. Open the View page, and you can see the file SWAT is putting together for you.

Password

Use the Password page if you intend to support encrypted passwords. You'll want to give your users a way to modify their own passwords without having to log into the Linux server. This page allows users to do just that.

CREATING A SHARE

We will walk through the process of creating a share under the /tmp directory to be shared on the Samba server. We'll first create the directory to be shared and then edit Samba's configuration file (/etc/samba/smb.conf) to create an entry for the share.

Of course, this can be done easily using SWAT's web interface, which was installed earlier, but we will not use SWAT here. SWAT is easy and intuitive to use. But it is probably useful to understand how to configure Samba in its rawest form, and this will also make it easier to understand what SWAT does in its back-end so that you can tweak things to your liking. Besides, one never knows when one might be stranded in the Amazon jungle without any nice graphical user interface (GUI) configuration tools available. So let's get on with it:

1. Create a directory under the **/tmp/**folder called **testshare**. Type

[root@serverA ~]# **mkdir /tmp/testshare**

2. Create some empty files (**foo1,foo2,moo3**) under the directory you created in Step 1. Type

[root@serverA ~]# **touch /tmp/testshare/{foo1,foo2,moo3}**

3. Set up the permissions on the **testshare** folder so that its contents can be browsed by other users on the system. Type

[root@serverA ~]# **chmod -R 755 /tmp/testshare/***

4. Open up Samba's configuration file for editing in any text editor of your choice, and append the entry listed next to the end of the file. Please omit the line numbers 1–5. The lines are added only to aid readability.

1) [samba-share]
2) comment=This folder contains shared documents
3) path=/tmp/testshare
4) public=yes
5) writable=no

- Line 1 is the name of the share (or "service" in Samba parlance). This is the name that SMB clients will see when they try to browse the shares stored on the Samba server.
- Line 2 is just a descriptive/comment text that users will see next to a share when browsing.
- Line 3 is important. It specifies the location on the file system that stores the actual content to be shared.

- Line 4 specifies that no password is required to access the share (this is called "connecting to the service" in Samba-speak). The privileges on the share will be translated to the permissions of the guest account. If the value were set to "no" instead, the share would not be accessible by the general public, but only by authenticated and permitted users.
- Line 5, with the value of the directive set to "no," means that users of this service may not create or modify the files stored therein.

5. Save your changes to the /etc/samba/smb.conf file, and exit the editor.You should note that we have accepted all the other default values in the file. You may want to go back and personalize some of the settings to suit your environment.

One setting you may want to change quickly is the directive ("workgroup") that defines the workgroup. This controls what workgroup your server will appear to be in when queried by clients or when viewed in the Windows Network Neighborhood.

Also note that the default configuration may contain other share definitions. You should comment (or delete) those entries if it is not your intention to have them.

6. Use the **testparm** utility to check the **smb.conf** file for internal correctness (i.e., absence of syntax errors). Type

[root@serverA ~]# **testparm –s | less**

...<OUTPUT TRUNCATED>...

[samba-share]

comment = This folder contains shared documents

path = /tmp/testshare

guest ok = Yes

Study the output for any serious errors, and try to fix them by going back to correct them in the **smb.conf** file.

Note that because you piped the output of **testparm** to the **less** command,you may have to press q on your keyboard to quit the command.

7. Now restart (or start) Samba to make the software acknowledge your changes. Type

[root@serverA ~]# **service smb restart**

We are done creating our test share. In the next section, we will attempt to access the share.

Using smbclient

The **smbclient** program is a command-line tool that allows your Linux-based system to act as a Windows client. You can use this utility to connect to other Samba servers or even to actual Microsoft Windows servers. **smbclient** is a flexible program and can be used to browse other servers, send and retrieve files from them, or even print to them. As you can imagine, this is also a great debugging tool, since you can quickly and easily check whether a new Samba installation works correctly without having to find a Windows client to test it.

In this section, we'll show you how to do basic browsing, remote file access, and remote printer access with **smbclient.** However, remember that **smbclient** is a flexible program, limited only by your imagination.

CREATING SAMBA USERS

When configured to do so, Samba will honor requests from users that are stored in user databases that are, in turn, stored in various back-ends—e.g., LDAP (**ldapsam,tdbsam,xmlsam**) or MySQL (**mysqlsam**).
Here, we will add a sample user that already exists in the local /**etc/passwd** file to Samba's user database. We will accept and use Samba's native/default user database back-end (**tdbsam**) for demonstration purposes, as the other possibilities are beyond the scope of this chapter.
Let's create a Samba entry for the user yyang. We will also set the user's Samba password. Use the **smbpasswd** command to create a Samba entry for the user yyang. Choose a good password when prompted to do so. Type

```
[root@serverA ~]# smbpasswd   -a yyang
New SMB password:
Retype new SMB password:
Added user yyang.
```

The new user will be created in Samba's default user database, **tdbsam**. With a Samba user now created, you can make the shares available to only authenticated users, such as the one we just created for the user yyang.
If the user yyang now wants to access a resource on the Samba server that has been configured strictly for her use (a protected share or nonpublic share), the user can use the **smbclient** command shown here; for example,

```
[root@clientB ~]# smbclient  –U yyang  –L  //serverA
```

It is, of course, also possible to access a protected Samba share from a native Microsoft Windows box. One only needs to supply the proper Samba username and corresponding password when prompted on the Microsoft Windows system.

Allowing Null Passwords

If you need to allow users to have no passwords (which is a bad idea, by the way, but for which there might be legitimate reasons), you can do so by using the **smbpasswd** program with the **-n** option, like so:

```
[root@serverA ~]# smbpasswd -n username
```

where **username** is the name of the user whose password you want to set to empty.For example, to allow the user yyang to access a share on the Samba server with a null password, type

[root@serverA ~]# **smbpasswd -n yyang**

User yyang password set to none.
You can also do this via the SWAT program using its web interface.

Changing Passwords with smbpasswd

Users who prefer the command line over the web interface can use the **smbpasswd** command to change their Samba passwords. This program works just like the regular **passwd** program, except this program does not update the **/etc/passwd** file by default. Because **smbpasswd** uses the standard protocol for communicating with the server regarding password changes, you can also use this to change your password on a remote Windows machine.
For example, to change the user yyang's Samba password, issue this command:

[root@serverA ~]# **smbpasswd yyang**
New SMB password:
Retype new SMB password:

Samba can be configured to allow regular users to run the **smbpasswd** command themselves to manage their own passwords; the only caveat is that they must know their previous/old password.

USING SAMBA TO AUTHENTICATE AGAINST A WINDOWS SERVER

Thus far, we've been talking about using Samba in the Samba/Linux world. Or, to put it literarily, we've been using Samba in its native environment, where it is lord and master of its domain (no pun intended). What this means is that our Samba server, in combination with the Linux-based server, has been responsible for managing all user authentication and authorization issues.
The simple Samba setup that we created earlier in the chapter had its own user database, which mapped the Samba users to real Linux/UNIX users. This allowed any files and directories created by Samba users to have the proper ownership contexts. But what if we wanted to deploy a Samba server in an environment with existing Windows servers that are being used to manage all users in the domain? And we don't want to have to manage a separate user database in Samba? Enter ...the **winbindd** daemon.
The **winbindd** daemon is used for resolving user accounts (users and groups) information from native Windows servers. It can also be used to resolve other kinds of system information. It is able to do this through its use of **pam_winbind** (a PAM module that

interacts with the **winbindd** daemon to help authenticate users using Windows NTLM authentication), the **ntlm_auth** tool (a tool used to allow external access to **winbind**'s NTLM authentication function), and **libnss_winbind** (**winbind**'s Name Service Switch library) facility.

The steps to set up a Linux machine to consult a Windows server for its user authentication issues are straightforward. They can be summarized in this way:

1. Configure Samba's configuration file (**smb.conf**) with the proper directives.
2. Add winbind to the Linux system's name service switch facility (/**etc/nsswitch.conf**).
3. Join the Linux/Samba server to the Windows domain.
4. Test things out.

Here we present a sample scenario where a Linux server named serverA wishes to use a Windows server for its user authentication issues. The Samba server is going to act as a Windows domain member server. The Windows server we assume here is running the Windows 200x Server operating system, and it is a domain controller (as well as the WINS server). Its IP address is 192.168.1.100. The domain controller is operating in mixed mode. (Mixed mode operation provides backward compatibility with Windows NT–type domains, as well as Windows 200x–type domains.) The Windows domain name is "WINDOWS-DOMAIN." We have commented out any share definitions in our Samba configuration, so you'll have to create or specify your own (see the earlier parts of the chapter for how to do this). Let's break down the process in better detail:

1. First, create an smb.conf file similar to this one:

```
#Sample smb.conf file
[global]
workgroup = WINDOWS-DOMAIN
security = DOMAIN
username map = /etc/samba/smbusers
log file = /var/log/samba/%m
smb ports = 139 445
name resolve order = wins bcast hosts
wins server = 192.168.1.100
idmap uid = 10000-20000
idmap gid = 10000-20000
template primary group = "Domain Users"
template shell = /bin/bash
winbind separator = +
# Share definitions
#[homes]
# comment = Home Directories
# browseable = no
# writable = yes
```

2. Edit the /**etc/nsswitch.conf** file on the Linux server so that it will have entries similar to this one:

passwd: files winbind
shadow: files winbind
group: files winbind

3. On Fedora, RHEL, and Centos distributions, start the **winbindd** daemon using the **service** command. Type

[root@serverA ~]# **service winbind start**
Starting Winbind services: [OK]

4. Join the Samba server to the Windows domain using the **net** command. Assuming the Windows Administrator account password, type

[root@serverA ~]# **net rpc join -U root% windows_administrator_password**
Joined domain WINDOWS-DOMAIN

where the password for the account in the Microsoft Windows domain with permission to join systems to the domain is windows_administrator_password.

5. Use the **wbinfo** utility to list all users available in the Windows domain to make sure that things are working properly. Type

[root@serverA ~]# **wbinfo -u**

TROUBLESHOOTING SAMBA

The following are a few typical solutions to simple problems one might encounter with Samba:

- **Restart Samba** This may be necessary because either Samba has entered an undefined state or (more likely) you've made major changes to the configuration but forgot to reload Samba so that the changes take effect.
- **Make sure the configuration options are correct** Errors in the **smb.conf** file are typically in directory names, usernames, network numbers, and hostnames. A common mistake is when a new client is added to a group that has special access to the server, but Samba isn't told the name of the new client being added. Don't forget that for syntax-type errors, the **testparm** utility is your ally.
- **Monitor encrypted passwords** These may be mismatched—the server is configured to use them and the clients aren't, or (more likely) the clients are using encrypted passwords and Samba hasn't been configured to use them. If you're under the gun to get a client working, you may just want to disable client-side encryption using the **regedit** scripts that come with Samba's source code (see the **docs** subdirectory).

Setting Up And Configuring A Linux Mail Server

Setting up Linux mail server and SMTP (Simple Mail Transfer Protocol) is essential if you want to use email, so we're going to look at how we can install and configure mail server along with some other email-related protocols, like Post Office Protocol (POP3) and Internet Message Access Protocol (IMAP).

Linux SMTP Server

SMTP stands for Simple Mail Transfer Protocol (**SMTP**) and it's used for transmitting electronic mail. It's platform-independent, so long as the server can send ASCII text and can connect to port 25 (the standard SMTP port).

Sendmail and **Postfix** are two of the commonest SMTP implementations and are usually included in most Linux distributions.

Sendmail is a free and popular mail server, but it's not all that secure and doesn't seem to have been designed for ease of use, which is to say that it's a bit tricky to get to grips with. Postfix is better in both these regards, however.

Linux Email Server Components

There are three components to a mail service on a Linux email server:

1. Mail user agent (MUA) is the GUI, the part that lets you write and send emails, like Thunderbird or Outlook.
2. Mail transport agent (MTA) is the bit that moves the mail (as the name suggests). MTAs like Sendmail and Postfix are the parts that waft your communications from place to place through the ether.
3. Mail delivery agent (MDA) is the component that sends out messages sent to you on your local machine, so they get to the appropriate user mailbox. Postfix-maildrop and Procmail are examples.

Setup Linux Email Server

In order to configure a Linux mail server, you'll first need to check if Postfix is already installed. It's the default mail server on the lion's share of Linux distributions these days, which is good because server admins like it a lot.

