LINUX and Shell Scripting

DECAP448

Edited by
Rishi Chopra

LOVELY
PROFESSIONAL
UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

LINUX and Shell Scripting

Edited By:
Rishi Chopra

Title: LINUX AND SHELL SCRIPTING

Author's Name: Dr. Divya

Published By : Lovely Professional University

Publisher Address: Lovely Professional University, Jalandhar Delhi GT road, Phagwara - 144411

Printer Detail: Lovely Professional University

Edition Detail: (I)

ISBN: 978-93-94068-25-4

CONTENT

Unit 01: Getting Started with Linux

CONTENTS

Objectives

After studying this unit, you will be able to

- understand the operating system
- know the history of Linux
- understand the features of Linux
- understand the basic commands of Linux
- understand the shell of Linux

Introduction

An operating system is the low-level software that schedules tasks, allocates storage, and handles the interfaces to peripheral hardware, such as printers, disk drives, the screen, keyboard, and mouse. An operating system has two main parts: the kernel and the system programs.

- The kernel allocates machine resources—including memory, disk space, and CPU cycles—to all other programs that run on the computer.
- The system programs include device drivers, libraries, utility programs, shells (command interpreters), configuration scripts and files, application programs, servers, and documentation.

The Linux kernel was developed by Linus Torvalds. He released Linux version 0.01 in September 1991. The name 'Linux' is a combination of Linus and UNIX. The Linux OS, is a product of the Internet and is a free OS. Linux is free software. "Free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer." The UNIX OS is the common ancestor of Linux. A Linux distribution comprises the Linux kernel, utilities, and application programs. Many distributions are available, including Ubuntu, Fedora, Red Hat, Mint, OpenSUSE, Mandriva, CentOS, and Debian.

1.1 The History of UNIX and GNU–Linux

The Heritage of Linux: UNIX

The UNIX system was developed by researchers who needed a set of modern computing tools to help them with their projects. When the UNIX OS became widely available in 1975, Bell Labs offered it to educational institutions at nominal cost. The schools, colleges, universities and industries accepted the OS and worked on it.

Linux-BSD

One version is called the Berkeley Software Distribution (BSD) of the UNIX system (or just Berkeley UNIX).

Fade to 1983

Richard Stallman announced the GNU Project for creating an operating system, both kernel and system programs. GNU, which stands for Gnu's Not UNIX, is the name for the complete UNIX-compatible software system.

Next Scene, 1991

The GNU Project has moved well along toward its goal. Much of the GNU operating system, except for the kernel, is complete.

The Code is Free

The tradition of free software dates back to the days when UNIX was released to universities at nominal cost, which contributed to its portability and success. This tradition eventually died as UNIX was commercialized. Another problem with the commercial versions of UNIX related to their complexity.

1.2 What is so good about Linux

- Standards
- Applications
- Peripherals
- Software
- Platforms
- Emulators
- Virtual Machines
- Xen
- VMWare
- KVM
- Qemu
- Virtual Box

Standards

In 1985, the POSIX standard was developed, which is based largely on the SVID and other earlier standardization efforts. These efforts were spurred by the U.S. government, which needed a standard computing environment to minimize its training and procurement costs.

Applications

A rich selection of applications is available for Linux—both free and commercial—as well as a wide variety of tools: graphical, word processing, networking, security, administration, Web server, and many others.

Peripherals

Linux often supports a peripheral or interface card before any company does. Unfortunately some types of peripherals—particularly proprietary graphics cards—lag in their support.

Software's

Also important to users is the amount of software that is available—not just source code but also prebuilt binaries that are easy to install and ready to run. These programs include more than free software.

Platforms

Linux is not just for Intel-based platforms: It has been ported to and runs on the PowerPC. Nor is Linux just for single-processor machines.

Emulators

Linux supports programs, called emulators, that run code intended for other operating systems.

Virtual Machines

A virtual machine appears to the user and to the software running on it as a complete physical machine. The software that provides the virtualization is called a virtual machine monitors (VMM) or hypervisor. Each VM can run a different OS from the other VMs.

Xen

Xen, which was created at the University of Cambridge and is now being developed in the open-source community, is an open-source VMM. Xen introduces minimal performance overhead when compared with running each of the operating systems natively.

VMWare

VMware, Inc. offers VMware Server, a free, downloadable, proprietary product you can install and run as an application under Linux. VMware Server enables you to install several VMs, each running a different OS, including Windows and Linux.

KVM

The Kernel-based Virtual Machine (KVM) is an open-source VM and runs as part of the Linux kernel.

Qemu

Qemu, is an open-source VMM that runs as a user application with no CPU requirements. It can run code written for a different CPU than that of the host machine.

Virtual Box

Virtual Box is an open-source VM developed by Sun Microsystems.

1.3 Why Linux Is Popular with Hardware Companies and Developers

Two trends in the computer industry set the stage for the growing popularity of UNIX and Linux. These are proprietary and generic operating systems.

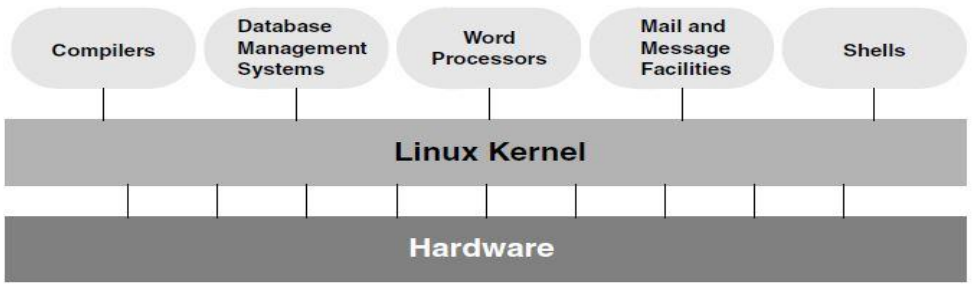
Proprietary operating systems: A proprietary OS is one that is written and owned by the manufacturer of the hardware (for example, DEC/Compaq owns VMS). Today's manufacturers need a generic OS that they can easily adapt to their machines.

Generic operating systems: Generic OS is written outside of the company manufacturing the hardware and is sold (UNIX, OS X, Windows) or given (Linux) to the manufacturer. Linux is a generic OS.

Linux emerged to serve both needs: It is a generic OS that takes advantage of available hardware. A portable OS is one that can run on many different machines. More than 95% of the Linux OS is written in the C programming language. Because Linux is portable, it can be adapted (ported) to different machines and can meet special requirements. The ancestor of Linux is UNIX. The UNIX OS was written in assembly language. For this reason, the original UNIX OS was not portable. To make UNIX portable, Thompson developed the B programming language, a machine-independent language. Dennis Ritchie developed the C programming language by modifying B and, with Thompson, rewrote UNIX in C in 1973.

1.4 Overview of Linux

Like UNIX, it is also a well-thought-out family of utility programs and a set of tools that allow users to connect and use these utilities to build systems and applications.



Kernel Programming Interface

The Linux kernel—the heart of the Linux OS—is responsible for allocating the computer's resources and scheduling user jobs so each one gets its fair share of system resources. Programs interact with the kernel through system calls. A programmer can use a single system call to interact with many kinds of devices. For example, there is one write() system call, rather than many device-specific ones. It also makes it possible to move programs to new versions of the OS without rewriting them (provided the new version recognizes the same system calls).

Supports Many Users

Depending on the hardware and the types of tasks the computer performs, a Linux system can support from 1 to more than 1,000 users, each concurrently running a different set of programs.

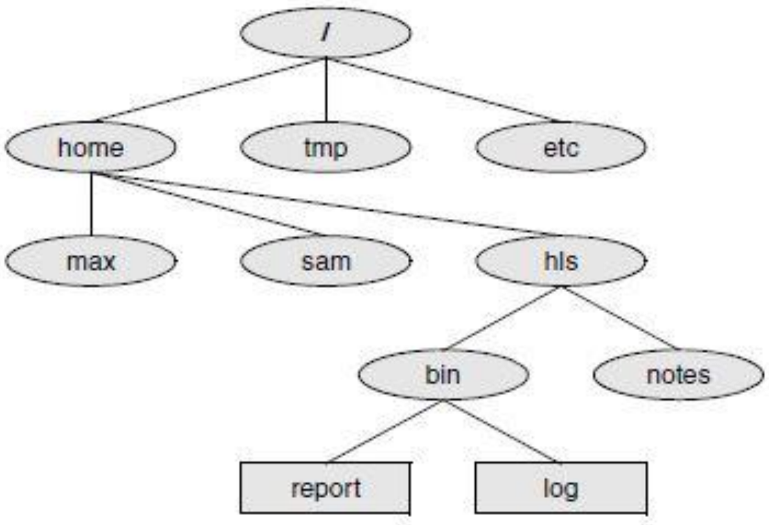
Runs Many Tasks

Linux is a fully protected multitasking OS, allowing each user to run more than one job at a time. Processes can communicate with one another but remain fully protected from one another, just as the kernel remains protected from all processes.

Secure Hierarchical File system

The Linux file system provides a structure whereby files are arranged under directories, which are like folders or boxes. Each directory has a name and can hold other files and directories. Directories, in turn, are arranged under other directories, and so forth, in a treelike organization.

Linux File System Structure



Standards

The Linux File system Standard (FSSTND) was developed, which has since evolved into the Linux File system Hierarchy Standard (FHS).

Links

A link allows a given file to be accessed by means of two or more names. The alternative names can be located in the same directory as the original file or in another directory.

Security

Like most multiuser OSs, Linux allows users to protect their data from access by other users. It also allows users to share selected data and programs with certain other users by means of a simple but effective protection scheme.

The Shell

The shell can work as a command interpreter as well as a programming language.

The shell as a command interpreter: In a textual environment, the shell—the command interpreter—acts as an interface between you and the OS. A number of shells are available for Linux. The four most popular shells are:

- Bourne Again Shell
- Debian Almquist Shell
- TC Shell
- Z Shell

The shell as a Programming Language: The shell is a high-level programming language. Shell commands can be arranged in a file for later execution which is known as shell scripts. This flexibility allows users to perform complex operations with relative ease.

Filename Generation

When you type commands to be processed by the shell, you can construct patterns using characters that have special meanings to the shell. These characters are called wildcard characters. The patterns, which are called ambiguous file references, are a kind of shorthand.

Completion

In conjunction with the Readline library, the shell performs command, filename, pathname, and variable completion.

Device-Independent Input and Output

- Redirection: When you give a command to the Linux OS, you can instruct it to send the output to any one of several devices or files. This diversion is called output redirection.
- Device independence: In a similar manner, a program's input, which normally comes from a keyboard, can be redirected so that it comes from a disk file instead.

Shell Functions

Many shells, including the BASH, support shell functions that the shell holds in memory so it does not have to read them from the disk each time you execute them.

Job Control

Job control is a shell feature that allows users to work on several jobs at once, switching back and forth between them as desired.

A Large Collection of Useful Utilities

Linux includes a family of several hundred utility programs, often referred to as commands. These utilities perform functions that are universally required by users. For example: Sort utility.

Inter process Communication

Linux enables users to establish both pipes and filters on the command line. A pipe sends the output of one program to another program as input. A filter is a special kind of pipe that processes a stream of input data to yield a stream of output data.

System Administration

On a Linux system the system administrator is frequently the owner and only user of the system. This person has many responsibilities. The first responsibility may be to set up the system, install the software, and possibly edit configuration files.

Additional Features of Linux

The developers of Linux included features from BSD, System V, and Sun Microsystems' Solaris, as well as new features, in their OS.

GUIs: Graphical User Interfaces

X11:

Given a terminal or workstation screen that supports X, a user can interact with the computer through multiple windows on the screen, display graphical information, or use special-purpose applications to draw pictures, monitor processes, or preview formatted output. Usually two layers run on top of X: a desktop manager and a window manager.

Desktop manager: A desktop manager is a picture-oriented user interface that enables you to interact with system programs by manipulating icons instead of typing the corresponding commands to a shell.

Window manager: A window manager is a program that runs under the desktop manager and allows you to open and close windows, run programs, and set up a mouse so it has different effects depending on how and where you click.

(Inter)Networking Utilities

Linux network support includes many utilities that enable you to access remote systems over a variety of networks. In addition to sending email to users on other systems, you can access files on disks mounted on other computers.

Software Development

One of Linux's most impressive strengths is its rich software development environment. Linux supports compilers and interpreters for many computer languages.

Utilities

These utilities facilitate you for a large task. There are various utilities like ls, cd, cat, mv, cp, echo and date.

Summary

- An operating system has two main parts: the kernel and the system programs.
- A Linux distribution comprises the Linux kernel, utilities, and application programs.
- Linux supports programs, called emulators, that run code intended for other operating systems.
- The software that provides the virtualization is called a virtual machine monitor.
- The Kernel-based Virtual Machine (KVM) is an open-source VM and runs as part of the Linux kernel.
- A portable OS is one that can run on many different machines.
- The four most popular shells are: Bourne Again Shell, Debian Almquist Shell, TC Shell and Z Shell.
- Shell commands can be arranged in a file for later execution which are known as shell scripts.

Keywords

- **ls utility**: A utility used for listing the contents of a directory.
- **mv utility:** A utility used for moving the files from one directory to another.
- **cd utility:** A utility used for changing the directory.
- **Operating System:** An operating system is the low-level software that schedules tasks, allocates storage, and handles the interfaces to peripheral hardware, such as printers, disk drives, the screen, keyboard, and mouse.
- **Emulators**: Linux supports programs, called emulators, that run code intended for other operating systems.
- **Proprietary OS:** It is the one that is written and owned by the manufacturer of the hardware.
- **Generic OS:** It is written outside of the company manufacturing the hardware and is sold or given to the manufacturer.