Here's how to check if it's already on the system:

```
$ rpm -qa | grep postfix
```

If not, this is how you install it on Red Hat distributions:

```
$ dnf -y install postfix
```

Next, run it and activate it on system start-up:

```
$ systemctl start postfix

$ systemctl activate postfix
```

For distributions based on Debian, like Ubuntu, you'd install them like this:

```
$ apt-get -y install postfix
```

As you configure Linux mail server you will receive a prompt to choose how you want to configure your Postfix mail server.
You'll be presented with these choices:

- No configuration
- Internet site
- Internet with smarthost
- Satellite system and Local only

Let's go with the No configuration option for our Linux email server.

Configure Linux Mail Server

After installing the Postfix mail server, you will need to set it up, and most of the files you'll need for this can be found inside the /etc/postfix/ directory.
You can find the main configuration for Postfix Linux mail server in the /etc/postfix/main.cf file.

This file contains numerous options like:

myhostname

Use this one to specify the hostname of the mail server, which is where postfix will obtain its emails. The hostnames will look something like mail.mydomain.com, smtp.mydomain.com.
You incorporate the hostname this way:

myhostname = mail.mydomain.com
exampledomain.com

This option is the mail domain that you will be servicing, like mydomain.com

The syntax looks like this:

mydomaindomain.com = mydomain.com

myorigin

All emails sent from this mail server will look as though they came from the one that you specify in this option. You can set this to $exampledomain.com.
myorigin = $exampledomain.com

Use any value that you want for this option but put a dollar sign in front of it like this: $exampledomain.com.

mydestination

This option shows you which domains the Postfix server uses for incoming emails to your Linux email server. You can assign values like this:

mydestination = $myhostname, localhost.$exampledomain.com, $exampledomain.com, mail.$exampledomain.com, www.$exampledomain.com

mail_spool_directory

A Postfix Linux mail server can use two modes of delivery:

- straight to someone's mailbox.
- to a central spool directory, which means the mail will sit in /var/spool/mail with a file for every user.

mail_spool_directory = /var/spool/mail

mynetworks

This will let you arrange which servers can relay through your Postfix server.
It should only take local addresses like local mail scripts on your server.

If this isn't the case, then spammers can piggyback on your Linux mail server. That means your lovely shiny server will be doing the heavy lifting for some bad guys and it will also end up getting banned.

Here's the syntax for this option:
mynetworks = 127.0.0.0/8, 192.168.1.0/24
smtpd_banner

This one determines what message is sent after the client connects successfully.

Consider changing the banner so it doesn't give away any potentially compromising information about your server.

inet_protocols

This option designates which IP protocol version is used for server connections.

inet_protocols = ipv4

When you change any of files used to configure Linux mail server for Postfix, you must reload the service, with this directive:

```
$ systemctl reload postfix
```

Of course, we all get distracted and typing things in can often result in mistakes, but you can track down any misspellings that might compromise your Linux mail server using this command:

```
$ postfix check
```

Checking the Mail Queue

Things like network failure (and many other reasons) can mean that the mail queue on your Linux email server can end up getting full, but you can check the Postfix mail queue with this command:

```
$ mailq
```

If that reveals that its full then you can flush the queue using this command:

```
$ postfix flush
```

Look at it again and you should see that your Linux email server queue is clear.

Test Linux Mail Server

Once your configuration is done you need to test your Linux mail server.

The first thing to do is use a local mail user agent such as mailx or mail which is a symlink to mailx.

Send your first test to someone on the Linux mail server and if that works then send the next one to somewhere external.

```
$ echo "This is the body of the message" | mailx -s "Here we have a Subject" -r "for instance <small example@mydomain.com>" -a /path/to/attachment someone@mydomain.com
```

Then check if your Linux email server can pick up external mail.

If you run into any snags, have a peek at the logs. The Red Hat log file can be found in /var/log/maillog and for Debian versions in /var/log/mail.log, or wherever else the rsyslogd configuration specifies.

I would suggest you review the Linux syslog server for an in-depth clarification on logs and how to set up rsyslogd.

If you run into any more difficulties, take a look at your DNS settings and use Linux network commands to check your MX records.

Fight Spam with SpamAssassin

Nobody likes spam, and SpamAssassin is probably the best free, open source spam fighting ninja that you could hope to have in your corner.
Installing it is as simple as doing this:

```
$ dnf -y install spamassassin
```

Then you just start the service and activate it at start-up:

```
$ systemctl start spamassassin

$ systemctl activate spamassassin
```

Once you've done that, you can see how it's configured in the /etc/mail/spamassassin/local.cf file.
SpamAssassin runs a number of scripts to test how spammy an email is. The higher the score that the scripts deliver, the more chances there are that it's spam.

In the configuration file, if the parameter required_hits is 6, this tells you that SpamAssassin will consider an email to be spam if it scores 6 or more.
The report_safe command will have values of 0, 1, or 2. A 0 tells you that email marked as spam is sent without modification, and only the headers will label it as spam.

A 1 or a 2 means that a new report message will be created by SpamAssassin and delivered to the recipient.
A value of 1 indicates that the spam message is coded as content message/rfc822, and if it's a 2, that means the message has been coded as text or plain content.

Text or plain is less dangerous because some mail clients execute message/rfc822, which is not good if they contain any kind of malware.

The next thing to do is integrate it into Postfix, and the easiest way to do that is with procmail. We'll make a file called/etc/procmailrc, and add this to it:

```
:0 hbfw | /usr/bin/spamc
```

Then we'll edit the Postfix configuration file /etc/postfix/main.cf and alter the mailbox_command, thus:

```
mailbox_command = /usr/bin/procmail
```

Last but not least, restart Postfix and SpamAssassin services:

```
$ systemctl restart postfix

$ systemctl restart spamassassin
```

Unfortunately, SpamAssassin can't catch everything, and spam messages can still sneak through to fill up the mailboxes on your Linux email server.

But never fear because you can filter messages before they even get to the Postfix server with Realtime Blackhole Lists (RBLs).

Open the Postfix server configuration at /etc/postfix/main.cf and change

smtpd_recipient_restrictions option by adding the following options like this:

strict_rfc821_envelopes = yes
relay_domains_reject_code = 554

unknown_address_reject_code = 554

unknown_client_reject_code = 554

unknown_hostname_reject_code = 554

unknown_local_recipient_reject_code = 554

unknown_relay_recipient_reject_code = 554

unverified_recipient_reject_code = 554

smtpd_recipient_restrictions =
reject_invalid_hostname,

reject_unknown_recipient_domain,

reject_unauth_pipelining,

permit_mynetworks,

permit_sasl_authenticated,

reject_unauth_destination,

reject_rbl_client dsn.rfc-ignorant.org,

reject_rbl_client dul.dnsbl.sorbs.net,

reject_rbl_client list.dsbl.org,
reject_rbl_client sbl-xbl.spamhaus.org,

reject_rbl_client bl.spamcop.net,

reject_rbl_client dnsbl.sorbs.net,

permit

Now, restart your postfix Linux mail server:

```
$ systemctl restart postfix
```

The above RBLs are the most common ones found, but there are plenty more on the web for you to track down and try.

POP3 and IMAP Protocol Basics

We now know how a SMTP Linux mail server sends and receives emails, but what about other user needs, like when they want local copies of emails to view off-line?
mbox file format isn't supported; it's used by many mail user agents such as mailx and mutt. Due to security concerns, some mail servers restrict access to the shared mail spool directories. Another class of protocols—called mail access protocols—was introduced to deal with such situations.

The commonest ones are POP and IMAP – Post Office Protocol and Internet Message Access Protocol. POP's underlying methodology is very simple: a central Linux mail server is online 24/7 for reception and storage of all user emails.

When an email is sent, the email client relays it through the central Linux mail server using SMTP. Be aware that the SMTP server and POP server can easily be on the same system, and that this is a common thing to do.
IMAP was developed because previously you couldn't keep a master copy of a user's email on the server.

With IMAP, your Linux email server supports three kinds of access:

- online mode is like having direct access to the Linux email server file system.
- offline mode feels like POP, where the client only connects to the network to get their mail, and the server won't keep a copy.
- disconnected mode lets users keep cached copies of their emails and the server keeps one too.

There are a few different implementations for IMAP and POP, with the most prevalent being dovecot server, which offers both.

POP3, POP3S, IMAP, and IMAPS listen on ports 110, 995, 143, and 993 respectively.

Dovecot Installation

Dovecot is preinstalled on the majority of Linux distributions, and there's no problem putting it in Red Hat too:

```
$ dnf -y install dovecot
```

For Debian, a pair of packages provide the IMAP and POP3 functionality. Here's how to install them:

```
$ apt-get -y install dovecot-imapd dovecot-pop3d
```

You will be prompted to create self-signed certificates for using IMAP and POP3 over SSL/TLS. Select yes and type in the hostname of your system when asked to do so.

Then you can run the service and activate it at start-up like this:

```
$ systemctl start dovecot
$ systemctl activate dovecot
```

Configure Dovecot

The main configuration file for Dovecot is /etc/dovecot/dovecot.conf file. Some varieties of Linux keep the configuration in the/etc/dovecot/conf.d/ directory and then have the include directive include the settings in the files.

Here are a few of the parameters used to configure dovecot:

protocols: the ones you want to support.
protocols = imap pop3 lmtp
lmtp stands for local mail transfer protocol.
listen: IP addresses to listen on.

listen = *, ::

The asterisk means all ipv4 interfaces and :: means all ipv6 interfaces
userdb: user database to authenticate users.

userdb { driver = pam }

passdb: password database two authenticate users.

passdb { driver = passwd }

mail_location: this entry is in the /etc/dovecot/conf.d/10-mail.conf file, and it's written like this:

mail_location = mbox:~/mail:INBOX=/var/mail/%u

Secure Dovecot

Dovecot features generic SSL certificates and key files used with /etc/dovecot/conf.d/10-ssl.conf

ssl_cert = </etc/pki/dovecot/certs/dovecot.pem

ssl_key = </etc/pki/dovecot/private/dovecot.pem

If you try to connect to a dovecot server and certificates haven't been signed, then you'll get a warning, but if you go to a certificate authority you can buy one, so no worries there.

Alternatively, you can point to them using Let's Encrypt certificates:

ssl_cert = </etc/letsencrypt/live/yourdomain.com/fullchain.pem

ssl_key = </etc/letsencrypt/live/yourdomain.com/privkey.pem

You'll need to open dovecot server ports in your iptables firewall by adding iptables rules for ports 110, 995, 143, 993, 25.

Do that and save the rules.

Or if you have a firewall then do this:

```
$ firewall-cmd --permanent --add-port=110/tcp --add-port=995/tcp
$ firewall-cmd --permanent --add-port=143/tcp --add-port=993/tcp
$ firewall-cmd --reload
```

Finally, for troubleshooting, check through the log files /var/log/messages, /var/log/maillog, and /var/log/mail.log files.

Linux mail server (and particularly Postfix) is one of the simplest systems you can work with.

References:

- Using Samba, A File & Print Server for Linux, Unix & Mac OS X, Gerald Carter, Jay Ts, Robert Eckstein, ISBN-10:978-0-596-00769-0
- Linux System Administration Recipes 1st Edition, by Kemp Juliet, Publisher: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG
- Linux: The Complete Reference, Sixth Edition, by Richard Pearson, Tata McGraw Hill Company Limited.
- **Www.samba.org**
- **www.redhat.com**
- **www.web.mit.edu**
- **wiki.dovcot.org**
- **www.plesk.com**

UNIT -8

Securing Server with iptables and Configuring Web Server

Securing Server with iptables:

What is a Firewall?

A firewall is a security device that monitors network traffic. It protects the internal network by filtering incoming and outgoing traffic based on a set of established rules. Setting up a firewall is the simplest way of adding a security layer between a system and malicious attacks.

How Does a Firewall Work?

A firewall is placed on the hardware or software level of a system to secure it from malicious traffic. Depending on the setup, it can protect a single machine or a whole network of computers. The device inspects incoming and outgoing traffic according to predefined rules.

Communicating over the Internet is conducted by requesting and transmitting data from a sender to a receiver. Since data cannot be sent as a whole, it is broken up into manageable data packets that make up the initially transmitted entity. The role of a firewall is to examine data packets traveling to and from the host.

What does a firewall inspect? Each data packet consists of a **header** (control information) and **payload** (the actual data). The header provides information about the sender and the receiver. Before the packet can enter the internal network through the defined port, it must pass through the firewall. This transfer depends on the information it carries and how it corresponds to the predefined rules.

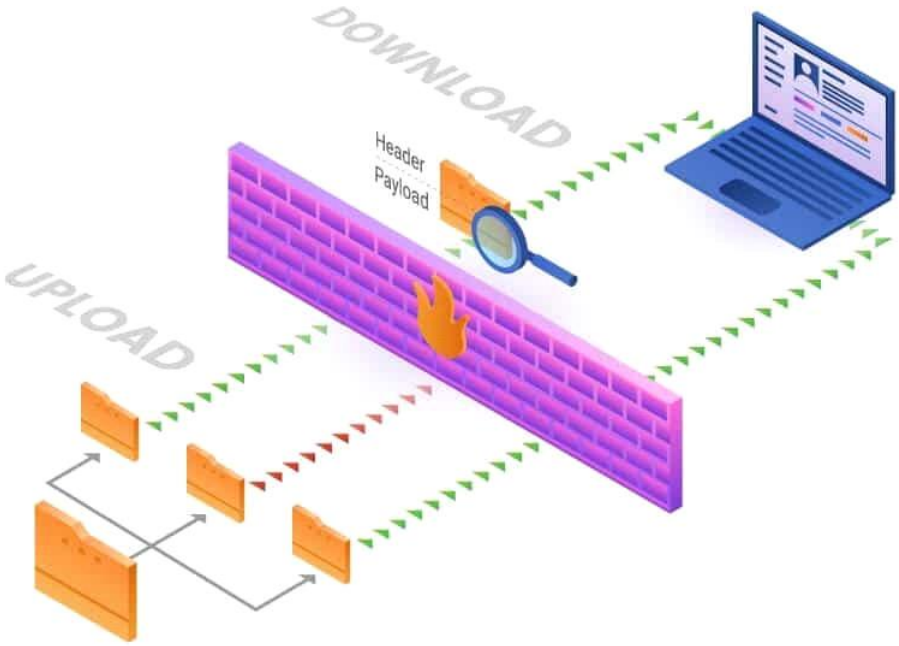


Fig 8.1 The firewall illustration

For example, the firewall can have a rule that excludes traffic coming from a specified IP address. If it receives data packets with that IP address in the header, the firewall denies access. Similarly, a firewall can deny access to anyone except the defined trusted sources. There are numerous ways to configure this security device. The extent to which it protects the system at hand depends on the type of firewall.

Types of Firewalls

Although they all serve to prevent unauthorized access, the operation methods and overall structure of firewalls can be quite diverse. According to their structure, there are three types of firewalls – software firewalls, hardware firewalls, or both. The remaining types of firewalls specified in this list are firewall techniques which can be set up as software or hardware.

1.Software Firewalls

A software firewall is installed on the host device. Accordingly, this type of firewall is also known as a Host Firewall. Since it is attached to a specific device, it has to utilize its resources to work. Therefore, it is inevitable for it to use up some of the system's RAM and CPU.

If there are multiple devices, you need to install the software on each device. Since it needs to be compatible with the host, it requires individual configuration for each. Hence, the main disadvantage is the time and knowledge needed to administrate and manage firewalls for each device.

On the other hand, the advantage of software firewalls is that they can distinguish between programs while filtering incoming and outgoing traffic. Hence, they can deny access to one program while allowing access to another

2.Hardware Firewalls

As the name suggests, hardware firewalls are security devices that represent a separate piece of hardware placed between an internal and external network (the Internet). This type is also known as an Appliance Firewall.

Unlike a software firewall, a hardware firewall has its resources and doesn't consume any CPU or RAM from the host devices. It is a physical appliance that serves as a gateway for traffic passing to and from an internal network.

They are used by medium and large organizations that have multiple computers working inside the same network. Utilizing hardware firewalls in such cases is more practical than installing individual software on each device. Configuring and managing a hardware firewall requires knowledge and skill, so make sure there is a skilled team to take on this responsibility.

Types of firewalls based on their method of operation

Packet-Filtering Firewalls

When it comes to types of firewalls based on their method of operation, the most basic type is the packet-filtering firewall. It serves as an inline security checkpoint attached to a router or switch. As the name suggests, it monitors network traffic by filtering incoming packets according to the information they carry.

As explained above, each data packet consists of a header and the data it transmits. This type of firewall decides whether a packet is allowed or denied access based on the header information. To do so, it inspects the protocol, source IP address, destination IP, source port, and destination port. Depending on how the numbers match the **access control list** (rules defining wanted/unwanted traffic), the packets are passed on or dropped.

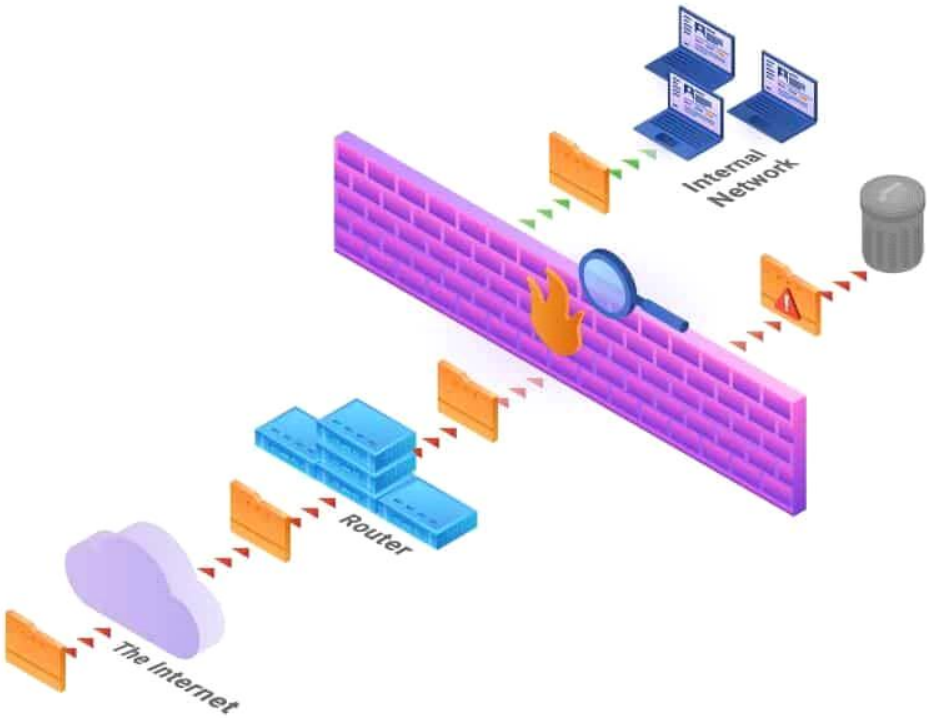


Fig. 8.2 Data transmition and firewall

If a data packet doesn't match all the required rules, it won't be allowed to reach the system.

A packet-filtering firewall is a fast solution that doesn't require a lot of resources. However, it isn't the safest. Although it inspects the header information, it doesn't check the data (payload) itself. Because malware can also be found in this section of the data packet, the packet-filtering firewall is not the best option for strong system security.

PACKET-FILTERING FIREWALLS			
Advantages	Disadvantages	Protection Level	Who is it for:
– Fast and efficient for filtering headers. – Don't use up a lot of resources. – Low cost.	– No payload check. – Vulnerable to IP spoofing. – Cannot filter application layer protocols. – No user	– Not very secure as they don't check the packet payload.	– A cost-efficient solution to protect devices within an internal network. – A means of isolating traffic internally between different

	authentication.		departments.

Circuit-Level Gateways

Circuit-level gateways are a type of firewall that work at the session layer of the OSI model, observing TCP (Transmission Control Protocol) connections and sessions. Their primary function is to ensure the established connections are safe.

In most cases, circuit-level firewalls are built into some type of software or an already existing firewall.

Like pocket-filtering firewalls, they don't inspect the actual data but rather the information about the transaction. Additionally, circuit-level gateways are practical, simple to set up, and don't require a separate proxy server.

Stateful Inspection Firewalls

A stateful inspection firewall keeps track of the state of a connection by monitoring the TCP 3-way handshake. This allows it to keep track of the entire connection – from start to end – permitting only expected return traffic inbound.

When starting a connection and requesting data, the stateful inspection builds a database (state table) and stores the connection information. In the state table, it notes the source IP, source port, destination IP, and destination port for each connection. Using the stateful inspection method, it dynamically creates firewall rules to allow anticipated traffic.

This type of firewall is used as additional security. It enforces more checks and is safer compared to stateless filters. However, unlike stateless/packet filtering, stateful firewalls inspect the actual data transmitted across multiple packets instead of just the headers. Because of this, they also require more system resources.

Proxy Firewalls:

A proxy firewall serves as an intermediate device between internal and external systems communicating over the Internet. It protects a network by forwarding requests from the original client and masking it as its own. Proxy means to *serve as a substitute* and, accordingly, that is the role it plays. It substitutes for the client that is sending the request.

When a client sends a request to access a web page, the message is intersected by the proxy server. The proxy forwards the message to the web server, pretending to be the client. Doing so hides the client's identification and geolocation, protecting it from any restrictions and potential attacks. The web server then responds and gives the proxy the requested information, which is passed on to the client.

Next-Generation Firewalls

The next-generation firewall is a security device that combines a number of functions of other firewalls. It incorporates packet, stateful, and deep packet inspection. Simply put, NGFW checks the actual payload of the packet instead of focusing solely on header information.

Unlike traditional firewalls, the next-gen firewall inspects the entire transaction of data, including the TCP handshakes, surface-level, and deep packet inspection.

Using NGFW is adequate protection from malware attacks, external threats, and intrusion. These devices are quite flexible, and there is no clear-cut definition of the functionalities they offer. Therefore, make sure to explore what each specific option provides.

Cloud Firewalls:

A cloud firewall or firewall-as-a-service (Faas) is a cloud solution for network protection. Like other cloud solutions, it is maintained and run on the Internet by third-party vendors.

Clients often utilize cloud firewalls as proxy servers, but the configuration can vary according to the demand. Their main advantage is scalability. They are independent of physical resources, which allows scaling the firewall capacity according to the traffic load.

Businesses use this solution to protect an internal network or other cloud infrastructures (Iaas/Paas).

CLOUD FIREWALLS			
Advantages	Disadvantages	Protection Level	Who is it for:
– Availability. – Scalability that offers increased bandwidth and new site protection. – No hardware required. – Cost-efficient in terms of managing and maintaining equipment.	– A wide range of prices depending on the services offered. – The risk of losing control over security assets. – Possible compatibility difficulties if migrating to a new cloud provider.	– Provide good protection in terms of high availability and having a professional staff taking care of the setup.	– A solution suitable for larger businesses that do not have an in-staff security team to maintain and manage the on-site security devices.

Setting Up a Firewall with system-config-firewall:

Since the 2.4 kernel, Linux has used iptables to configure firewall rules in the kernel. There are a number of tools that allow one to configure the firewall: iptables on the command-line, Shorewall, and a number of other GUI tools. On a Fedora system, the default firewall configuration tool is simply called Firewall Configuration, which can be found on the command-line by executing "system-config-firewall" or System | Administration | Firewall in the GNOME menus.

This GUI allows you to set which services are allowed to be accessed via the Internet using a very simple interface. It defines a number of trusted services pre-configured; to allow access, you simply need to check the box next to the entry. Each entry lists the service name, the port and protocol, and any additional iptables modules (conntrack helpers) it uses. So if you wanted to allow SSH access to the system, you would check off the box next to the SSH service as in **Figure**.