Self Assessment

1. An operating system is responsible for
   A. Scheduling the tasks.
   B. Allocation the storage
   C. Handling the interfaces for peripherals
   D. All of the above
2. An operating system has
   A. Kernel
   B. System programs
   C. Both of the above
   D. None of the above
3. Which of these are distributions of Linux?
   A. Fedora
   B. RedHat
   C. CentOS
   D. All of the above
4. A Linux distribution comprises of
   A. Application programs
   B. Utilities
   C. Kernel
   D. All of the above
5. Which of these shells are available in Linux?
   A. TC Shell
   B. Z Shell
   C. Debian Almquist Shell
   D. All of the above
6. Which is the core of operating system?
   A. Kernel
   B. Commands
   C. Shell
   D. Script

7. Which of these utilities is used to show which files and folders are available in the system?
   A. ls
   B. cat
   C. rm
   D. All of the above
8. Linux OS supports
   A. Multi User
   B. Multi Process
   C. Multi-tasking
   D. All of the above
9. Which of these utilities is used for concatenation of files?
   A. ls
   B. cat
   C. rm
   D. echo
10. Which of these utilities is used for displaying the contents of a file?
    A. ls
    B. cat
    C. rm
    D. echo
11. Linux is an example of
    A. Web browser
    B. Word processing software
    C. Operating system
    D. Photo editor
12. Who founded the Linux Kernel?
    A. Richard Stallman
    B. Ben Thomas
    C. Linus Torvalds
    D. None of the above
13. Which of the following OS is not based upon Linux?
    A. BSD
    B. Redhat
    C. Ubuntu
    D. CentOS
14. What is available to users in Linux OS?
    A. Source code
    B. Prebuilt binaries
    C. Both source code and prebuilt binaries
    D. None of the above

15. Which of these represents system programs?

    A. Libraries
    B. Device drivers
    C. Servers
    D. All of the above

Answers for Self Assessment

1.	D	2.	C	3.	D	4.	D	5.	D
6.	A	7.	A	8.	D	9.	B	10.	B
11.	C	12.	C	13.	A	14.	C	15.	D

Review Questions:

1. What is an operating system? Explain its main parts.
2. What are the features of Linux? Explain.
3. What are proprietary and generic operating systems? Explain why Linux is popular with hardware companies and developers?
4. What is so good about Linux? Explain about its applications, peripherals, platforms and standards.
5. What is kernel programming interface? Explain.
6. What are the basic utilities in Linux? Explain.

Further Readings

Mark G Sobell, A Practical Guide to Linux Commands, Editors, and Shell Programming, Second Edition.

Web Links

https://www.redhat.com/en/topics/linux/what-is-linux

Unit 02: Installation Guide

CONTENTS

Objectives

After studying this unit, you will be able to

- Understand the booting process
- Understand the installation process of Linux
- Understand the partitioning of hard drives
- Understand the file system types
- Understand how to log in the system

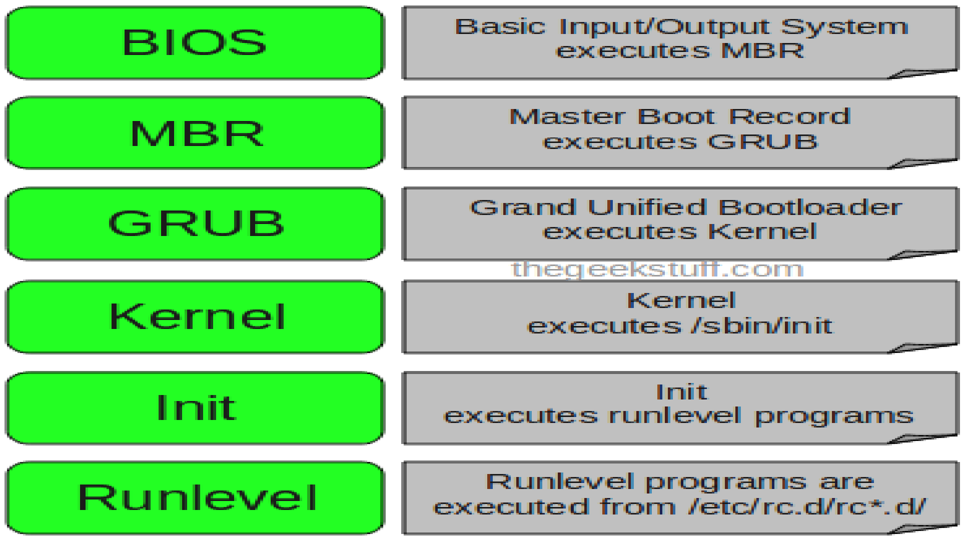
Introduction: An operating system (OS) is a system software that manages computer hardware, software resources, and provides common services for computer programs. Booting is a bootstrapping process that starts operating systems when the user turns on a computer system. A boot sequence is the set of operations the computer performs when it is switched on that load an operating system.

2.1 Booting sequence:

The booting sequence follows several steps. These are:

- Turn on the system
- CPU jump to address of BIOS
- BIOS runs POST (Power-On Self-Test)
- Find bootable devices
- Loads and execute boot sector form MBR
- Load OS

The first and foremost task is to turn on the system. So that the other processes can start. Rest of the process includes jumping of CPU to the address of CPU, BIOS runs POST, finding of bootable devices, loads and execute boot sector from MBR and loading of operating system.


BIOS
Basic Input/Output System executes MBR
MBR
Master Boot Record executes GRUB
GRUB
Grand Unified Bootloader executes Kernel
thegeekstuff.com
Kernel
Kernel executes /sbin/init
Init
Init executes runlevel programs
Runlevel
Runlevel programs are executed from /etc/rc.d/rc*.d/

BIOS (Basic Input/ Output System)

BIOS refer to the software code run by a computer when first powered on. It identifies your computer's hardware, configures it, tests it, and connects it to the operating system for further instruction. This is called the boot process. The primary function of BIOS is code program embedded on a chip that recognizes and controls various devices that make up the computer.

MBR(Master Boot Record)

OS is booted from a hard disk, where the Master Boot Record (MBR) contains the primary boot loader.The MBR is a 512-byte sector, located in the first sector on the disk (sector 1 of cylinder 0, head 0).After the MBR is loaded into RAM, the BIOS yields control to it.

Boot loader

Boot loader could be more adeptly called the kernel loader. The task at this stage is to load the Linux kernel.GRUB and LILO are the most popular Linux boot loader.Examples of boot loaders are:

Example: GRUB, LILO, GRUB2WIN, BOOTCAMP, BOOTKEY, NTLDR and Syslinux

GRUB:

GRUB stands for GRand Unified Bootloader.It is an operating system independent boot loader. It is a multiboot software packet from GNU.It has a flexible command line interface. It has file system access. It supports multiple executable formats. It supports diskless system.

LILO: LInux Loader

This boot loader does not depend on a specific file system.It can boot from hard-disk and floppy

Task of kernel

The kernel helps in process management, memory management. The device management is also one of the tasks of kernel.The system calls are also handled by kernel.

Init process

The first thing the kernel does is to execute init program. Init is the root/parent of all processes executing on Linux. The first processes that init starts is a script /etc/rc.d/rc.sysinit. Based on the appropriate run-level, scripts are executed to start various processes to run the system and make it functional. The init process is identified by process id "1".Init is responsible for starting system processes as defined in the /etc/inittab file.Upon shutdown, init controls the sequence and processes for shutdown.

Runlevels

A run-level is a software configuration of the system which allows only a selected group of processes to exist. Init can be in one of seven run-levels: 0-6.

Runlevel	**ScriptsDirectory (RedHat/Fedora Core)**	**State**
0	/etc/rc.d/rc0.d/	shutdown/halt system
1	/etc/rc.d/rc1.d/	Single user mode
2	/etc/rc.d/rc2.d/	Multiuser with no network services exported
3	/etc/rc.d/rc3.d/	Default text/console only start. Full multiuser
4	/etc/rc.d/rc4.d/	Reserved for local use. Also X-windows (Slackware/BSD)
5	/etc/rc.d/rc5.d/	XDM X-windows GUI mode (Redhat/System V)
6	/etc/rc.d/rc6.d/	Reboot
s or S		Single user/Maintenance mode (Slackware)
M		Multiuser mode (Slackware)

2.2 Download Linux

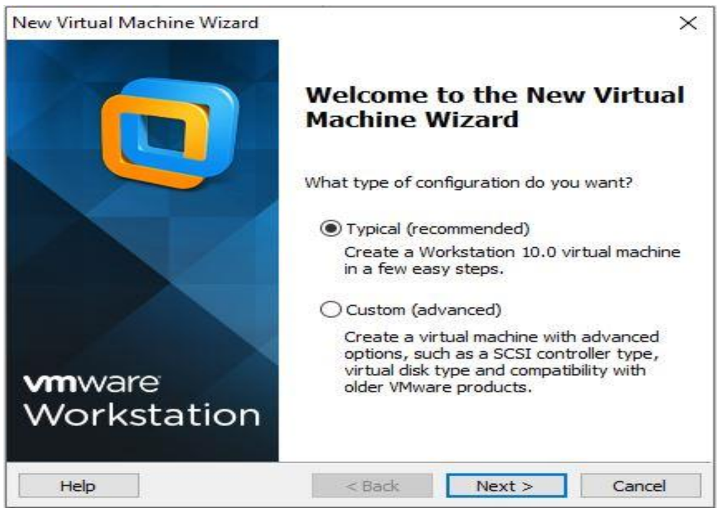
- To install Red Hat, you will need to download the ISO images (CD Images) of the installation CD-ROMs from http://fedora.redhat.com
- Download the i386 images for 32 Intel Processors, PPC images for Apple Macintosh and x86_64 for 64-bit AMD Processors.

Here the installation of Linux distribution is shown under VMWare. It can be installed as an individual operating system as well. Generally, when we buy a new system, the seller gives us the Windows operating system by default in that.

2.3 Installation of Linux

New Virtual Machine Wizard:

After the installation of VMWare in the system, the New Virtual Machine Wizard will open. It asks for the type of configuration you want to go with: typical or custom. The typical configuration is the recommended one and custom is the configuration with the advanced options. Choose the appropriate option and go to next step.



New Virtual Machine Wizard

Welcome to the New Virtual Machine Wizard

What type of configuration do you want?

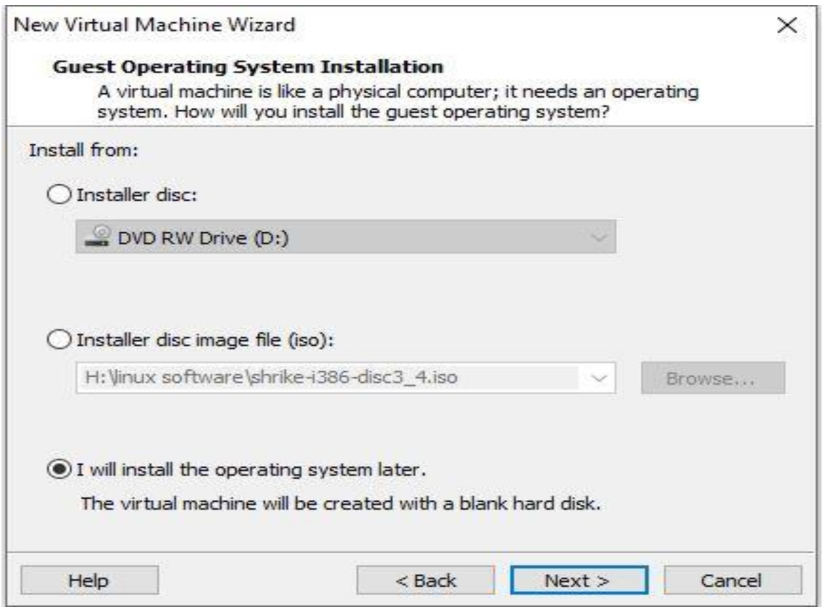
[✓ Typical (recommended)]
Create a Workstation 10.0 virtual machine in a few easy steps.

Custom (advanced)
Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.

Installation of operating system

In this step, the guest operating system will be installed. Here the available options are

- Installer disc
- Installer disc image file (iso)
- Later installation.



New Virtual Machine Wizard

Guest Operating System Installation
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

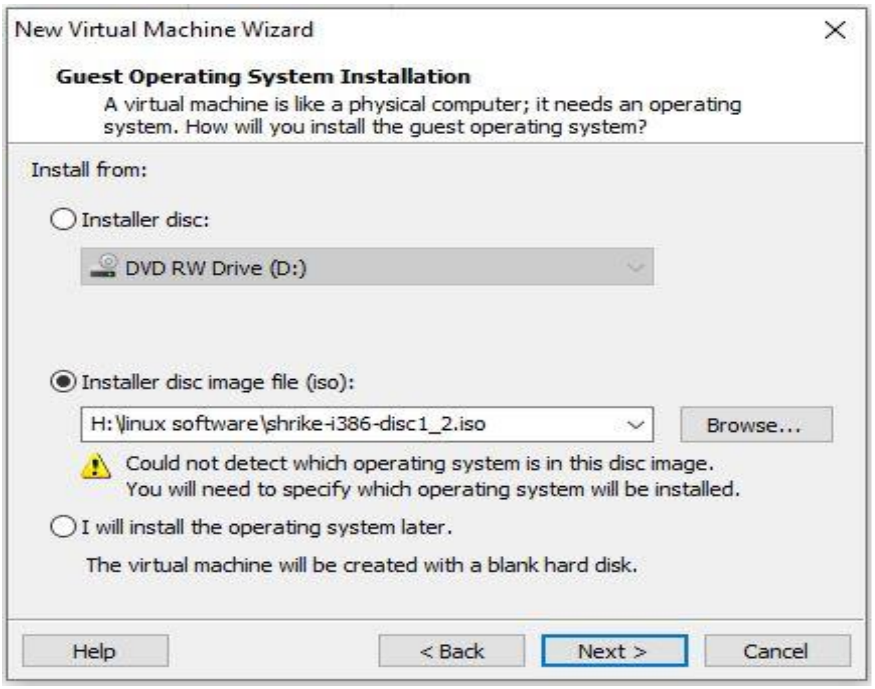
Install from:

Installer disc:
DVD RW Drive (D:)

Installer disc image file (iso):
H:\linux software\shrike-i386-disc3_4.iso

[✓ I will install the operating system later.]
The virtual machine will be created with a blank hard disk.

So out of these three options, choose the option depending upon the availability of the operating system. Here the installation using iso images is shown. The whole operating system is divided into three iso images.