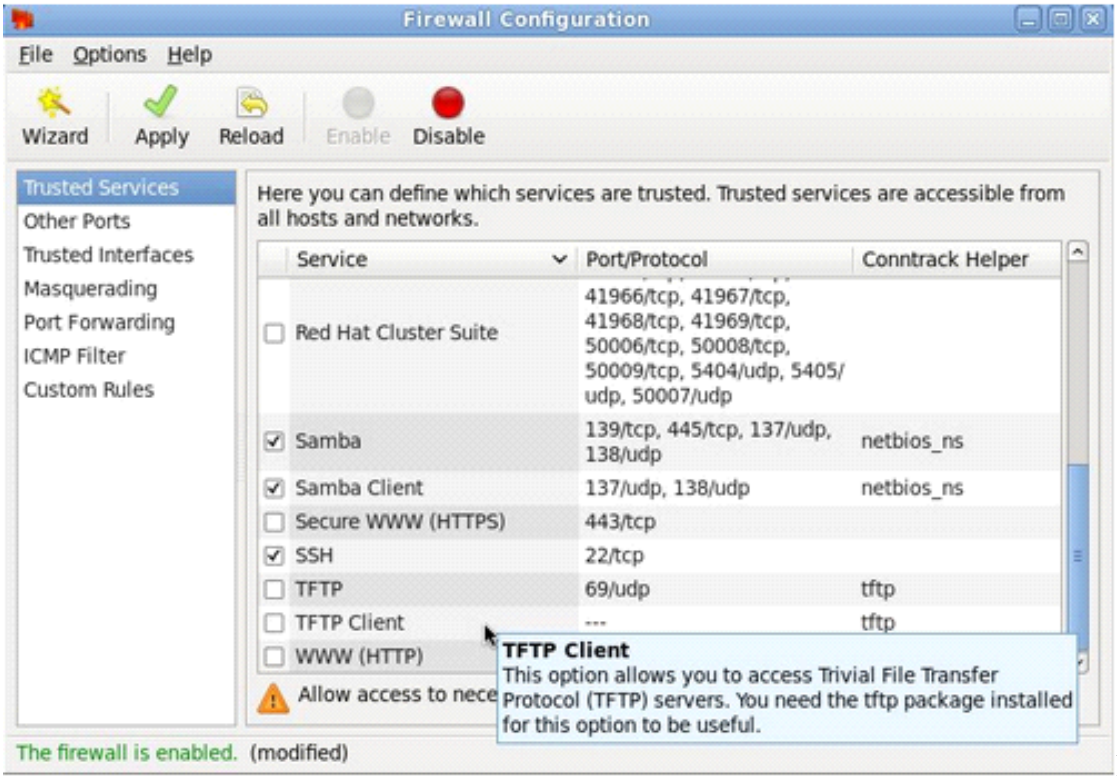


Fig. 8.3 Firewall configuration in Linux

You can move beyond simple service-level filtering, however. With the Trusted Interfaces section, you can define, on a multi-interface system, which interfaces are trusted. A trusted interface is one that does not have any firewall rules applied; for instance if *eth0* faced the Internet and *eth1* faced the local network, you might select that the eth1 interface is trusted. This would allow all connections coming in on the eth1 interface, while applying the firewall rules to all of the other interfaces.

The Other Ports section allows you to add new ports to filter that are not in the Trusted Services list. It pulls up a scrollable interface that lists the ports and protocols as defined in /etc/services, so all known ports and protocol types will be listed here. If there is a custom service you want that is not listed, select User Defined and provide the port and protocol manually.

With the Firewall Configuration GUI, you can also define masquerading, which allows you to use the system as a router; meaning you can use it as a gateway to forward connections from other local machines through it to the Internet. You can also define port forwarding; for instance, any incoming connections on port 22 would get forwarded to another defined host, great for allowing specific access to systems behind the firewall. You can define the incoming interface, protocol, and port to forward on, and then which IP address to forward to and an optional other port (i.e., forwarding connections to port 522 on the eth0 interface to port 22 on 192.168.1.2).

Finally, you can also change how the firewall will handle ICMP (Internet Control Message Protocol) packets. By default, all ICMP types are permitted, but here you can decide whether the system will respond to ping and other ICMP packets.

When you make changes to the firewall, use the Apply button to save them and the Reload button to refresh and activate the firewall rules. If you want to take a look at the actual iptables commands, the tool saves them to /etc/sysconfig/iptables which is used by the iptables-restore command to load the firewall rules. If you are familiar enough with iptables commands, you can edit this file directly rather than using the GUI.

On the command-line, use "service iptables restart" to reload the firewall, and "service iptables stop" to disable the firewall completely.

Iptables has a lot of different commands and can be used to create some very sophisticated firewall rules as tools like Shorewall prove. Shorewall, however, can be complicated to set up correctly, so while it is a good tool, it is really only useful for dedicated firewalls or servers. The Firewall Configuration GUI, on the other hand, is simple enough that anyone can use it to create customized firewalls for any Linux system, and powerful enough that you don't really need anything else.

Another way:

The GUI screen to control the firewall is available from the menu (System > Administration > Firewall) or can be started from the command line using the **system-config-firewall** command. If it is not already present, it can be installed using the following command.

```
# yum install system-config-firewall
```

Once started, the toolbar provides buttons to allow the firewall to be enabled/disabled. You can also configure basic trusted services, such as SSH, FTP and HTTP, by putting a tick in the appropriate checkbox and clicking the "Apply" button on the toolbar.

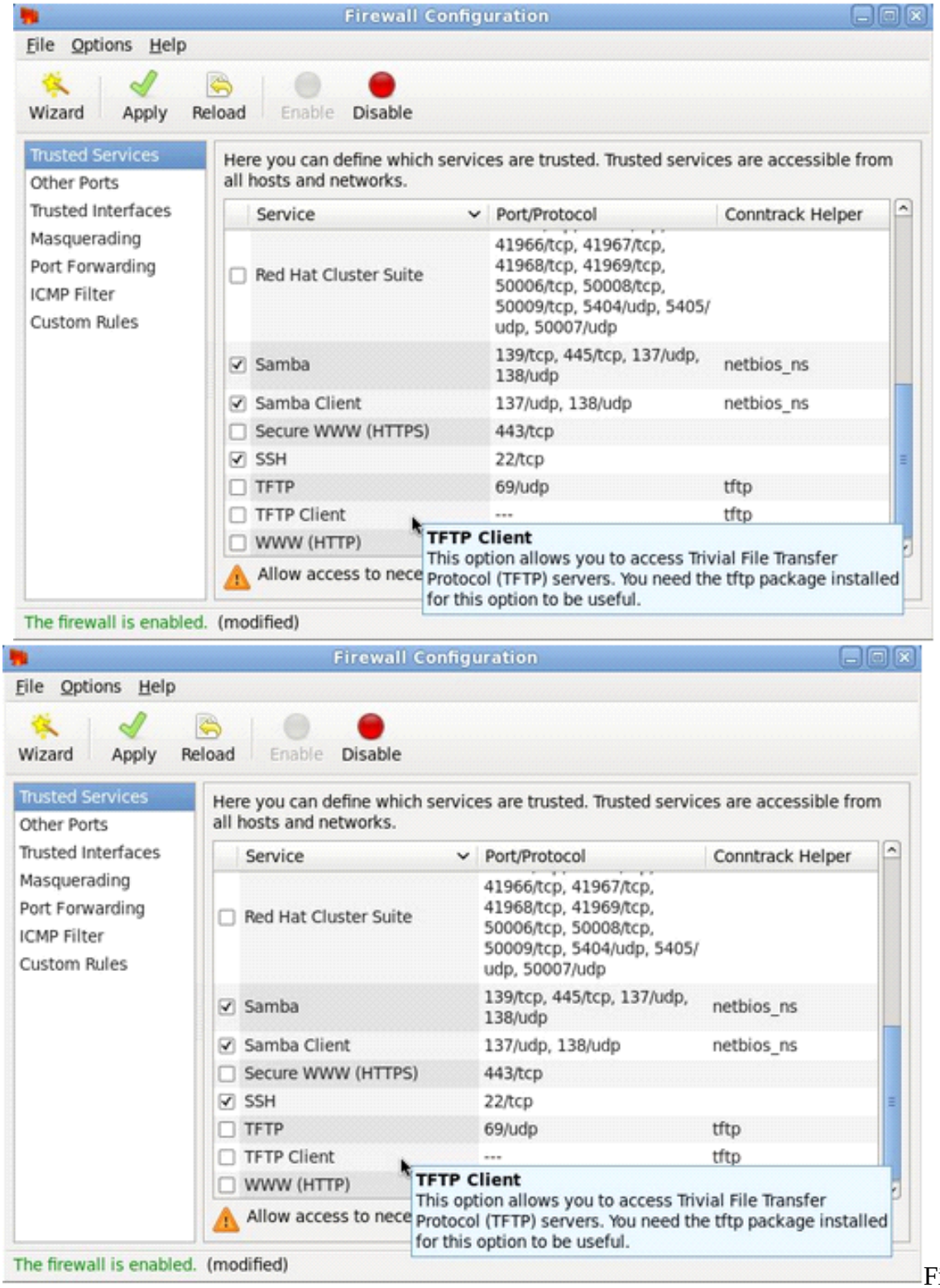


Fig. 8.4 Firewall configuration in Linux

The "Other Ports" section allows you to open ports that are not covered in the "Trusted Services" section.

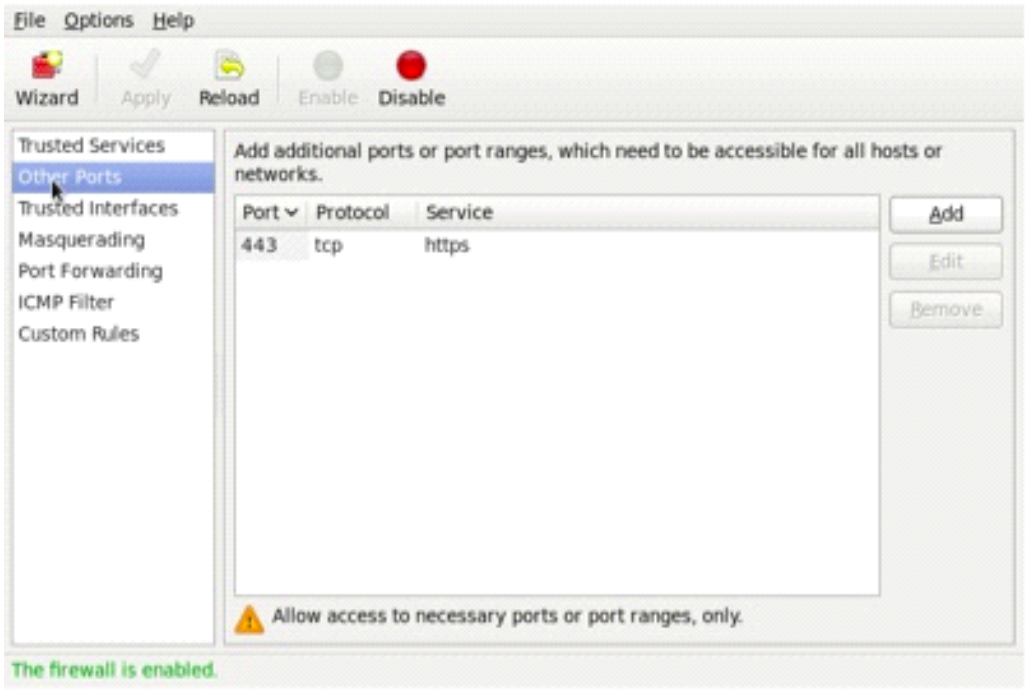


Fig. 8.5 Other ports

Setting Up a Firewall with iptables:

Most installations will include the firewall functionality. If you need to manually install it, the following commands will install the IP4 and IP6 firewall functionality. In this article we will only consider the IP4 settings.

```
# yum install iptables

# yum install iptables-ipv6
```

Make sure the service is started and will auto-start on reboot.

```
# service iptables start
# chkconfig --level 345 iptables on
```

You can check the current status of the service using the following command.

```
# service iptables status
Table: filter
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
2    ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
3    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
4    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:21
5    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
6    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:80
7    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:443
8    REJECT     all  --  0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination
1    REJECT     all  --  0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
```

Fig. 8.6 The iptables command output

To disable the firewall, run the following commands.

```
# service iptables stop
# chkconfig iptables off
```

System-config-firewall-tui:

The TUI utility is similar to the GUI utility shown above, but it feels incredibly clumsy in comparison. If it is not already present, it can be installed using the following command.

```
# yum install system-config-firewall-tui
```

Running the system-config-firewall-tui command from the command line produces the top-level screen, allowing you to enable/disable the firewall. Use the space bar to toggle the setting, the tab key to navigate between buttons and the return key to click them.

Fig. 8.7 Firewall sytem configuration

To alter the Trusted Services, tab to the "Customize" button and press the return key. Amend the list using the arrow and space keys.