New Virtual Machine Wizard

Guest Operating System Installation

A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

○ Installer disc:

DVD RW Drive (D:)

● Installer disc image file (iso):

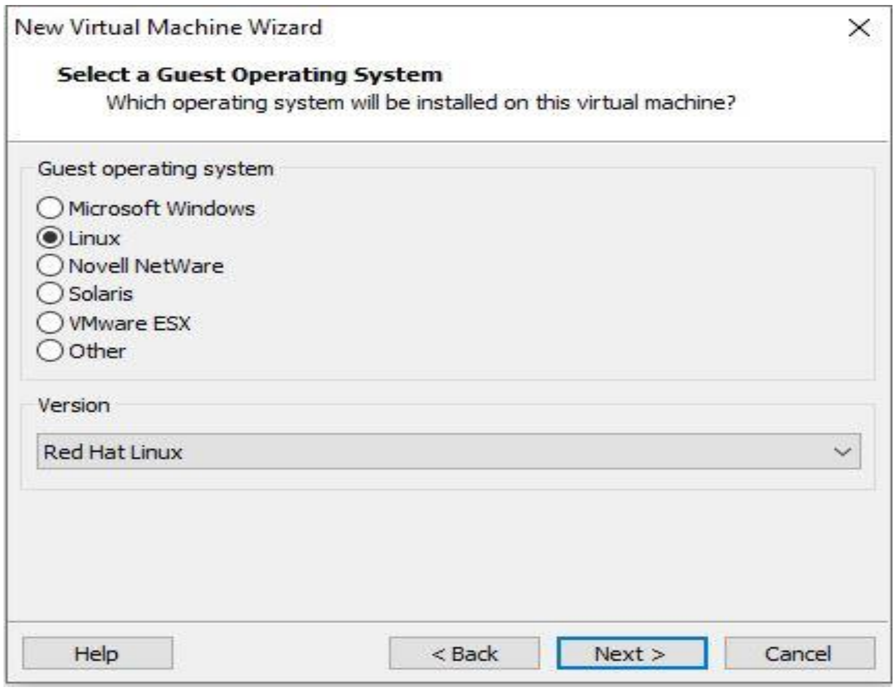
H:\linux software\shrike-i386-disc1_2.iso

Could not detect which operating system is in this disc image.
You will need to specify which operating system will be installed.

○ I will install the operating system later.

The virtual machine will be created with a blank hard disk.

After selecting the first iso image, choose which guest operating system you want to install. Here the Linux is chosen for installation on the virtual machine. The version of the operating system will also be selected here. Once it is done, click on Next to go further.


New Virtual Machine Wizard

Select a Guest Operating System

Which operating system will be installed on this virtual machine?

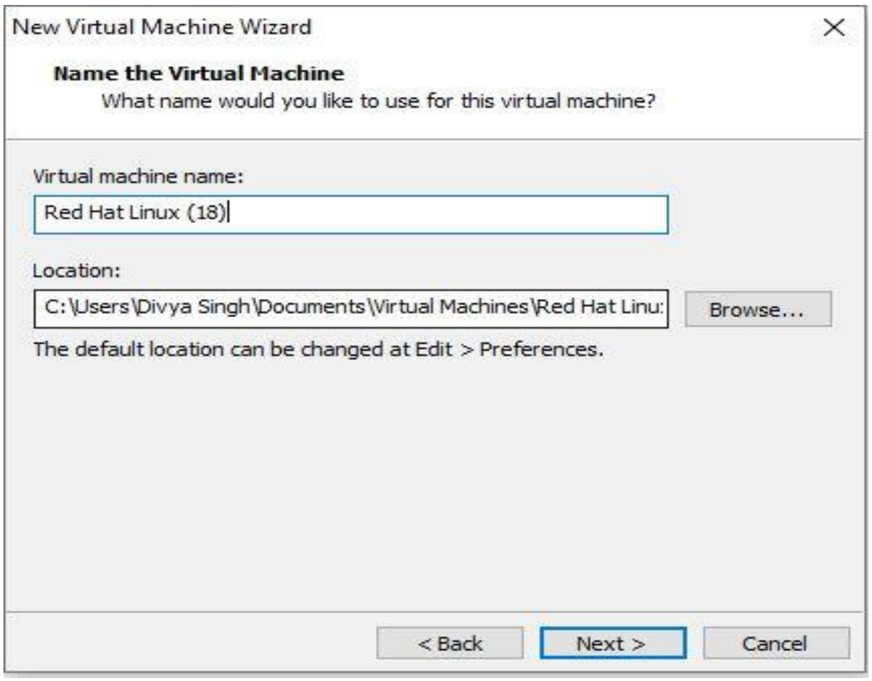
Guest operating system

○ Microsoft Windows
● Linux
○ Novell NetWare
○ Solaris
○ VMware ESX
○ Other

Version

Red Hat Linux

Write the name of the virtual machine and choose the location where you want to have all of its files. Once this is done, click on next to go further.



New Virtual Machine Wizard

Name the Virtual Machine

What name would you like to use for this virtual machine?

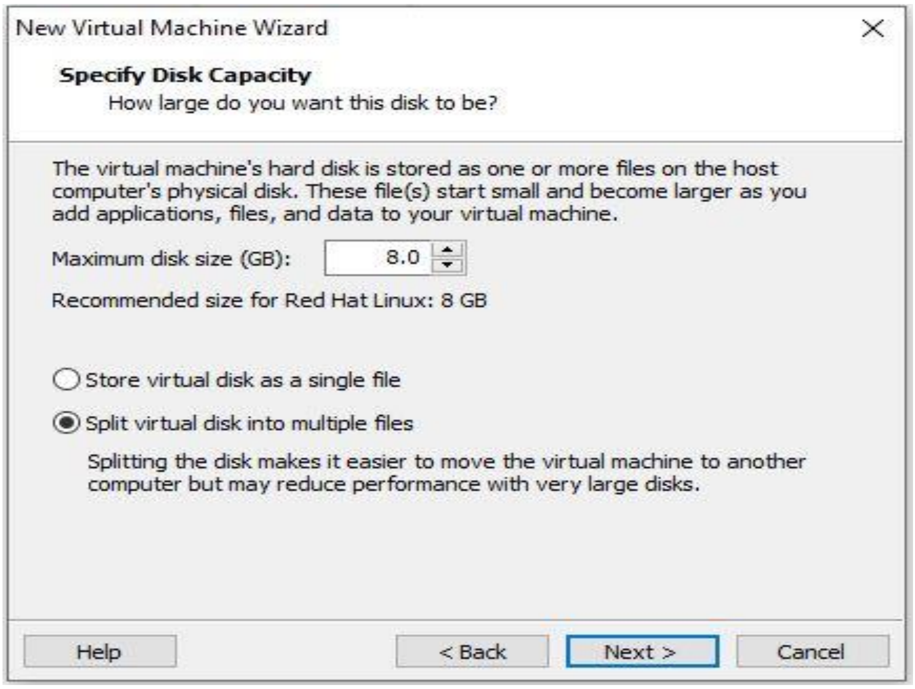
Virtual machine name: Red Hat Linux (18)

Location: C:\Users\Divya Singh\Documents\Virtual Machines\Red Hat Linu:

The default location can be changed at Edit > Preferences.

Specify disk capacity

When the name and location is specified, next task is to specify the disk capacity. By default, the recommended maximum disk size is 8 GB. We can increase or decrease it as per our requirements. Then it will ask whether you want to store virtual disk as a single file or into multiple files. Choose the appropriate option and go further.



New Virtual Machine Wizard

Specify Disk Capacity

How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

Maximum disk size (GB): 8.0

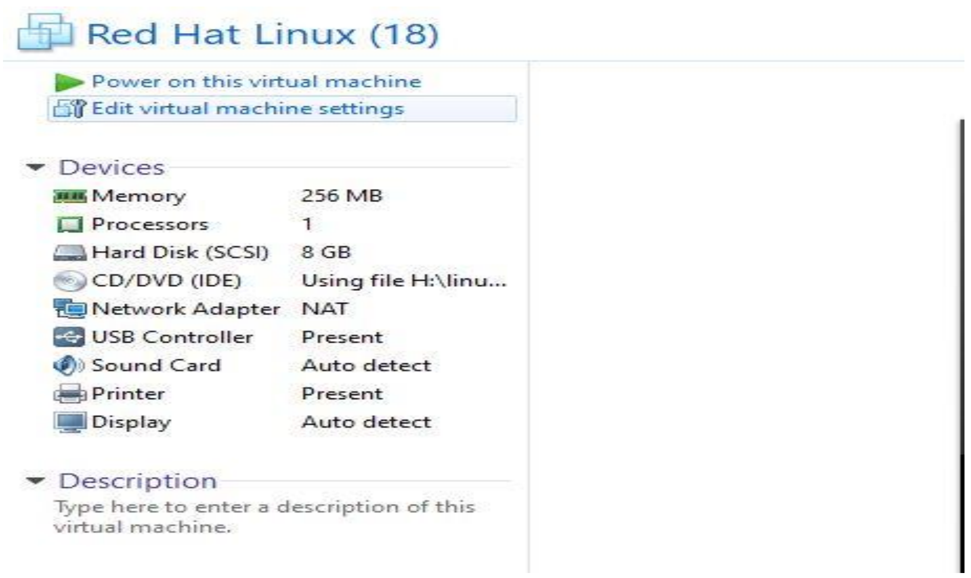
Recommended size for Red Hat Linux: 8 GB

○ Store virtual disk as a single file

● Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

After this, there are two options. One is to power on this virtual machine and other is to edit virtual machine settings. If you have done any mistake while setting the machine, then edit those settings. Otherwise turn on the power machine.



Red Hat Linux (18)

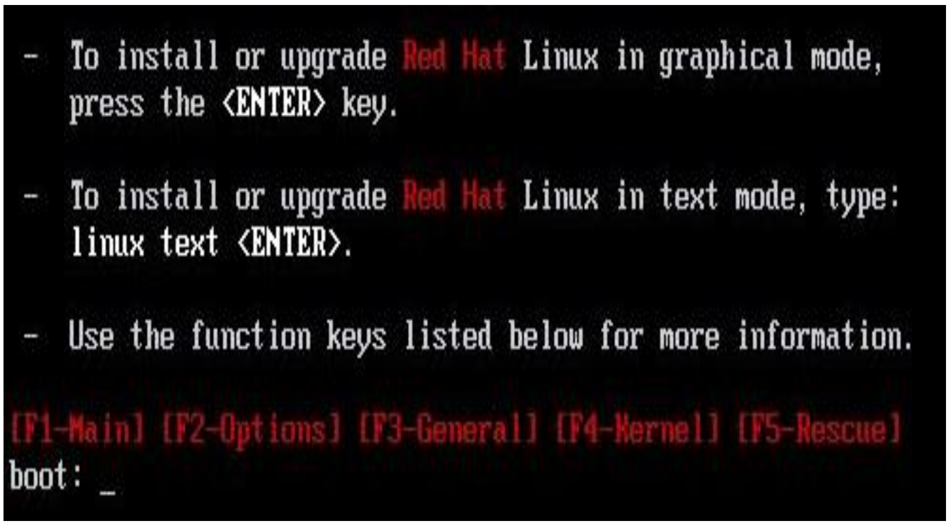
Power on this virtual machine
Edit virtual machine settings

Devices

Memory	256 MB
Processors	1
Hard Disk (SCSI)	8 GB
CD/DVD (IDE)	Using file H:\linu...
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

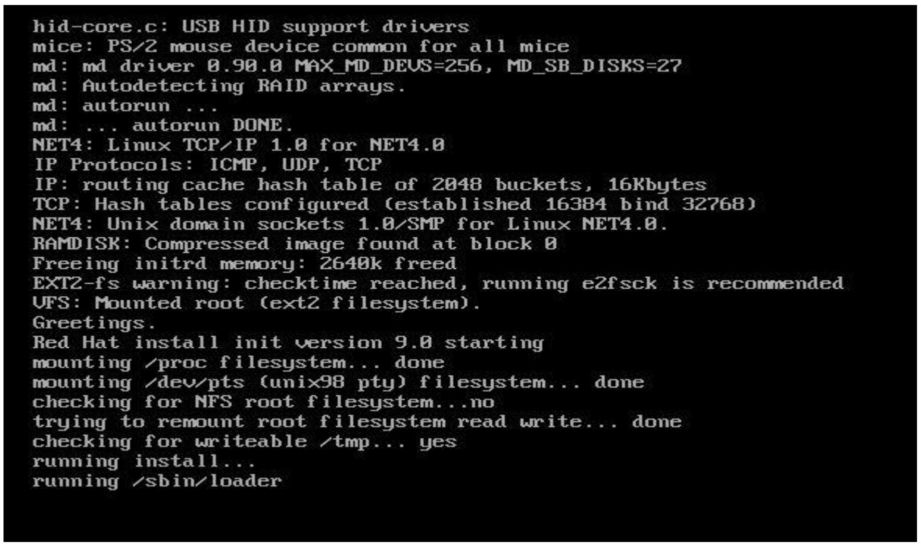
Description
Type here to enter a description of this virtual machine.

When you power on the virtual machine, the next screen will be:



```
- To install or upgrade Red Hat Linux in graphical mode,
  press the <ENTER> key.

- To install or upgrade Red Hat Linux in text mode, type:
  linux text <ENTER>.

- Use the function keys listed below for more information.

[F1-Main] [F2-Options] [F3-General] [F4-Kernel] [F5-Rescue]
boot: _
```

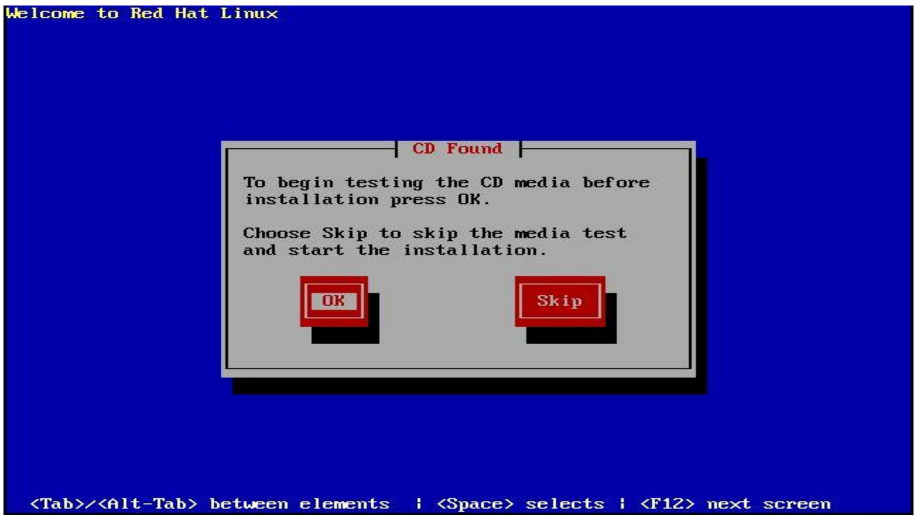
When you press the ENTER key, the next screen will be:



```
hid-core.c: USB HID support drivers
mice: PS/2 mouse device common for all mice
md: md driver 0.90.0 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 2048 buckets, 16Kbytes
TCP: Hash tables configured (established 16384 bind 32768)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
RAMDISK: Compressed image found at block 0
Freeing initrd memory: 2640k freed
EXT2-fs warning: checktime reached, running e2fsck is recommended
VFS: Mounted root (ext2 filesystem).
Greetings.
Red Hat install init version 9.0 starting
mounting /proc filesystem... done
mounting /dev/pts (unix98 pty) filesystem... done
checking for NFS root filesystem...no
trying to remount root filesystem read write... done
checking for writeable /tmp... yes
running install...
running /sbin/loader
```

Testing the CD Media

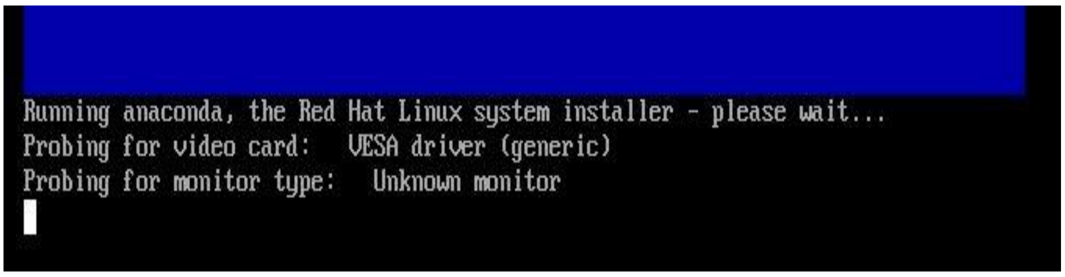
If you are using the CD media for the first time, then it is advised to test it before the installation process. If the same media is used for installation earlier, then it can be skipped.



```
Welcome to Red Hat Linux

CD Found

To begin testing the CD media before
installation press OK.

Choose Skip to skip the media test
and start the installation.

OK        Skip

<Tab>/<Alt-Tab> between elements  | <Space> selects | <F12> next screen
```

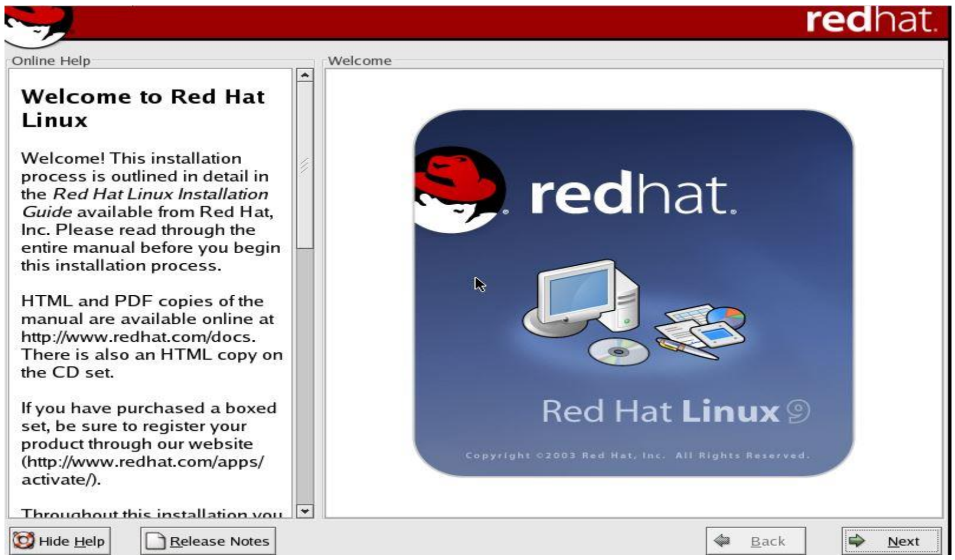
Running Anaconda

The Anaconda starts running. It is best to install Anaconda for the local user, which does not require administrator permissions and is the most robust type of installation. However, if you need to, you can install Anaconda system wide, which does require administrator permissions.



```
Running anaconda, the Red Hat Linux system installer - please wait...
Probing for video card:   VESA driver (generic)
Probing for monitor type:   Unknown monitor
```

The process starts with the screen "Welcome to Red Hat Linux". Click on Next for further process.



Online Help

Welcome to Red Hat Linux

Welcome! This installation process is outlined in detail in the *Red Hat Linux Installation Guide* available from Red Hat, Inc. Please read through the entire manual before you begin this installation process.

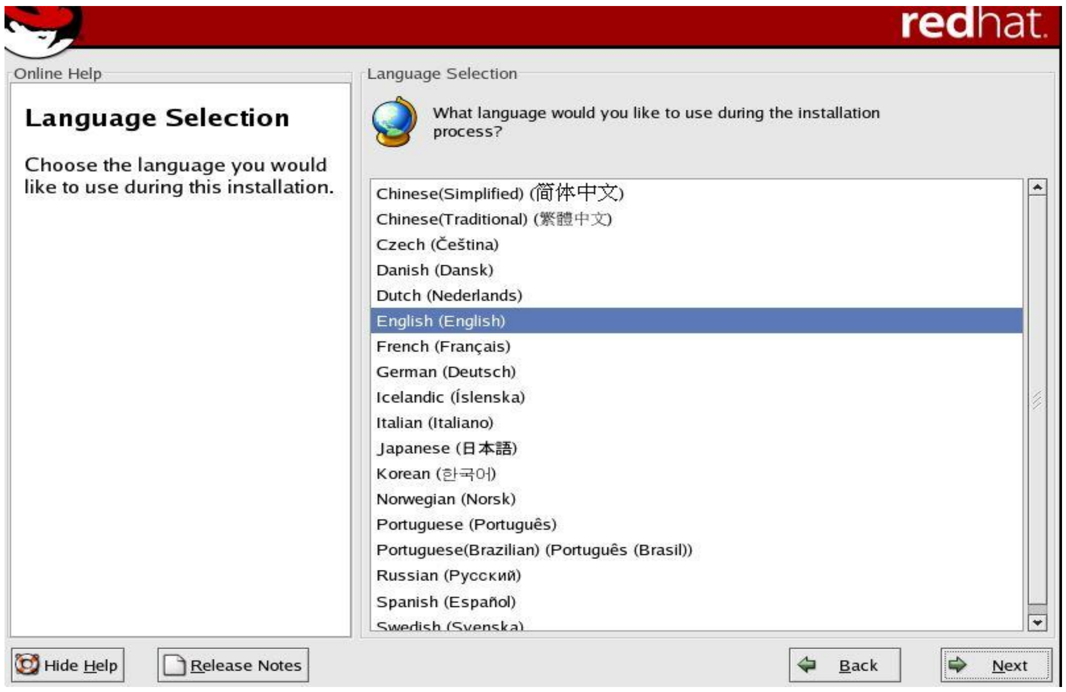
HTML and PDF copies of the manual are available online at http://www.redhat.com/docs. There is also an HTML copy on the CD set.

If you have purchased a boxed set, be sure to register your product through our website (http://www.redhat.com/apps/activate/).

Throughout this installation you

Language Selection

The next screen is for language selection. Choose the language that you want to use during installation process. After selection of language, click on next.



Language Selection

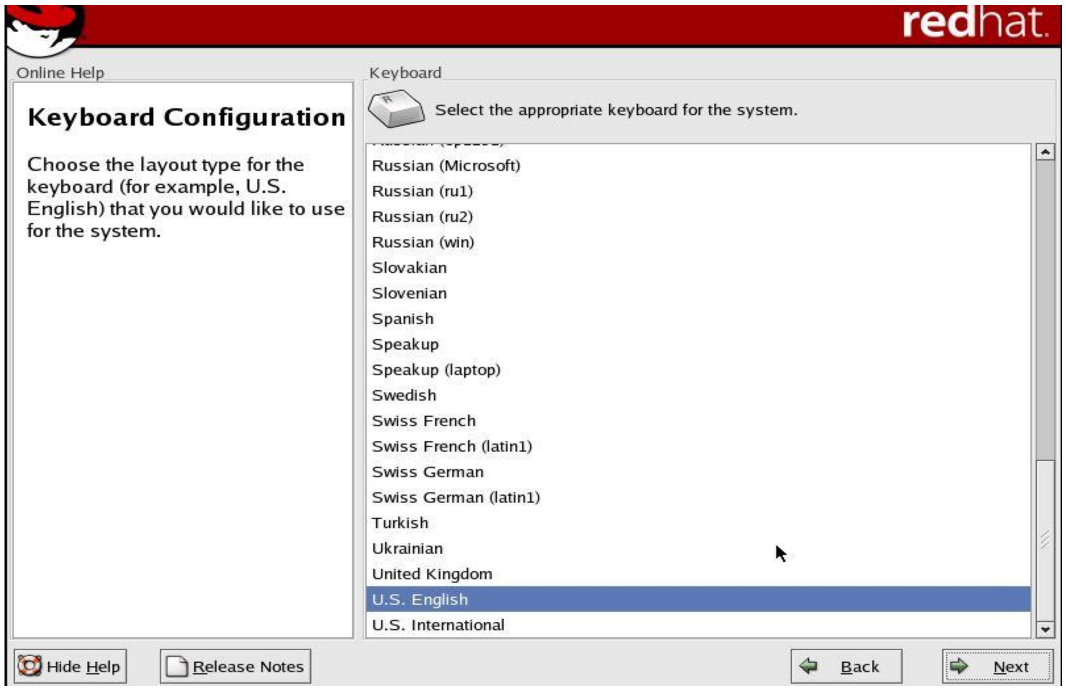
Choose the language you would like to use during this installation.

What language would you like to use during the installation process?

Chinese(Simplified) (简体中文)
Chinese(Traditional) (繁體中文)
Czech (Čeština)
Danish (Dansk)
Dutch (Nederlands)
English (English)
French (Français)
German (Deutsch)
Icelandic (Íslenska)
Italian (Italiano)
Japanese (日本語)
Korean (한국어)
Norwegian (Norsk)
Portuguese (Português)
Portuguese(Brazilian) (Português (Brasil))
Russian (Русский)
Spanish (Español)
Swedish (Svenska)

Keyboard Configuration

The screen is for choosing of layout type of keyboard that we want to use for the system. After chosen, click on next.



Keyboard Configuration

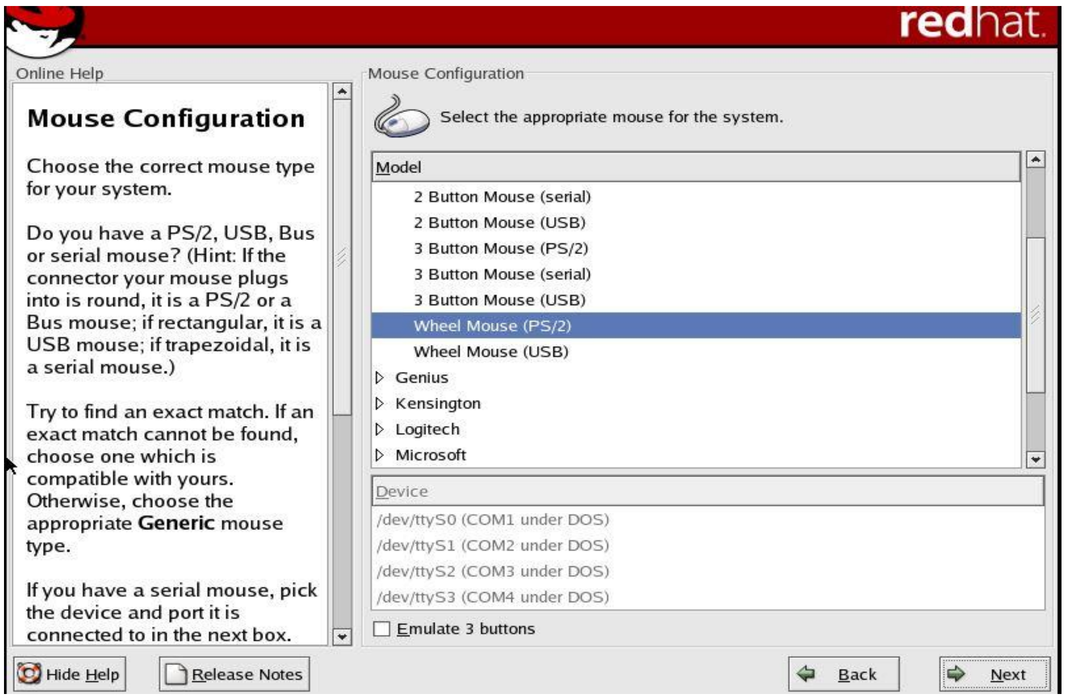
Choose the layout type for the keyboard (for example, U.S. English) that you would like to use for the system.

Select the appropriate keyboard for the system.

Russian (Microsoft)
Russian (ru1)
Russian (ru2)
Russian (win)
Slovakian
Slovenian
Spanish
Speakup
Speakup (laptop)
Swedish
Swiss French
Swiss French (latin1)
Swiss German
Swiss German (latin1)
Turkish
Ukrainian
United Kingdom
U.S. English
U.S. International

Mouse Configuration

The screen is for mouse configuration. By default, it will choose the appropriate option. So, it is advised to go with the recommended settings. Click on next.



Online Help

Mouse Configuration

Choose the correct mouse type for your system.

Do you have a PS/2, USB, Bus or serial mouse? (Hint: If the connector your mouse plugs into is round, it is a PS/2 or a Bus mouse; if rectangular, it is a USB mouse; if trapezoidal, it is a serial mouse.)

Try to find an exact match. If an exact match cannot be found, choose one which is compatible with yours. Otherwise, choose the appropriate **Generic** mouse type.

If you have a serial mouse, pick the device and port it is connected to in the next box.

Mouse Configuration

Select the appropriate mouse for the system.