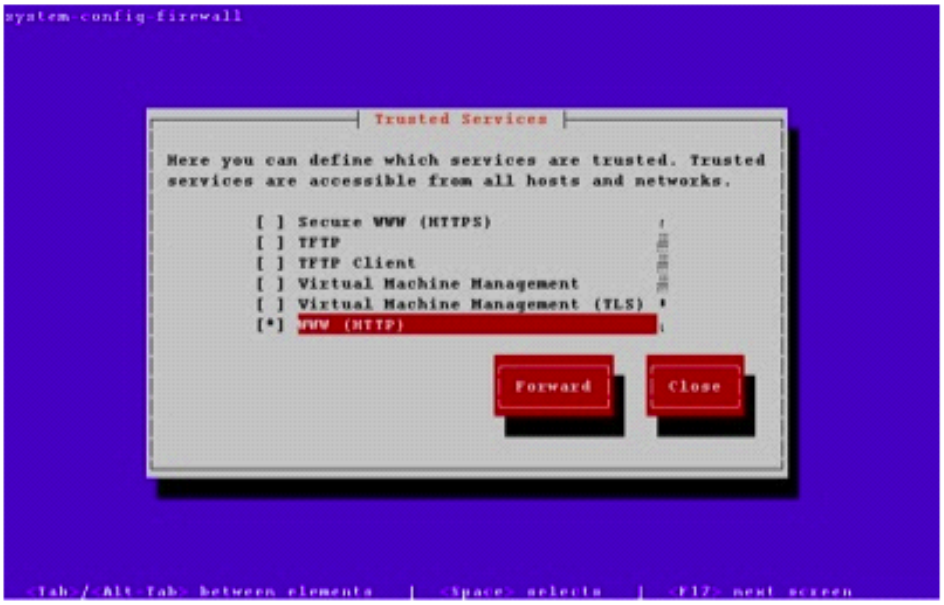


Fig. 8.8 Firewall customization

You can close out of the customization section at any point. The other sections of the GUI tool are available by clicking the "Forward" button on each successive screen.

Iptables:

In addition to the GUI and TUI interfaces, the firewall rules can be amended directly using the iptables command.

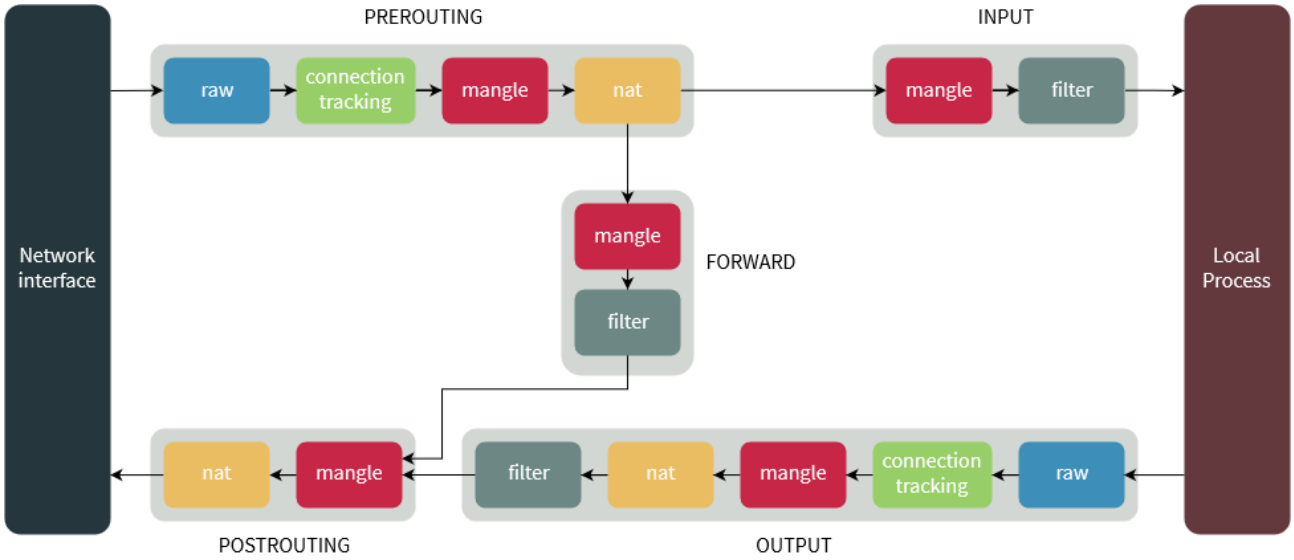


Fig. 8.9 The iptable routing

The firewall consists of chains of rules that determine what action should be taken for packets processed by the system. By default, there are three chains defined:

•INPUT : Used to check all packets coming into the system.

•OUPUT : Used to check all packets leaving the system.

•FORWARD : Used to check all packets being routed by the system. Unless you are using your server as a router, this chain is unnecessary.

Each chain can contain multiple explicit rules that are checked in order. If a rule matches, the associated action (ACCEPT and DROP being the most common) is taken. If no specific rule is found, the default policy is used to determine the action to take.

Since the default policy is a catch-all, one of two basic methods can be chosen for each chain.

•Set the default policy to ACCEPT and explicitly DROP things you don't want.

•Set the default policy to DROP and explicitly ACCEPT things you do want.

The safest option is to set the default policy to DROP for the INPUT and FORWARD chains, so it is perhaps a little surprising that the GUI and TUI tools set the default policies to ACCEPT, then use an explicit REJECT as the last rule in these chains.

```
# iptables -L -v --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1       11   812 ACCEPT     all  --  any    any     anywhere             anywhere            state RELATED,ESTABLISHED
2        0     0 ACCEPT     icmp --  any    any     anywhere             anywhere
3        0     0 ACCEPT     all  --  lo     any     anywhere             anywhere
4        1   100 ACCEPT     tcp  --  any    any     anywhere             anywhere            state NEW tcp dpt:ssh
5        0     0 REJECT     all  --  any    any     anywhere             anywhere            reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 REJECT     all  --  any    any     anywhere             anywhere            reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 6 packets, 744 bytes)
num   pkts bytes target     prot opt in     out     source               destination
#
```

Fig. 8.10 The iptables options

The default policy for a chain is set using the "-P" flag. In the following example, assuming no specific rules were present, all communication to and from the server would be prevented.

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT DROP
```

Warning: If you are administering the firewall via SSH, having a default INPUT policy of DROP will cut your session off if you get rid of the explicit rules that accept SSH access. As a result, it makes sense to start any administration by setting the default policies to ACCEPT and only switch them back to DROP once the chains have been built to your satisfaction. The following example temporarily sets the default policies to ACCEPT.

```
# iptables -P INPUT ACCEPT
# iptables -P FORWARD ACCEPT
# iptables -P OUTPUT ACCEPT
```

The next thing we want to do if flush any existing rules, leaving just the default policies. This is done using the "-F" flag.

```
# iptables -F
```

Now we need to define specific rules for the type of access we want the server to have. Focusing on the INPUT chain, we can grant access to packets in a number of ways.

Accept packets from specific interfaces.
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -i eth0 -j ACCEPT

Accept packets from specific hosts.
iptables -A INPUT -s 192.168.122.1 -j ACCEPT

```
# Accept packets from a specific subnet.
iptables -A INPUT -s 192.168.122.0/24 -j ACCEPT
iptables -A INPUT -s 192.168.122.0/255.255.255.0 -j ACCEPT

# Accept packets from specific host, checking the MAC address.
iptables -A INPUT -s 192.168.122.1 -m mac --mac-source 52:54:00:91:6A:B3 -j ACCEPT

# Accept packets from a specific port.
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Accept packets from connections in a specific state.
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Combinations of all of the above.
iptables -A INPUT -i eth0 -p tcp -s 192.168.122.0/24 --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
```

Fig. 8.11 The iptables with multiple options

Once the explicit rules are defined, we need to set the real default policies.

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

Rule and policy definitions take effect immediately.To make sure they persists beyond reboot the current configuration must be saved to the "/etc/sysconfig/iptables" file using the following command.

service iptables save

If you are using Fedora, you may need to use the following command instead.

iptables-save > /etc/sysconfig/iptables

As you can imagine, even in a simple configuration this process can get a bit long-winded, so it makes sense to combine all the elements of the firewall definition into a single file so it can be amended and run repeatedly. Create a file called "/root/firewall.sh" with the following contents. Think of this as your starting point for each server.

```
#!/bin/bash

# Set the default policies to allow everything while we set up new rules.
# Prevents cutting yourself off when running from remote SSH.
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

# Flush any existing rules, leaving just the defaults
iptables -F

# Open port 21 for incoming FTP requests.
iptables -A INPUT -p tcp --dport 21 -j ACCEPT

# Open port 22 for incoming SSH connections.
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# Limit to eth0 from a specific IP subnet if required.
#iptables -A INPUT -i eth0 -p tcp -s 192.168.122.0/24 --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

# Open port 80 for incoming HTTP requests.
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
# Open port 443 for incoming HTTPS requests. (uncomment if required)
#iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# *** Put any additions to the INPUT chain here.
#
# *** End of additions to INPUT chain.
# Accept any localhost (loopback) calls.
iptables -A INPUT -i lo -j ACCEPT

# Allow any existing connection to remain.
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Reset the default policies to stop all incoming and forward requests.
iptables -P INPUT DROP
iptables -P FORWARD DROP
# Accept any outbound requests from this server.
iptables -P OUTPUT ACCEPT

# Save the settings.
service iptables save
# Use the following command in Fedora
#iptables-save > /etc/sysconfig/iptables

# Display the settings.
iptables -L -v --line-numbers
```

Fig. 8.12 The firewall.sh file

Make the file executable.

chmod u+x /root/firewall.sh

Run the file to set the required firewall rules.

```
# /root/firewall.sh
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
Chain INPUT (policy DROP 4457 packets, 522K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere            tcp dpt:ftp
2     1394  116K ACCEPT     tcp  --  any    any     anywhere             anywhere            tcp dpt:ssh
3        0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere            tcp dpt:http
4        8   400 ACCEPT     all  --  lo     any     anywhere             anywhere
5    96358  138M ACCEPT     all  --  any    any     anywhere             anywhere            state RELATED,ESTABLISHE

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 50890 packets, 2890K bytes)
num   pkts bytes target     prot opt in     out     source               destination
#
```

Fig. 8.13 The rules in the firewall.sh

The iptables command also allows you to insert (-I), delete (-D) and replace (-R) rules, but if you work using a file as described above, you never need to use these variations.

Quick Database Setup

If you are using the server as an Oracle database server, you will probably want to make sure the SSH and Oracle listener ports are accessible. You could lock these down to specific source IP addresses, but for a quick setup, you could just do the following, where "1521" is the port used for the listener.

service iptables start

chkconfig iptables on
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 1521 -j ACCEPT
service iptables save
service iptables status

Simple Firewall / Router Combination

As an example, we place a box between two networks, functioning as a router and a firewall. We'll just add some basic.

NETWORK A --------------------[router / firewall]---------------------- NETWORK B

192.168.1.0/24 192.168.1.1 192.168.2.1 192.168.2.0/24

Packets from A to B will pass the router, in an appearently transparent LAN. Considering there is no link with the internet, and all clients are 'trusted' desktop PC's, we barely need the firewall functionality and in this topology. The router configuration is very simple.

We use the following command to start routing:

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

Then, one day, we decide to strengthen the security between the two networks. We know Network B has some servers that host some websites that clients in Network A must be able to visit. We decide to implement the following:

- All packets between Network A and Network B must be dropped by default.
- Network A is allowed to visit websites hosted on webservers in Network B.

We implement the following in iptables on the router:

```
# iptables --policy FORWARD DROP
# iptables --append FORWARD --source 192.168.1.0/24 --destination 192.168.2.0/24 --match state --state NEW,ESTABLISHED --protocol tcp --destination-port 80 -j ACCEPT
# iptables --append FORWARD --source 192.168.2.0/24 --destination 192.168.1.0/24 --match state --state ESTABLISHED --protocol tcp --source-port 80 -j ACCEPT
```

First, we set the default action for everything that is forwarded by the router, to DROP. This disables all traffic between the two networks that is not explicitly allowed.

Second, we ACCEPT traffic from Network A to Network B, if the destination port is port 80 (HTTP), and the protocol is tcp. We allow all NEW and ESTABLISHED connections.

Third, we ACCEPT the responses which of course start at Network B and travel to the client in Network A. This is an already established connection, and we know the source port (the socket which the server uses to respond to the client), is port 80 (HTTP).

Here's another example, to show you how clients in Network A can also use the mail-, web-, and POP/IMAP- servers in Network B.