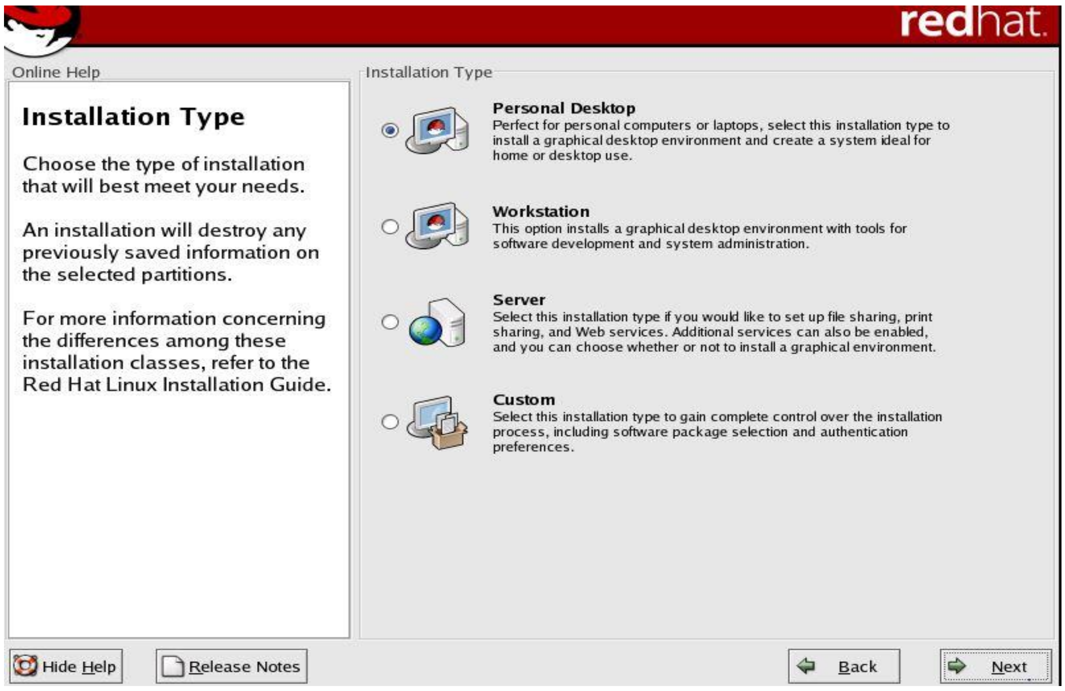
Model
- 2 Button Mouse (serial)
- 2 Button Mouse (USB)
- 3 Button Mouse (PS/2)
- 3 Button Mouse (serial)
- 3 Button Mouse (USB)
- [✓ Wheel Mouse (PS/2)]
- Wheel Mouse (USB)
- ▷ Genius
- ▷ Kensington
- ▷ Logitech
- ▷ Microsoft

Device
- /dev/ttyS0 (COM1 under DOS)
- /dev/ttyS1 (COM2 under DOS)
- /dev/ttyS2 (COM3 under DOS)
- /dev/ttyS3 (COM4 under DOS)

☐ Emulate 3 buttons

Installation Type

Next is to choose the installation type which best meet your needs. The installation type available are: Personal Desktop, workstation, server and custom. Choose the appropriate option and click on next.



Online Help

Installation Type

Choose the type of installation that will best meet your needs.

An installation will destroy any previously saved information on the selected partitions.

For more information concerning the differences among these installation classes, refer to the Red Hat Linux Installation Guide.

Installation Type

[✓ **Personal Desktop**]
Perfect for personal computers or laptops, select this installation type to install a graphical desktop environment and create a system ideal for home or desktop use.

Workstation
This option installs a graphical desktop environment with tools for software development and system administration.

Server
Select this installation type if you would like to set up file sharing, print sharing, and Web services. Additional services can also be enabled, and you can choose whether or not to install a graphical environment.

Custom
Select this installation type to gain complete control over the installation process, including software package selection and authentication preferences.

Disk Partitioning Setup

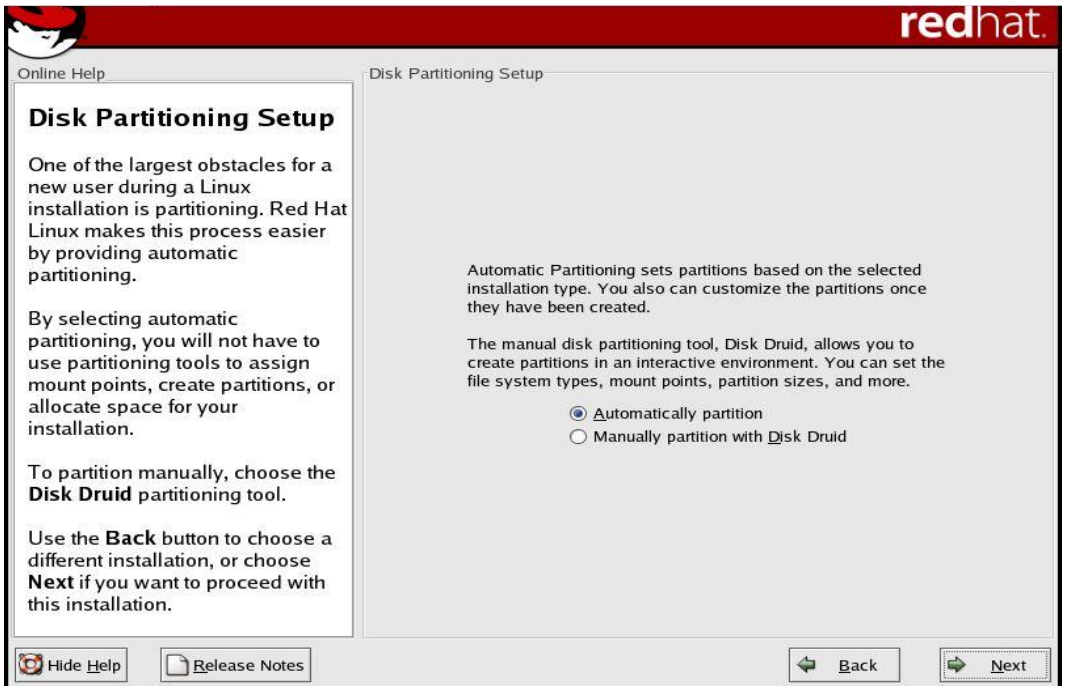
Partitioning is a means to divide a single hard drive into many logical drives. A partition is a contiguous set of blocks on a drive that are treated as an independent disk. A partition table is an index that relates sections of the hard drive to partitions.

The task is now to partition the drive. It can be done in two ways: automatically partition and manually partition with disk druid. The automatically partition is dependent upon the selected installation type. The manual partition uses the tool Disk Druid for partitioning. The ppartition fields are:

- **Device:** This field displays the partition's device name.
- **Start:** This field shows the sector on your hard drive where the partition begins.
- **End**: This field shows the sector on your hard drive where the partition ends.
- **Size**: This field shows the partition's size (in MB).
- **Type:** This field shows the partition's type (for example, ext2, ext3, or vfat).
- **Mount Point**: A mount point is the location within the directory hierarchy at which a volume exists; the volume is "mounted" at this location. This field indicates where the partition will be mounted.

The file system types are:

- **ext2 —** An ext2 filesystem supports standard Unix file types (regular files, directories, symbolic links, etc). It provides the ability to assign long file names, up to 255 characters. Versions prior to Red Hat Linux 7.2 used ext2 file systems by default.
- **ext3 —** The ext3 filesystem is based on the ext2 filesystem and has one main advantage — journaling. Using a journaling filesystem reduces time spent recovering a filesystem after a crash as there is no need to fsck the filesystem.
- **swap —** Swap partitions are used to support virtual memory. In other words, data is written to a swap partition when there is not enough RAM to store the data your system is processing.
- **vfat —** The VFAT filesystem is a Linux filesystem that is compatible with Windows 95/NT long filenames on the FAT filesystem.



Disk Partitioning Setup

One of the largest obstacles for a new user during a Linux installation is partitioning. Red Hat Linux makes this process easier by providing automatic partitioning.

By selecting automatic partitioning, you will not have to use partitioning tools to assign mount points, create partitions, or allocate space for your installation.

To partition manually, choose the **Disk Druid** partitioning tool.

Use the **Back** button to choose a different installation, or choose **Next** if you want to proceed with this installation.

Automatic Partitioning sets partitions based on the selected installation type. You also can customize the partitions once they have been created.

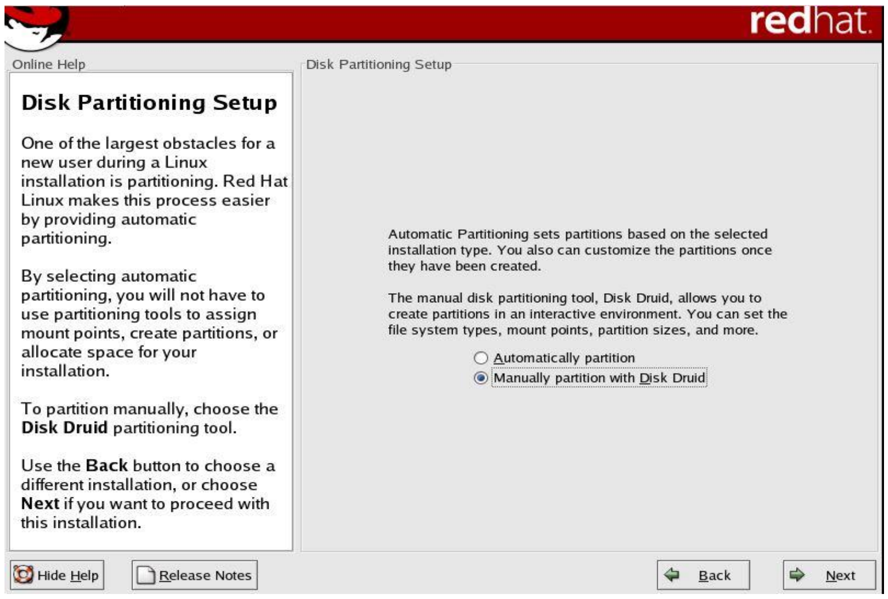
The manual disk partitioning tool, Disk Druid, allows you to create partitions in an interactive environment. You can set the file system types, mount points, partition sizes, and more.

[✓ Automatically partition]
Manually partition with Disk Druid

Recommended Partitioning Scheme

- Unless you have a reason for doing otherwise, it is recommended that you create the following partitions:
- /boot partition – contains kernel images and grub configuration and commands
- / partition
- /var partition
- Any other partition based on application (e.g /usr/local for squid)
- swap partition — swap partitions are used to support virtual memory. In other words, data is written to a swap partition when there is not enough RAM to store the data your system is processing. The size of your swap partition should be equal to twice your computer's RAM.

Here we are going with manual partition in which disk druid tool helps.



Online Help

Disk Partitioning Setup

One of the largest obstacles for a new user during a Linux installation is partitioning. Red Hat Linux makes this process easier by providing automatic partitioning.

By selecting automatic partitioning, you will not have to use partitioning tools to assign mount points, create partitions, or allocate space for your installation.

To partition manually, choose the **Disk Druid** partitioning tool.

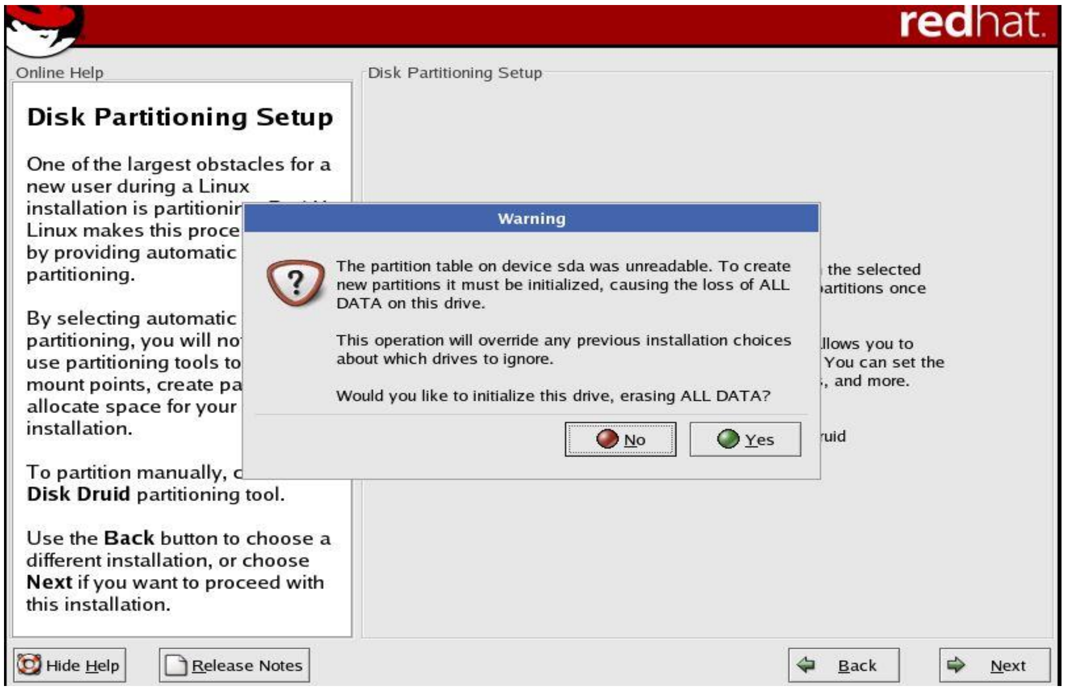
Use the **Back** button to choose a different installation, or choose **Next** if you want to proceed with this installation.

Disk Partitioning Setup

Automatic Partitioning sets partitions based on the selected installation type. You also can customize the partitions once they have been created.

The manual disk partitioning tool, Disk Druid, allows you to create partitions in an interactive environment. You can set the file system types, mount points, partition sizes, and more.

○ Automatically partition
[✓ ◉ Manually partition with Disk Druid]



Online Help

Disk Partitioning Setup

One of the largest obstacles for a new user during a Linux installation is partitionir
Linux makes this proce
by providing automatic partitioning.

By selecting automatic partitioning, you will no
use partitioning tools to
mount points, create pa
allocate space for your installation.

To partition manually, c
Disk Druid partitioning tool.

Use the **Back** button to choose a different installation, or choose **Next** if you want to proceed with this installation.