```
# iptables --policy FORWARD DROP
# iptables --append FORWARD --source 192.168.1.0/24 --destination 192.168.2.0/24 --match state --state NEW,ESTABLISHED --protocol tcp \
        --match multiport --destination-ports 25,80,110,143 -j ACCEPT
# iptables --append FORWARD --source 192.168.2.0/24 --destination 192.168.1.0/24 --match state --state ESTABLISHED \
        --match multiport --source-ports 25,80,110,143 -j ACCEPT
```

You can of course extend these commands to match your requirements. Notice the above examples only limit the network traffic between the two networks, and not traffic to, or from, the firewall itself.

NAT (Network Address Translation)

In computer networking, the process of network address translation (NAT, also known as network masquerading or IP-masquerading) involves re-writing the source and/or destination addresses of IP packets as they pass through a router or firewall. Most systems using NAT do so in order to enable multiple hosts on a private network to access the Internet using a single public IP address. According to specifications, routers should not act in this way, but many network administrators find NAT a convenient technique and use it widely. Nonetheless, NAT can introduce complications in communication between hosts.

Different types of NAT

- **Full cone NAT** is NAT where all requests from the same internal IP address and port are mapped to the same external IP address and port. Furthermore, any external host can send a packet to the internalhost, by sending a packet to the mapped external address. It is also known as "**one-to-one NAT"**.

- A **restricted cone NAT** is one where all requests from the same internal IP address and port are mapped to the same external IP address and port. Unlike a full cone NAT, an external host (with IP address X) can send a packet to the internal host only if the internal host had previously sent a packet to IP address X.

- A **port restricted cone NAT** is like a restricted cone NAT, but the restriction includes port numbers. Specifically, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P.

- A **symmetric NAT** is a NAT where all requests from the same internal IP address and port to a specific destination IP address and port are mapped to the same external source IP address and port. If the same internal host sends a packet with the same source address and port to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet can send a UDP packet back to the internal host.

Example Scenario: SOHO

In a SOHO (Small Office, Home Office) environment, you would typically have a single public IP address. We'll first show you how to setup basic routing, before actually running a secure firewall.

```
                        eth1            eth0
SOHO Network ----------------------[ router / firewall ]---------------------- Internet
192.168.1.0/24          192.168.1.1     public-ip-address       0.0.0.0/0
```

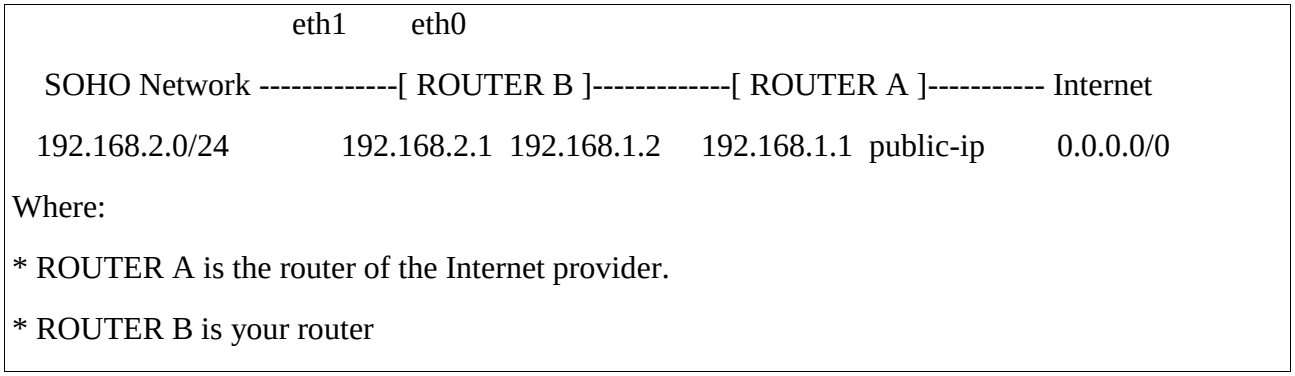
The above network topology requires the router to use one public IP address for packets from the SOHO Network to the Internet. Also, the router should accept inbound packets that are related to connections initiated from the SOHO Network (responses etc.). Notice that there is no modem and no provider supplied router between our router and the Internet. For the very basic router setup using iptables, you would use:

```
# iptables --policy INPUT DROP

# iptables --policy FORWARD DROP

# iptables --policy OUTPUT DROP

# iptables --append INPUT --in-interface eth1 --source 192.168.1.0/24 --match state --state NEW,ESTABLISHED --jump ACCEPT

# iptables --append OUTPUT --out-interface eth1 --destination 192.168.1.0/24 --match state --state NEW,ESTABLISHED --jump ACCEPT

# iptables --append FORWARD --in-interface eth1 --source 192.168.1.0/24 --destination 0.0.0.0/0 --match state --state NEW,ESTABLISHED --jump ACCEPT

# iptables --append FORWARD --in-interface eth0 --destination 192.168.1.0/24 --match state --state ESTABLISHED --jump ACCEPT

# iptables --table nat --append POSTROUTING --out-interface eth0 --jump MASQUERADE
```

The router now forwards packets between the two networks, masquerades the outgoing packets from the SOHO Network (so responses at least come back to the router again), and enables management from the SOHO Network as well. Notice that we allow NEW connections from the LAN to the Internet, but not the other way around. Also, because we use MASQUERADE, our router/firewall will only have to FORWARD traffic that comes back from the Internet as a reponse, because, as you may remember from the Packet Processing Overview earlier in this document, as a reply comes back from the Internet, our router will use the nat table to match the existing connection, apply PREROUTING destination NAT as the packet comes in, and hit the FORWARD chain with a new destination IP address.

A SOHO Network with a Seperate Router and Firewall:

The previous scenario suggests there is one single network device, apart from switches, hubs and modems, between the SOHO Network and the Internet. In most topologies, this is not the case. Many SOHO Networks have a router from the Internet provider to connect to the Internet. If that is the case, the network topology changes:



In this case, configure the following:

- Configure ROUTER A to have a static route to 192.168.2.0/24 via 192.168.1.2
- Configure ROUTER A to have 192.168.1.2 as a default (virtual) server, using NAT
- Configure ROUTER B to have a default route to 0.0.0.0/0 via 192.168.1.1

ROUTER A will now NAT all incoming packets to 192.168.1.2, and MASQUERADE all outgoing packets with the public IP. The incoming packets will end up with ROUTER B, which is also the firewall.

Suppose there is a webserver in the SOHO Network, which must be available to the public (the Internet) as well as the clients on the LAN. This means you need to forward all requests to ROUTER B, port 80 to the webserver (suppose this webserver is at 192.168.2.20):

```
# iptables -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.2.20<br />
```

The same goes for other services that are hosted on the LAN.

Configuring a Web Server:

Introducing Apache:

The Apache HTTP server is the most widely-used web server in the world. It provides many powerful features, including dynamically loadable modules, robust media support, and extensive integration with other popular software.

Configuring Apache:

Step 1 — Installing Apache

Apache is available within Ubuntu's default software repositories, so you can install it using conventional package management tools.

Update your local package index:

```
$ sudo apt update
```

Install the apache2 package:

```
$ sudo apt install apache2
```

Step 2 — Adjusting the Firewall

```
$ sudo ufw app list
```

```
Output

Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

Let's enable the most restrictive profile that will still allow the traffic you've configured, permitting traffic on port 80 (normal, unencrypted web traffic):

```
$ sudo ufw allow 'Apache'
```

Verify the change:

```
$ sudo ufw status
```

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Apache                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

Apache (v6)	ALLOW	Anywhere (v6)

Step 3 — Checking your Web Server

Check with the systemd init system to make sure the service is running by typing:

```
Output
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Tue 2018-04-24 20:14:39 UTC; 9min ago
 Main PID: 2583 (apache2)
    Tasks: 55 (limit: 1153)
   CGroup: /system.slice/apache2.service
           ├─2583 /usr/sbin/apache2 -k start
           ├─2585 /usr/sbin/apache2 -k start
           └─2586 /usr/sbin/apache2 -k start
```

$ sudo systemctl status apache2

Access the default Apache landing page to confirm that the software is running properly through your IP address:

http://your_server_ip e.g. http://127.0.0.1

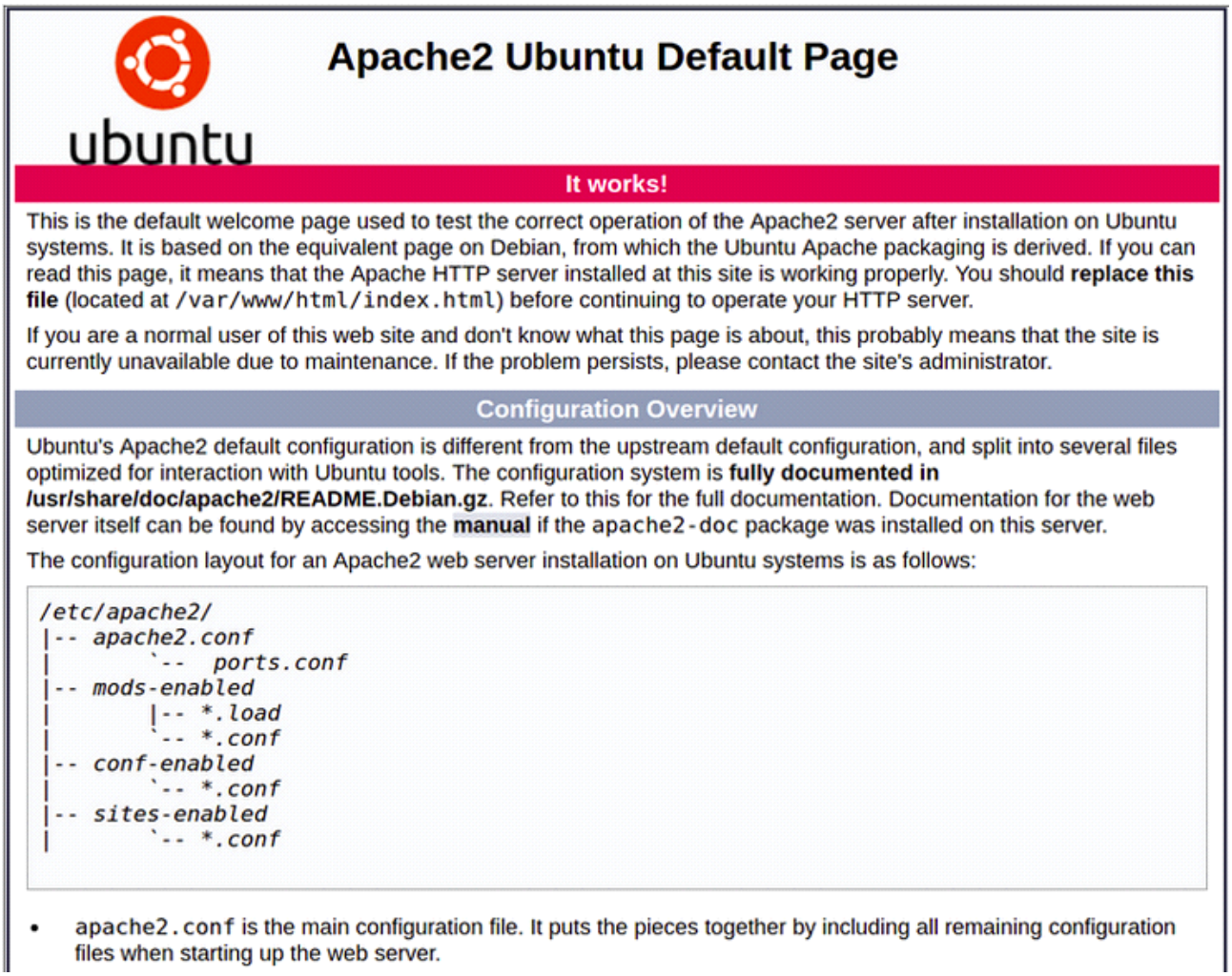


Fig. 8.14 The Apache front page

Step 4 — Setting Up Virtual Hosts (Recommended)

When using the Apache web server, you can use *virtual hosts* (similar to server blocks in Nginx) to encapsulate configuration details and host more than one domain from a single server. We will set up a domain called your_domain, but you should replace this with your own domain name.