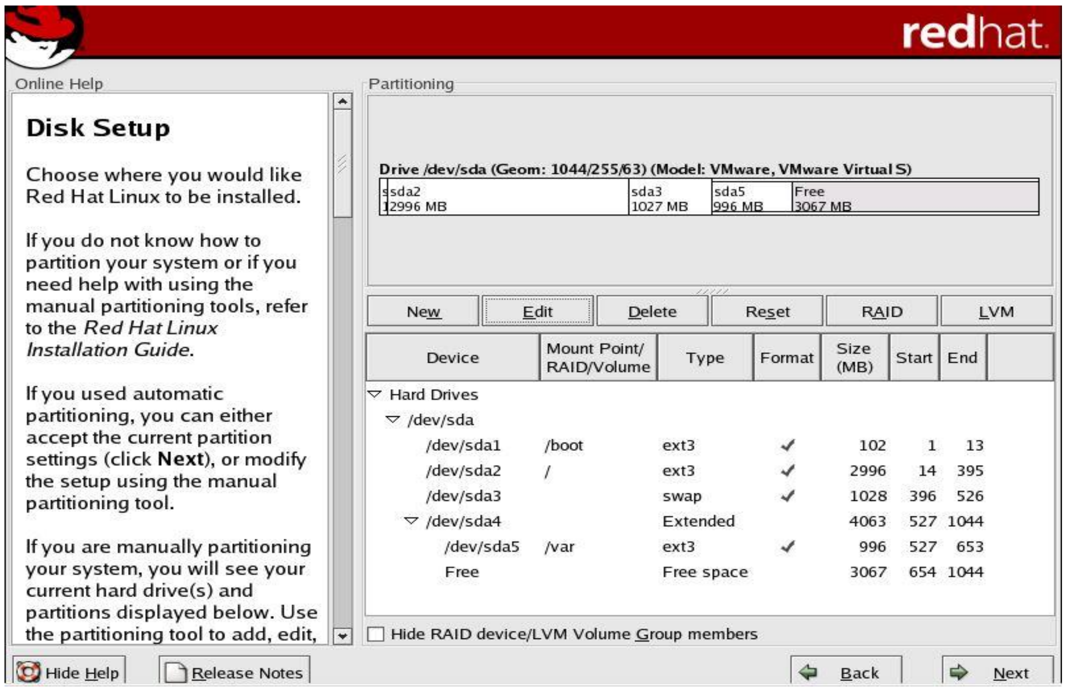
Disk Partitioning Setup

Warning

The partition table on device sda was unreadable. To create new partitions it must be initialized, causing the loss of ALL DATA on this drive.

This operation will override any previous installation choices about which drives to ignore.

Would you like to initialize this drive, erasing ALL DATA?



Disk Setup

Choose where you would like Red Hat Linux to be installed.

If you do not know how to partition your system or if you need help with using the manual partitioning tools, refer to the *Red Hat Linux Installation Guide*.

If you used automatic partitioning, you can either accept the current partition settings (click **Next**), or modify the setup using the manual partitioning tool.

If you are manually partitioning your system, you will see your current hard drive(s) and partitions displayed below. Use the partitioning tool to add, edit,

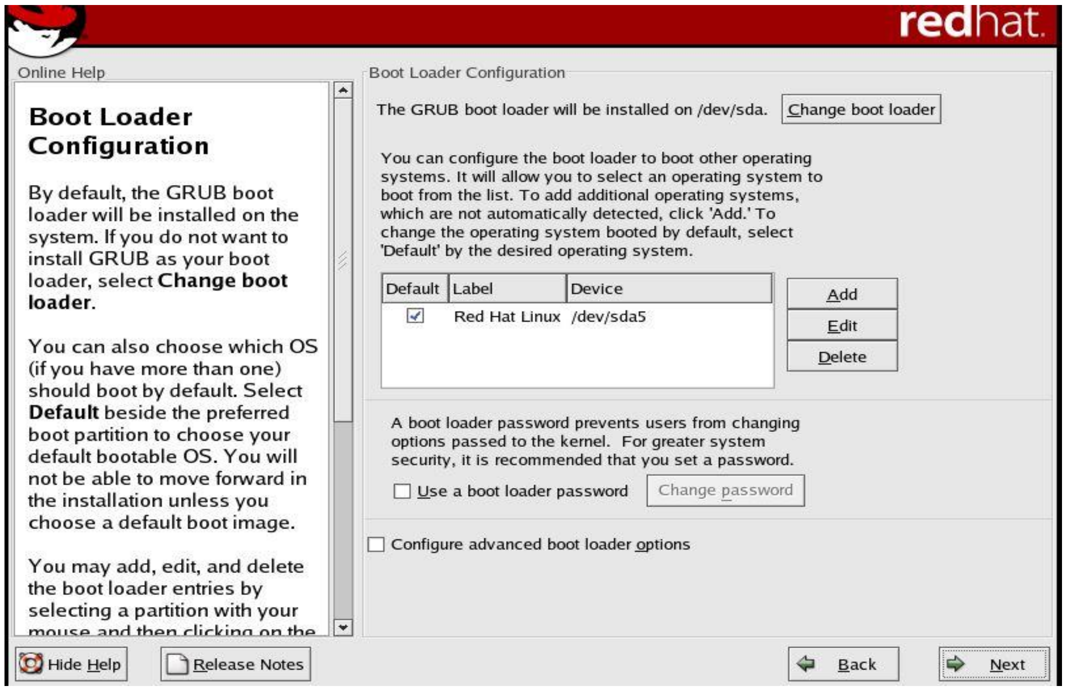
Drive /dev/sda (Geom: 1044/255/63) (Model: VMware, VMware Virtual S)

sda2 12996 MB | sda3 1027 MB | sda5 996 MB | Free 3067 MB

Device	Mount Point/ RAID/Volume	Type	Format	Size (MB)	Start	End
Hard Drives						
/dev/sda						
/dev/sda1	/boot	ext3	✓	102	1	13
/dev/sda2	/	ext3	✓	2996	14	395
/dev/sda3		swap	✓	1028	396	526
/dev/sda4		Extended		4063	527	1044
/dev/sda5	/var	ext3	✓	996	527	653
Free		Free space		3067	654	1044

☐ Hide RAID device/LVM Volume Group members

Boot Loader Configuration



Boot Loader Configuration

By default, the GRUB boot loader will be installed on the system. If you do not want to install GRUB as your boot loader, select **Change boot loader**.

You can also choose which OS (if you have more than one) should boot by default. Select **Default** beside the preferred boot partition to choose your default bootable OS. You will not be able to move forward in the installation unless you choose a default boot image.

You may add, edit, and delete the boot loader entries by selecting a partition with your mouse and then clicking on the

The GRUB boot loader will be installed on /dev/sda.

You can configure the boot loader to boot other operating systems. It will allow you to select an operating system to boot from the list. To add additional operating systems, which are not automatically detected, click 'Add.' To change the operating system booted by default, select 'Default' by the desired operating system.

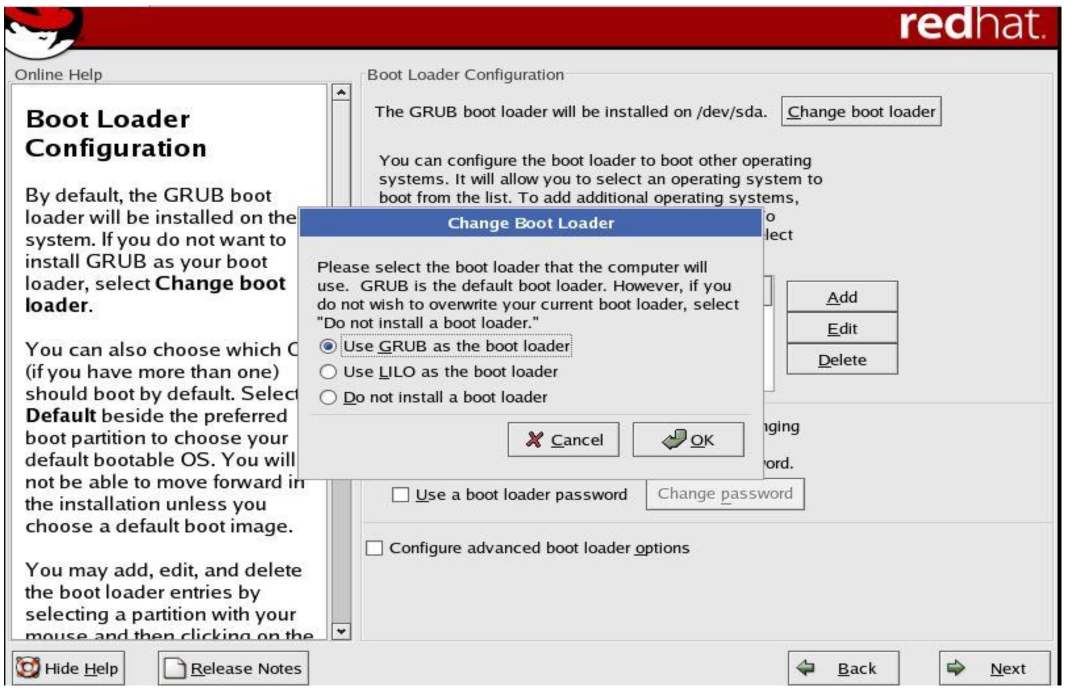
Default	Label	Device
✓	Red Hat Linux	/dev/sda5

A boot loader password prevents users from changing options passed to the kernel. For greater system security, it is recommended that you set a password.

☐ Use a boot loader password

☐ Configure advanced boot loader options

Change Boot Loader



Boot Loader Configuration

By default, the GRUB boot loader will be installed on the system. If you do not want to install GRUB as your boot loader, select **Change boot loader**.

You can also choose which C (if you have more than one) should boot by default. Selec **Default** beside the preferred boot partition to choose your default bootable OS. You will not be able to move forward in the installation unless you choose a default boot image.

You may add, edit, and delete the boot loader entries by selecting a partition with your mouse and then clicking on the

Boot Loader Configuration

The GRUB boot loader will be installed on /dev/sda. Change boot loader

You can configure the boot loader to boot other operating systems. It will allow you to select an operating system to boot from the list. To add additional operating systems,

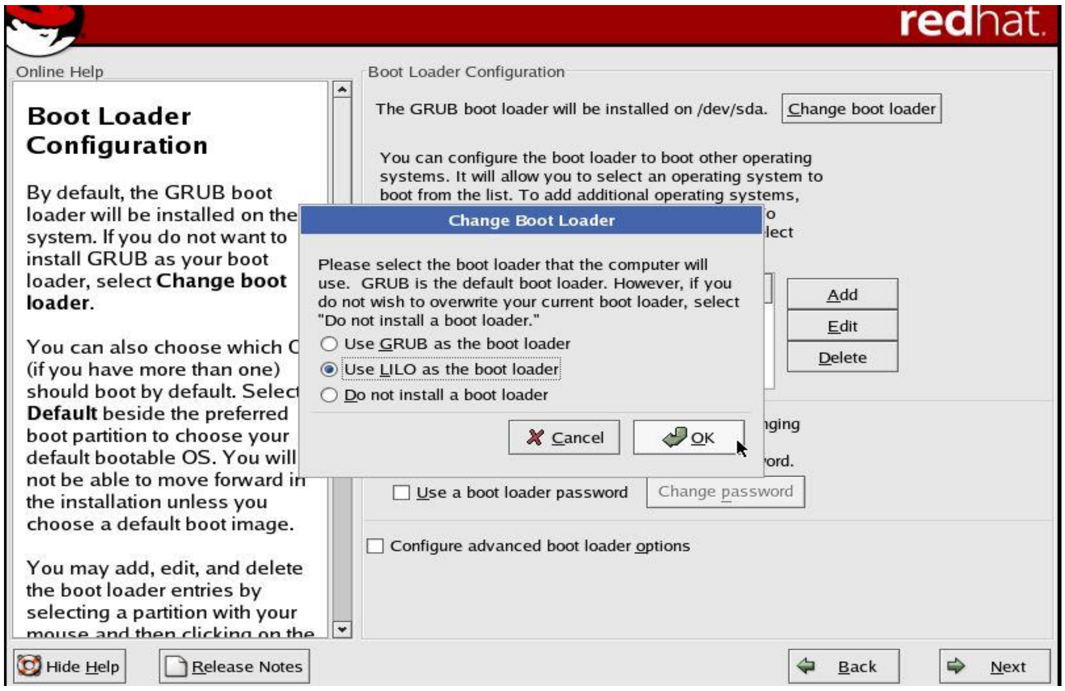
Change Boot Loader

Please select the boot loader that the computer will use. GRUB is the default boot loader. However, if you do not wish to overwrite your current boot loader, select "Do not install a boot loader."

[✓ Use GRUB as the boot loader]
Use LILO as the boot loader
Do not install a boot loader

Use a boot loader password — Change password

Configure advanced boot loader options



Boot Loader Configuration

By default, the GRUB boot loader will be installed on the system. If you do not want to install GRUB as your boot loader, select **Change boot loader**.

You can also choose which C (if you have more than one) should boot by default. Selec **Default** beside the preferred boot partition to choose your default bootable OS. You will not be able to move forward in the installation unless you choose a default boot image.

You may add, edit, and delete the boot loader entries by selecting a partition with your mouse and then clicking on the

Boot Loader Configuration

The GRUB boot loader will be installed on /dev/sda. Change boot loader

You can configure the boot loader to boot other operating systems. It will allow you to select an operating system to boot from the list. To add additional operating systems,

Change Boot Loader

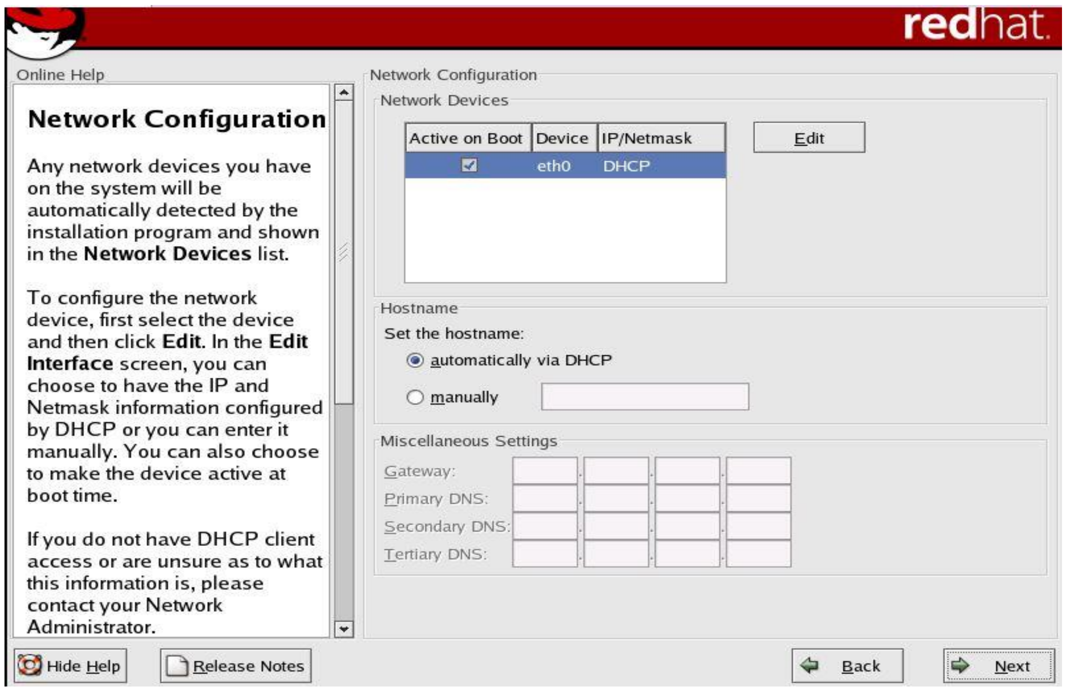
Please select the boot loader that the computer will use. GRUB is the default boot loader. However, if you do not wish to overwrite your current boot loader, select "Do not install a boot loader."

Use GRUB as the boot loader
[✓ Use LILO as the boot loader]
Do not install a boot loader

Use a boot loader password — Change password

Configure advanced boot loader options

Network Configuration



Network Configuration

Any network devices you have on the system will be automatically detected by the installation program and shown in the **Network Devices** list.

To configure the network device, first select the device and then click **Edit**. In the **Edit Interface** screen, you can choose to have the IP and Netmask information configured by DHCP or you can enter it manually. You can also choose to make the device active at boot time.

If you do not have DHCP client access or are unsure as to what this information is, please contact your Network Administrator.

Network Configuration

Network Devices

Active on Boot	Device	IP/Netmask
✓	eth0	DHCP

Hostname

Set the hostname:

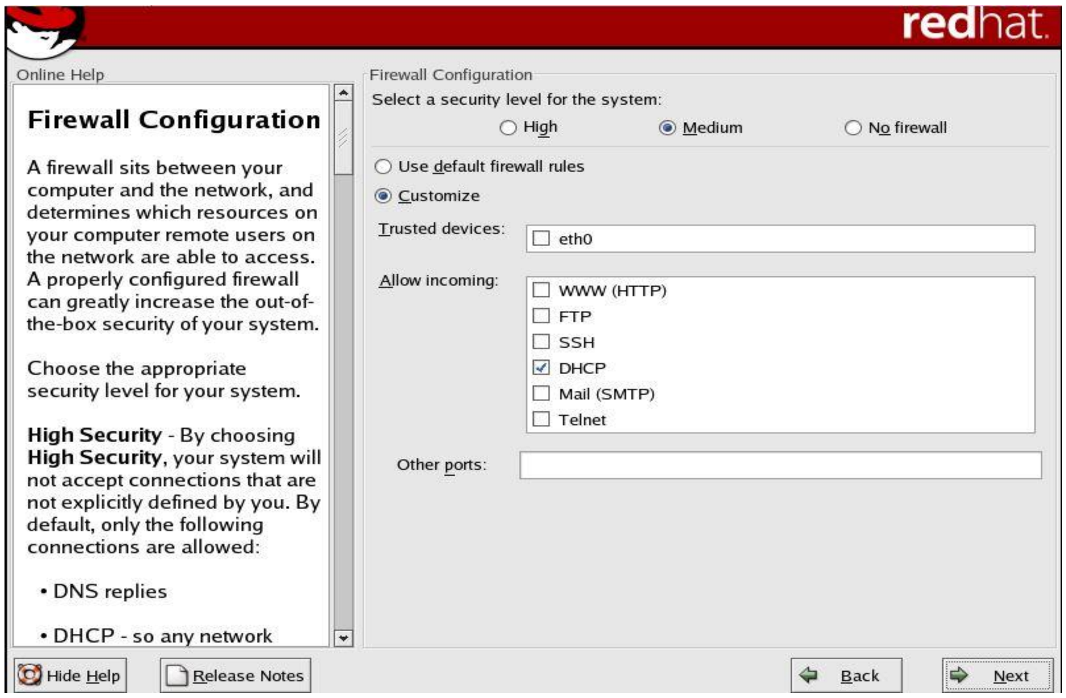
[✓ automatically via DHCP]

manually

Miscellaneous Settings

Gateway:
Primary DNS:
Secondary DNS:
Tertiary DNS:

Firewall Configuration



Firewall Configuration

A firewall sits between your computer and the network, and determines which resources on your computer remote users on the network are able to access. A properly configured firewall can greatly increase the out-of-the-box security of your system.

Choose the appropriate security level for your system.

High Security - By choosing **High Security**, your system will not accept connections that are not explicitly defined by you. By default, only the following connections are allowed:

• DNS replies

• DHCP - so any network

Firewall Configuration

Select a security level for the system:

High [✓ Medium] No firewall

Use default firewall rules

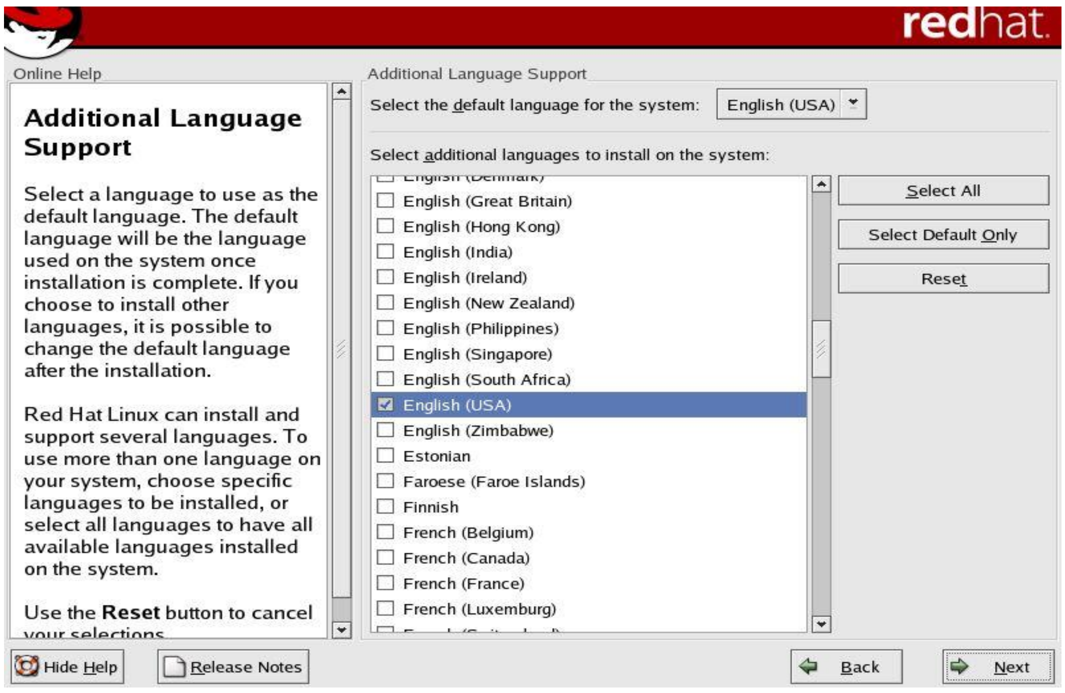
[✓ Customize]

Trusted devices: ☐ eth0

Allow incoming:
- ☐ WWW (HTTP)
- ☐ FTP
- ☐ SSH
- ☑ DHCP
- ☐ Mail (SMTP)
- ☐ Telnet

Other ports:

Additional Language Support



Additional Language Support

Select a language to use as the default language. The default language will be the language used on the system once installation is complete. If you choose to install other languages, it is possible to change the default language after the installation.

Red Hat Linux can install and support several languages. To use more than one language on your system, choose specific languages to be installed, or select all languages to have all available languages installed on the system.

Use the **Reset** button to cancel your selections.

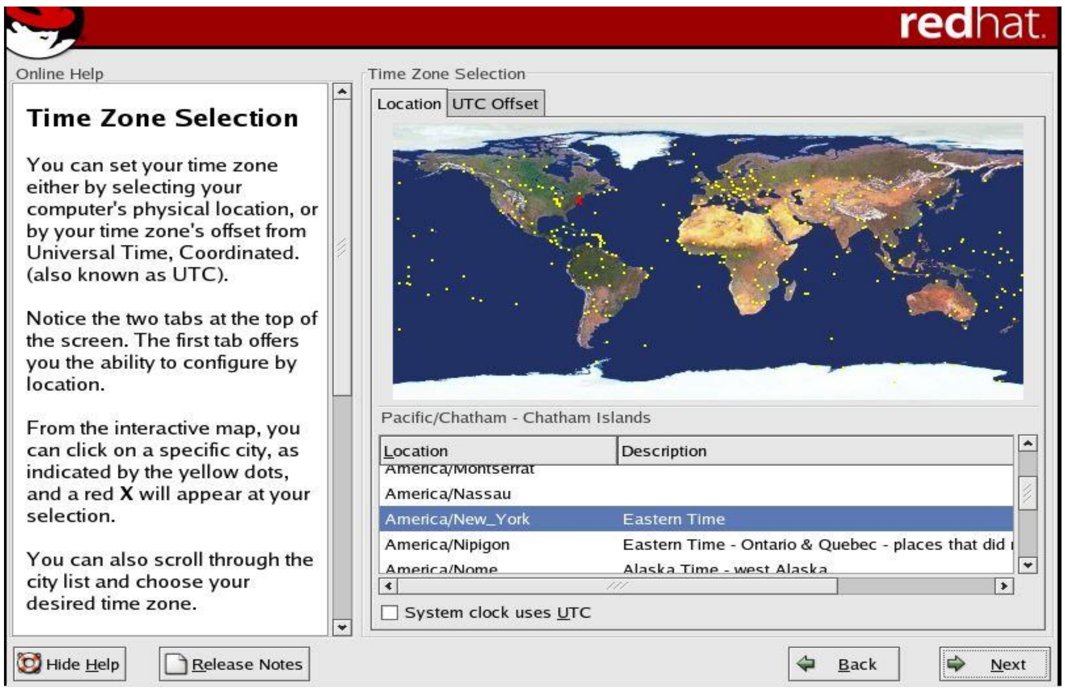
Additional Language Support

Select the default language for the system: English (USA)

Select additional languages to install on the system:

- ☐ English (Great Britain)
- ☐ English (Hong Kong)
- ☐ English (India)
- ☐ English (Ireland)
- ☐ English (New Zealand)
- ☐ English (Philippines)
- ☐ English (Singapore)
- ☐ English (South Africa)
- ☑ English (USA)
- ☐ English (Zimbabwe)
- ☐ Estonian
- ☐ Faroese (Faroe Islands)
- ☐ Finnish
- ☐ French (Belgium)
- ☐ French (Canada)
- ☐ French (France)
- ☐ French (Luxemburg)

Time Zone Selection



Time Zone Selection

You can set your time zone either by selecting your computer's physical location, or by your time zone's offset from Universal Time, Coordinated. (also known as UTC).

Notice the two tabs at the top of the screen. The first tab offers you the ability to configure by location.

From the interactive map, you can click on a specific city, as indicated by the yellow dots, and a red **X** will appear at your selection.

You can also scroll through the city list and choose your desired time zone.

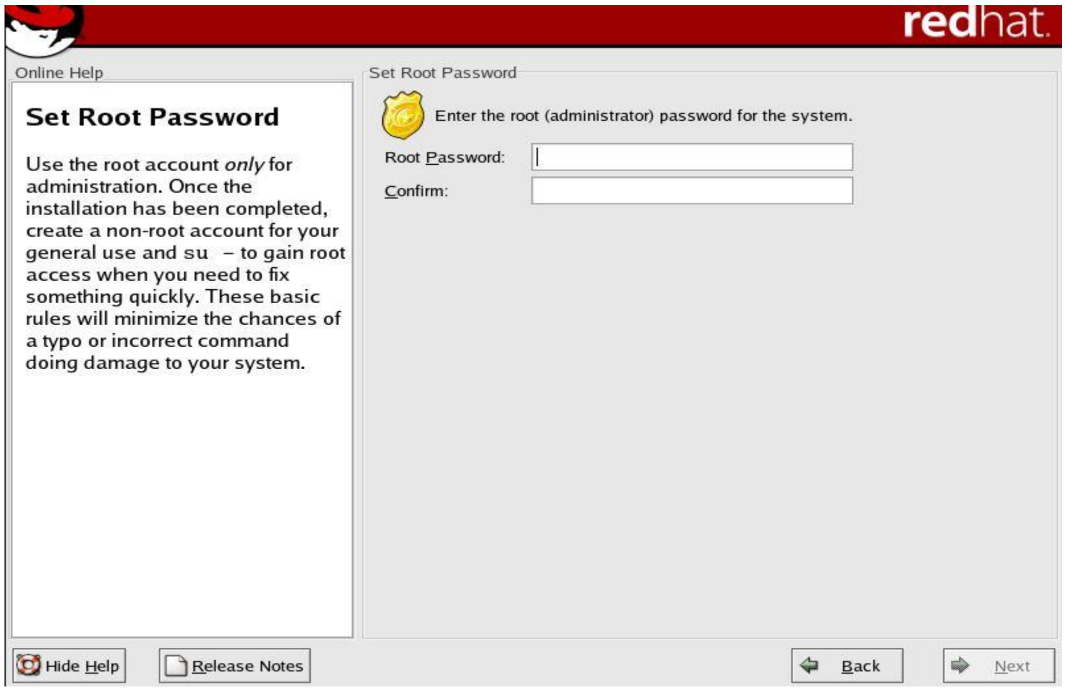
Time Zone Selection

Pacific/Chatham - Chatham Islands

Location	Description
America/Nassau	
America/New_York	Eastern Time
America/Nipigon	Eastern Time - Ontario & Quebec - places that did

☐ System clock uses UTC

Set Root Password



Set Root Password

Use the root account *only* for administration. Once the installation has been completed, create a non-root account for your general use and su – to gain root access when you need to fix something quickly. These basic rules will minimize the chances of a typo or incorrect command doing damage to your system.