Create the directory for your_domain:

$ sudo mkdir /var/www/your_domain

Assign ownership of the directory:

$ sudo chown -R $USER:$USER /var/www/your_domain

The permissions of your web roots should be correct if you haven't modified your unmask value, but you can make sure by typing:

$ sudo chmod -R 755 /var/www/your_domain

Create a sample index.html page using nano or your favorite editor:

$ nano /var/www/your_domain/index.html

Inside, add the following sample HTML:

/var/www/your_domain/index.html

```
<html>
    <head>
        <title>Welcome to Your_domain!</title>
    </head>
    <body>
        <h1>Success!  The your_domain virtual host is working!</h1>
    </body>
</html>
```

Fig. 8.15 Apache index.html file

Save and close the file when you are finished.

Make a new virtual host file at /etc/apache2/sites-available/your_domain.conf:

$ sudo nano /etc/apache2/sites-available/your_domain.conf

Paste in the following configuration block, updated for our new directory and domain name:

/etc/apache2/sites-available/your_domain.conf

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName your_domain
    ServerAlias your_domain
    DocumentRoot /var/www/your_domain
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Fig. 8.16 Apache your_domain.conf file

Save and close the file when you are finished.

Enable the file with a2ensite:

$ sudo a2ensite your_domain.conf

Disable the default site defined in 000-default.conf:

$ sudo a2dissite 000-default.conf

Test for configuration errors:

$ sudo apache2ctl configtest

You should see the following output:

```
Output
Syntax OK
```

Restart Apache to implement your changes:

$ sudo systemctl restart apache2

Apache should now be serving your domain name. You can test this by navigating to http://your_domain, where you should see something like this:

Implementing SSI:

What is SSI?

SSI stands for Server Side Includes. As the name suggests, they are simple server side scripts that are typically used as directives inside html comments.

Where to use SSI? There are several ways to SSI. The two most common reason to use SSI are to serve a dynamic content on your web page, and to reuse a code snippet as shown below.

1. Serve Dynamically Generated Content

For example, to display current time on your html page, you can use server side includes. You don’t need to use any other special server side scripting languages for it.

The following html code snippet shows this example. The line highlighted in bold is an SSI script.

```
<html>
 <head>
  <title>thegeekstuff.com</title>
 </head>
 <body>
  <p>
   Today is <!--#echo var="DATE_LOCAL" -->
  </p>
 </body>
</html>
```

Fig. 8.17 The SSI Script

2. Reuse the Same HTML Script

You can also use SSI to reuse a html snippet on multiple pages. This is very helpful to reuse header and footer information of a site on different pages.

The following is a sample header.html file that can be reused.

```
<header>
 <h1> The Geek Stuff</h1>
 <img  href="./images/logo.png"  alt="Logo" />
</header>
```

Fig. 8.18 header.html file

The following is a sample footer.html file that can be reused.

```
<footer>
 <span> The Geek Stuff, Los Angeles, USA <span>
 <ul>
  <li> <a href="aboutus.html">About Us</a> </li>
  <li> <a href="contactus.html">Contact Us</a> </li>
  <li> <a href="ourworks.html">Portfolio</a> </li>
 </ul>
</footer>
```

Fig. 8.19 footer.html file

Now, when it is time to reuse the above two files (header and footer), we simple include them on any other html page using the #include SSI as shown below.

This is the index.html, which includes both header and footer using server side includes.

```
<html>
 <head>
  <title>The Geek Stuff</title>
 </head>
 <body>
  <!--#include virtual="header.html" -->
   <div>
   <!-- Page content goes here  -->
   </div>
  <!--#include virtual="footer.html" -->
 </body>
</html>
```

Fig. 8.20 The index.html file

Similar to including a html page using SSI, you can also include the output of cgi script to the html using the following line:

```
<!--#include virtual="/cgi-bin/sometask.pl" -->
```

3. Setup SSI in .htaccess File

We can instruct the webserver to interpret Server Side Includes either using .htaccess or modifying the web-server config file directly.

Create .htaccess file in your web root and add the following lines of code:

```
AddType text/html .html
AddHandler server-parsed .html
Options Indexes FollowSymLinks Includes
```

The above lines instruct the web server to parse the .html extension for the server side includes present in it.

We can also instruct the server to parse the file with custom extensions as well. For example, we can use the following lines for parsing the ".shtml" file extensions.

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Similarly for parsing the cgi script we can add following lines:

```
AddType application/x-httpd-cgi .cgi
```

4. Modify Apache httpd.conf File

On Apache web server, the following directive lines should be present in httpd.conf file for SSI

```
Options +Includes
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

The first line tells Apache to allow the file to be parsed for SSI. The other lines tells the extension of the file to be parsed.

Enable CGI Module in Apache

To enable CGI in your Apache server. you need to Load module file mod_cgi.so or mod_cgid.so in your Apache configuration file.

The CentOS, Red Hat, Fedora and other rpm based distributions edit /etc/httpd/conf.modules.d/XX-cgi.conf configuration file and make sure below showing lines are not commented.

```
<IfModule mpm_worker_module>
   LoadModule cgid_module modules/mod_cgid.so
</IfModule>
<IfModule mpm_event_module>
   LoadModule cgid_module modules/mod_cgid.so
</IfModule>
<IfModule mpm_prefork_module>
   LoadModule cgi_module modules/mod_cgi.so
</IfModule>
```

Ubuntu, Debian, LinuxMint and other Debian derivatives use the following command to enable CGI module. This command creates a soft link of the module configuration file to /etc/apache2/mod-enabled/ directory.

$ sudo a2enmod cgi

After enabling CGI modules in Apache configuration you need to restart Apache service on your system for changes take effect.

Installing PHP

PHP is the component of your setup that will process code to display dynamic content. It can run scripts, connect to your MySQL databases to get information, and hand the processed content over to your web server to display.

Once again, leverage the apt system to install PHP. In addition, include some helper packages this time so that PHP code can run under the Apache server and talk to your MySQL database:

$ sudo apt install php libapache2-mod-php php-mysql

This should install PHP without any problems. We'll test this in a moment.

In most cases, you will want to modify the way that Apache serves files when a directory is requested. Currently, if a user requests a directory from the server, Apache will first look for a file called index.html. We want to tell the web server to prefer PHP files over others, so make Apache look for an index.php file first.

To do this, type this command to open the dir.conf file in a text editor with root privileges:

$ sudo nano /etc/apache2/mods-enabled/dir.conf

It will look like this:

/etc/apache2/mods-enabled/dir.conf

```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>
```

Move the PHP index file (highlighted above) to the first position after the DirectoryIndex specification, like this:

/etc/apache2/mods-enabled/dir.conf

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

When you are finished, save and close the file by pressing CTRL+X. Confirm the save by typing Y and then hit ENTER to verify the file save location.

After this, restart the Apache web server in order for your changes to be recognized. Do this by typing this:

$ sudo systemctl restart apache2

You can also check on the status of the apache2 service using systemctl:

$ sudo systemctl status apache2

Sample Output

```
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Tue 2018-04-23 14:28:43 EDT; 45s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 13581 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)
  Process: 13605 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)
    Tasks: 6 (limit: 512)
   CGroup: /system.slice/apache2.service
           ├─13623 /usr/sbin/apache2 -k start
           ├─13626 /usr/sbin/apache2 -k start
           ├─13627 /usr/sbin/apache2 -k start
           ├─13628 /usr/sbin/apache2 -k start
           ├─13629 /usr/sbin/apache2 -k start
           └─13630 /usr/sbin/apache2 -k start
```

Press Q to exit this status output.

Enabling SSL on your web server

The SSL protocol is a standard security technology used to establish an encrypted link between a web server and a web client. SSL facilitates secure network communication by identifying and authenticating the server as well as ensuring the privacy and integrity of all transmitted data. Since SSL prevents eavesdropping on or tampering with information

sent over the network, it should be used with any login or authentication mechanism and on any network where communication contains confidential or proprietary information.

The use of SSL ensures that names, passwords, and other sensitive information cannot be deciphered as they are sent between the Web Adaptor and the server. When you use SSL, you connect to your web pages and resources using the HTTPS protocol instead of HTTP.

In order to use SSL, you need to obtain an SSL certificate and bind it to the website that hosts the Web Adaptor. Each web server has its own procedure for loading a certificate and binding it to a website.

Creating an SSL certificate

To be able to create an SSL connection between the Web Adaptor and your server, the web server requires an SSL certificate. An SSL certificate is a digital file that contains information about the identity of the web server. It also contains the encryption technique to use when establishing a secure channel between the web server and ArcGIS Server. An SSL certificate must be created by the owner of the website and digitally signed. There are three types of certificates, CA-signed, domain, and self-signed, which are explained below.

CA-signed certificates

Certificate authority (CA) signed certificates should be used for production systems, particularly if your deployment of ArcGIS Server is going to be accessed from users outside your organization. For example, if your server is not behind your firewall and accessible over the Internet, using a CA-signed certificate assures clients from outside your organization that the identity of the website has been verified.

In addition to being signed by the owner of the website, an SSL certificate may be signed by an independent CA. A CA is usually a trusted third party that can attest to the authenticity of a website. If a website is trustworthy, the CA adds its own digital signature to that website's self-signed SSL certificate. This assures web clients that the website's identity has been verified.

When using an SSL certificate issued by a well-known CA, secure communication between the server and the web client occurs automatically with no special action required by the user. There is no unexpected behavior or warning message displayed in the web browser, since the website has been verified by the CA.

Domain certificates

If your server is located behind your firewall and using a CA-signed certificate is not possible, using a domain certificate is an acceptable solution. A domain certificate is an internal certificate signed by your organization's certificate authority. Using a domain certificate helps you reduce the cost of issuing certificates and eases certificate deployment, since certificates can be generated quickly within your organization for trusted internal use.

Users within your domain will not experience any of the unexpected behavior or warning messages normally associated with a self-signed certificate, since the website has been verified by the domain certificate. However, domain certificates are not validated by an external CA, which means users visiting your site from outside your domain will not be able verify that your certificate really represents the party it claims to represent. External users will see browser warnings about the site being untrusted which may lead them to think that they are actually communicating with a malicious party and be turned away from your site.

Creating a domain certificate in IIS

In IIS Manager, do the following to create a domain certificate:

•In the Connections pane, select your server in the tree view and double-click Server Certificates.

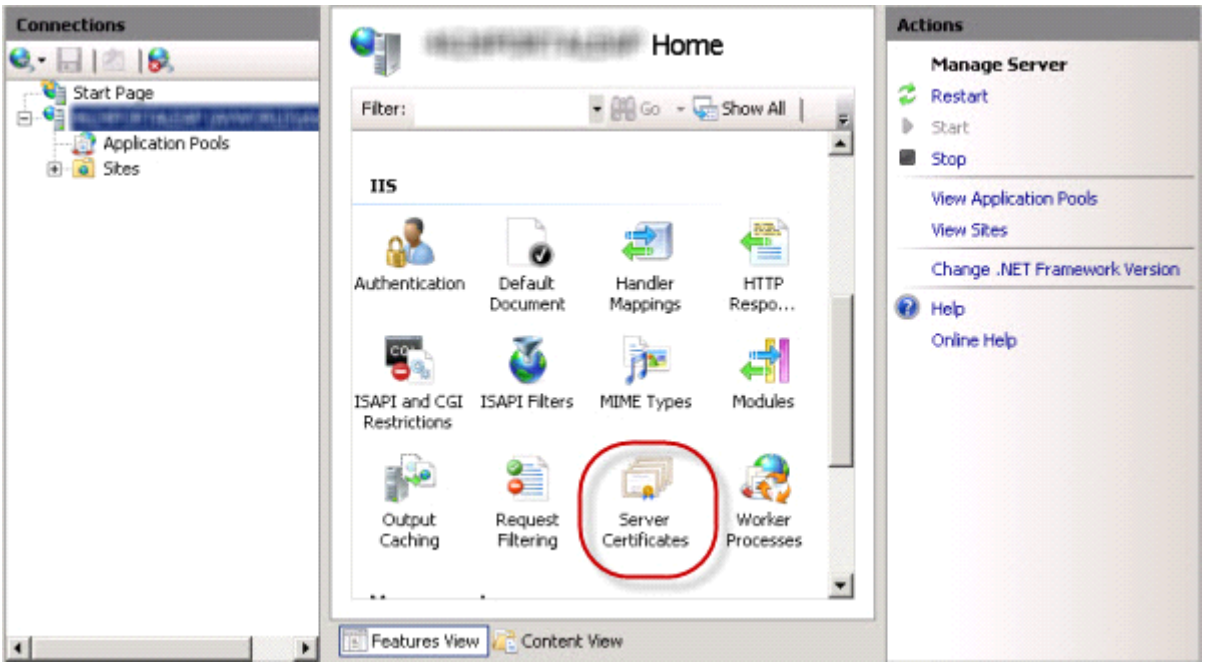


Fig. 8.21 The Server certificates location

•In the Actions pane, click Create Domain Certificate.

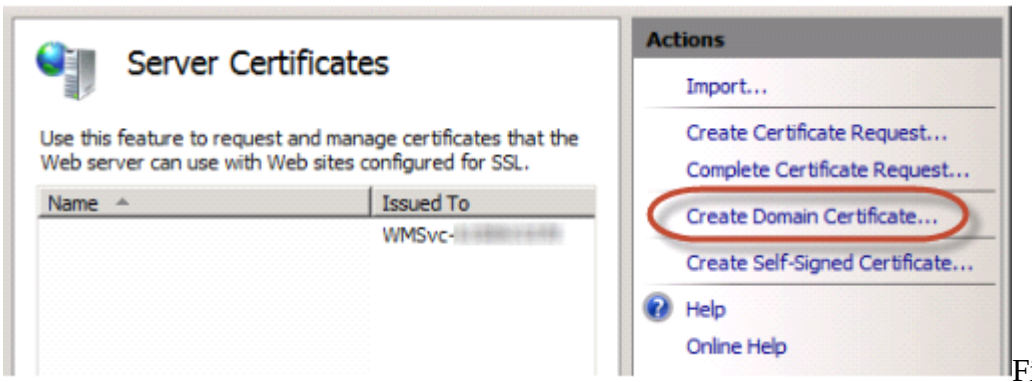


Fig. 8.22 Create domain certificate

•**In the Distinguished Name Properties dialog box, enter the required information for the certificate:**

•For the Common name, you must enter the fully qualified domain name of the machine, for example, gisserver.domain.com.

•For the other properties, enter the information specific for your organization and location.

•Click **Next**.

•In the Online Certification Authority dialog box, click Select and choose the certification authority within your domain that will sign the certificate. If this option is unavailable, enter your domain certification authority in the Specify Online Certification Authority field, for example, City Of Redlands Enterprise Root\REDCASRV.empty.local. If you need help with this step, consult your system administrator.

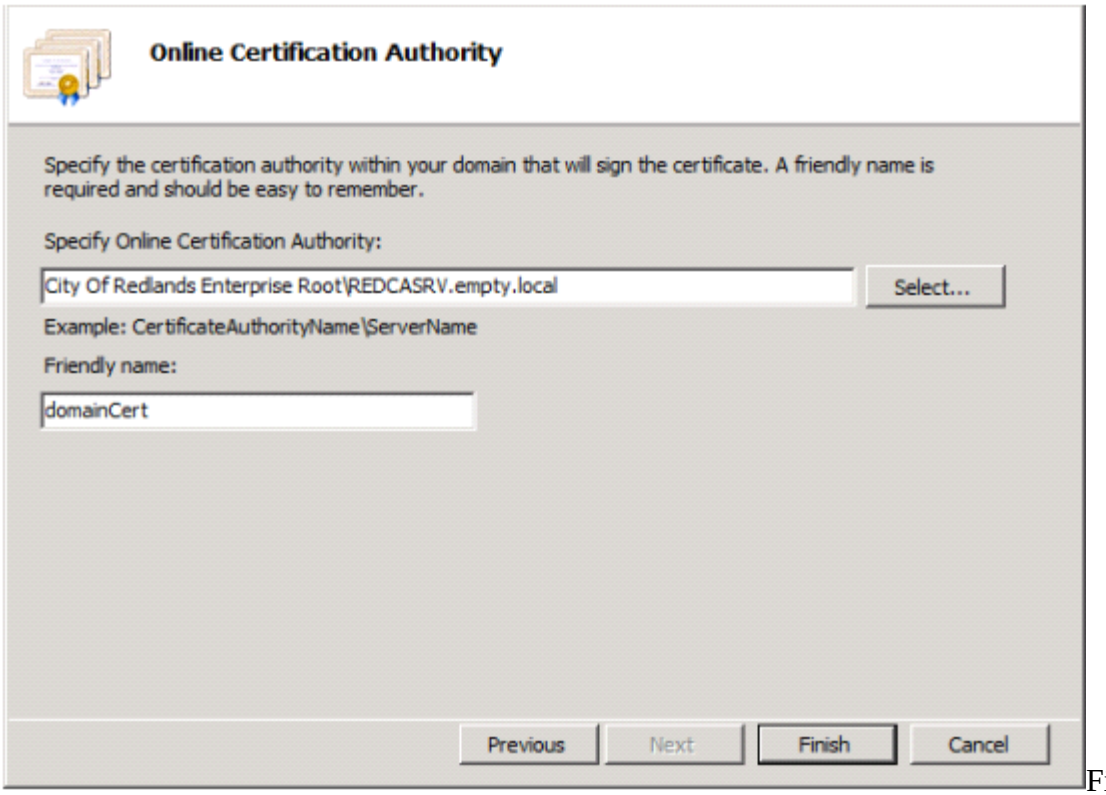


Fig. 8.23 Online Certification Authority

•Enter a user-friendly name for the domain certificate and click **Finish.**

The final step is for you to bind the domain certificate to SSL port 443.

Self-signed certificates

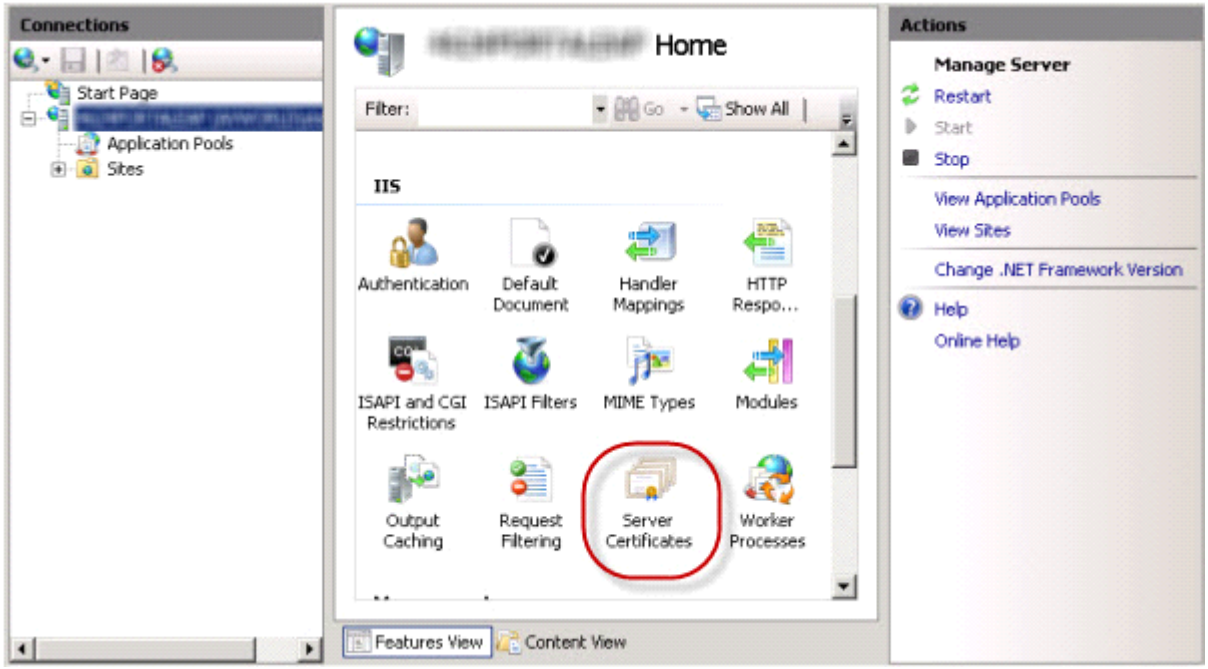
An SSL certificate signed only by the owner of the website is called a self-signed certificate. Self-signed certificates are commonly used on websites that are only available to users on the organization's internal (LAN) network. If you communicate with a

website outside your own network that uses a self-signed certificate, you have no way to verify that the site issuing the certificate really represents the party it claims to represent. You could actually be communicating with a malicious party, putting your information at risk.

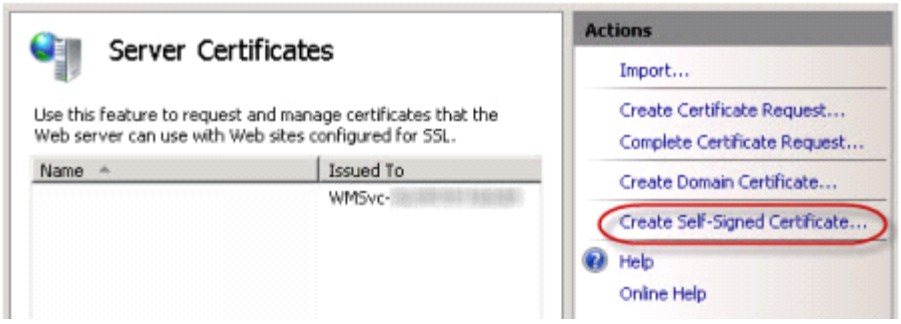
Creating a self-signed certificate in IIS

In IIS Manager, do the following to create a self-signed certificate:

•In the Connections pane, select your server in the tree view and double-click Server Certificates.



• In the Actions pane, click Create Self-Signed Certificate.



•Enter a user-friendly name for the new certificate and click OK.

The final step is for you to bind the self-signed certificate to SSL port 443.

Binding the certificate to the website

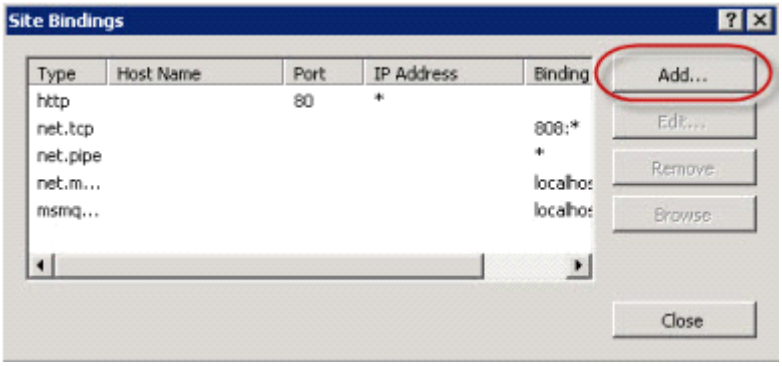
Once you've created an SSL certificate, you'll need to bind it to the website hosting the Web Adaptor. Binding refers to the process of configuring the SSL certificate to use port 443 on the website. The instructions for binding a certificate with the website vary depending on the platform and version of your web server. For instructions, consult your system administrator or your web server's documentation. For example, the steps for binding a certificate in IIS are below.

Binding a certificate to port 443 in IIS

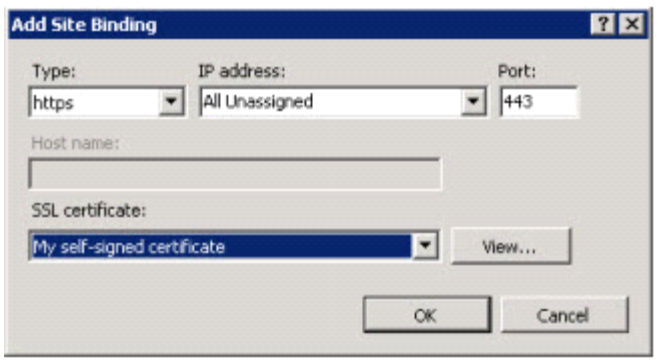
In IIS Manager, do the following to bind a certificate to SSL port 443:

•Select your site in the tree view and in the Actions pane, click Bindings.

•If port 443 is not available in the Bindings list, click Add. From the Type drop-down list, select https. Leave the port at 443.



•From the SSL certificate drop-down list, select your certificate name and click OK.



References:

•UNIX and Linux System Administration Handbook 4th Edition by Evi Nemeth, Pearson Education

- Linux System Administration Recipes 1st Edition, by Kemp Juliet, Publisher: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG
- Linux: The Complete Reference, Sixth Edition, by Richard Pearson, Tata McGraw Hill Company Limited.
- **www.linuxhomenetworking.com**
- **www.opensource.com**
- **www.linux.com**
- **www.phoenixnap.com**
- **www.booleanworld.com**
- **www.linuxtoday.com**