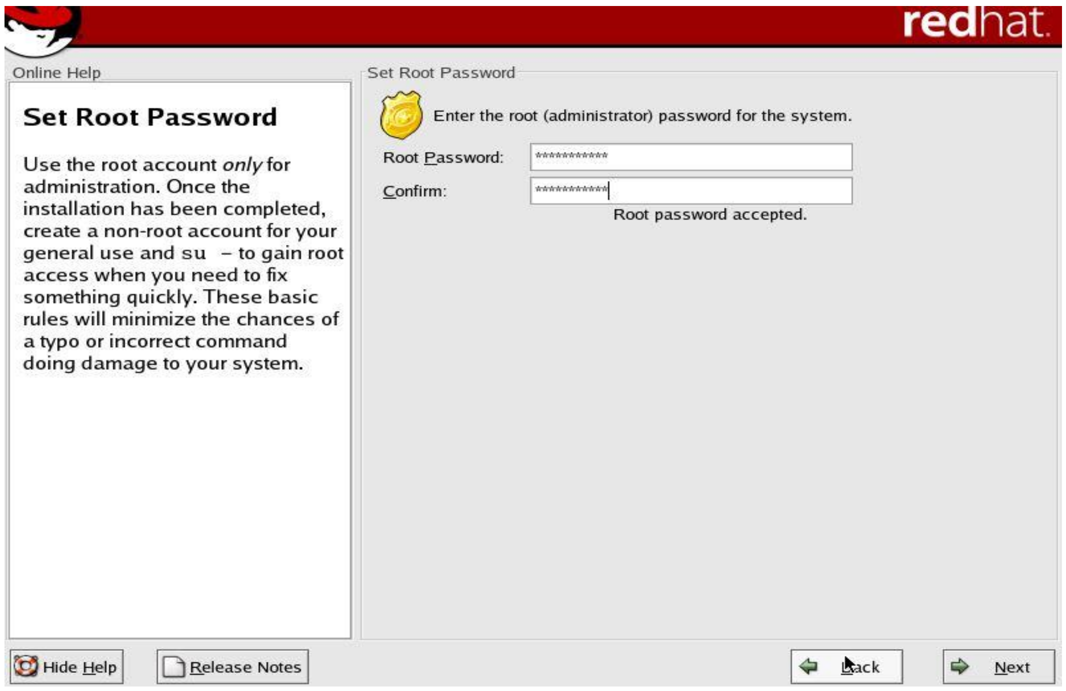
Set Root Password

Enter the root (administrator) password for the system.

Root Password:

Confirm:

Provide Root Password



Set Root Password

Use the root account *only* for administration. Once the installation has been completed, create a non-root account for your general use and su – to gain root access when you need to fix something quickly. These basic rules will minimize the chances of a typo or incorrect command doing damage to your system.

Set Root Password

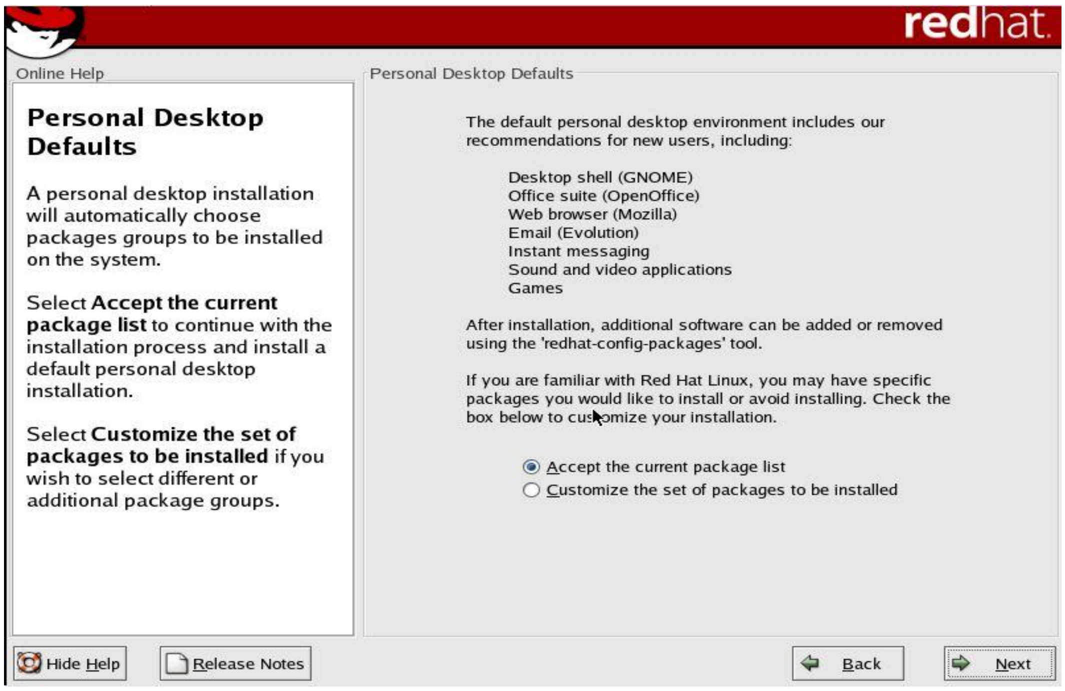
Enter the root (administrator) password for the system.

Root Password: ************

Confirm: ************

Root password accepted.

Personal Desktop Defaults (Packages)



Online Help

Personal Desktop Defaults

A personal desktop installation will automatically choose packages groups to be installed on the system.

Select **Accept the current package list** to continue with the installation process and install a default personal desktop installation.

Select **Customize the set of packages to be installed** if you wish to select different or additional package groups.

Personal Desktop Defaults

The default personal desktop environment includes our recommendations for new users, including:

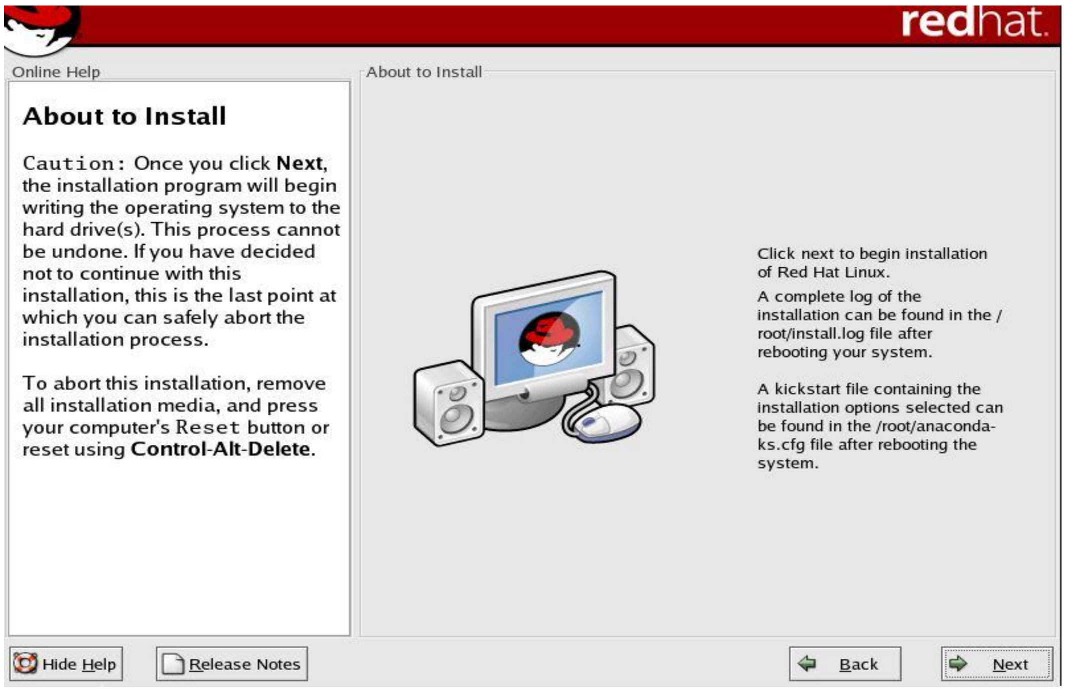
- Desktop shell (GNOME)
- Office suite (OpenOffice)
- Web browser (Mozilla)
- Email (Evolution)
- Instant messaging
- Sound and video applications
- Games

After installation, additional software can be added or removed using the 'redhat-config-packages' tool.

If you are familiar with Red Hat Linux, you may have specific packages you would like to install or avoid installing. Check the box below to customize your installation.

[✓ Accept the current package list]
Customize the set of packages to be installed

About to Install



Online Help

About to Install

Caution: Once you click **Next**, the installation program will begin writing the operating system to the hard drive(s). This process cannot be undone. If you have decided not to continue with this installation, this is the last point at which you can safely abort the installation process.

To abort this installation, remove all installation media, and press your computer's Reset button or reset using **Control-Alt-Delete**.

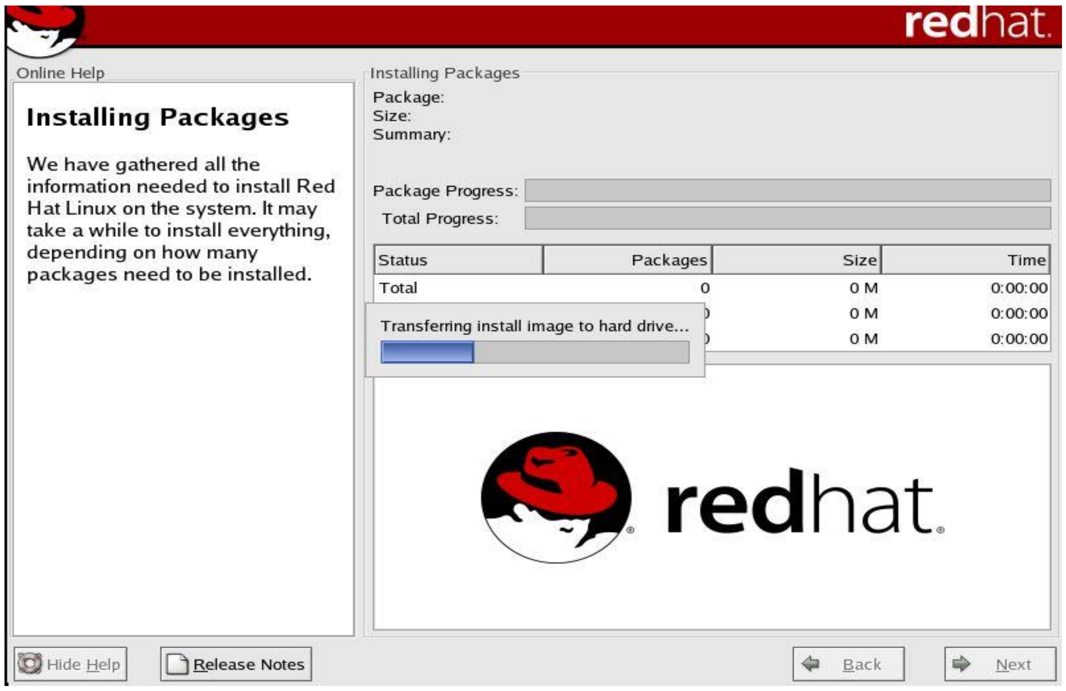
About to Install

Click next to begin installation of Red Hat Linux.

A complete log of the installation can be found in the /root/install.log file after rebooting your system.

A kickstart file containing the installation options selected can be found in the /root/anaconda-ks.cfg file after rebooting the system.

Installing Packages



Online Help

Installing Packages

We have gathered all the information needed to install Red Hat Linux on the system. It may take a while to install everything, depending on how many packages need to be installed.

Installing Packages

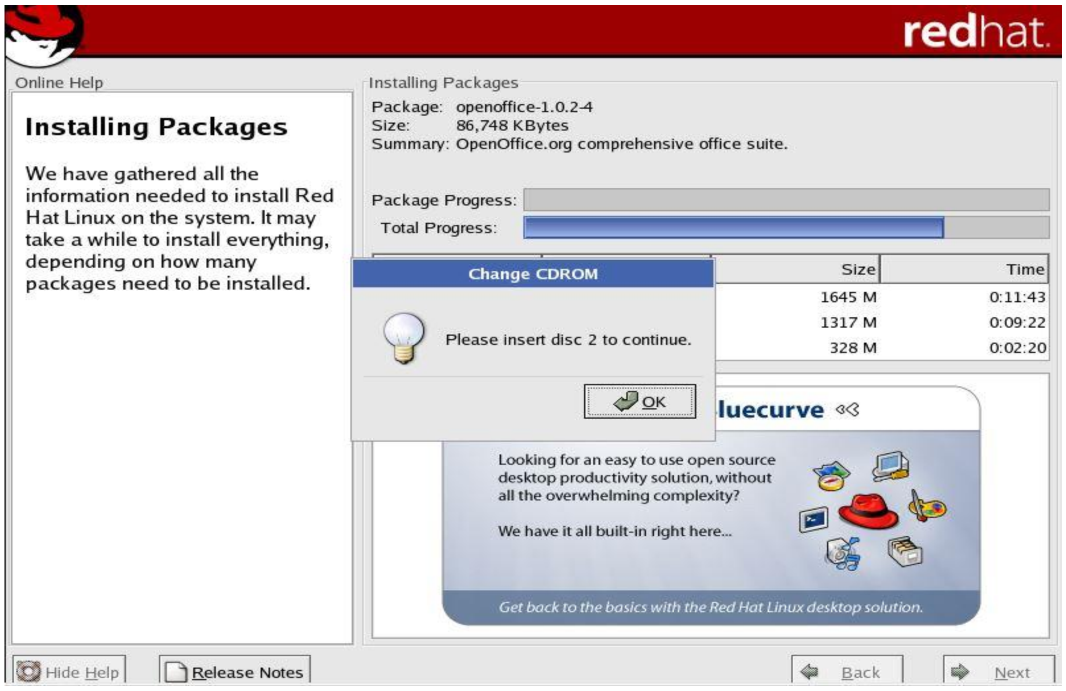
Package:
Size:
Summary:

Package Progress:
Total Progress:

Status	Packages	Size	Time
Total	0	0 M	0:00:00
		0 M	0:00:00
		0 M	0:00:00

Transferring install image to hard drive...

Insert Disc 2



Online Help

Installing Packages

We have gathered all the information needed to install Red Hat Linux on the system. It may take a while to install everything, depending on how many packages need to be installed.

Installing Packages

Package: openoffice-1.0.2-4
Size: 86,748 KBytes
Summary: OpenOffice.org comprehensive office suite.

Package Progress:
Total Progress:

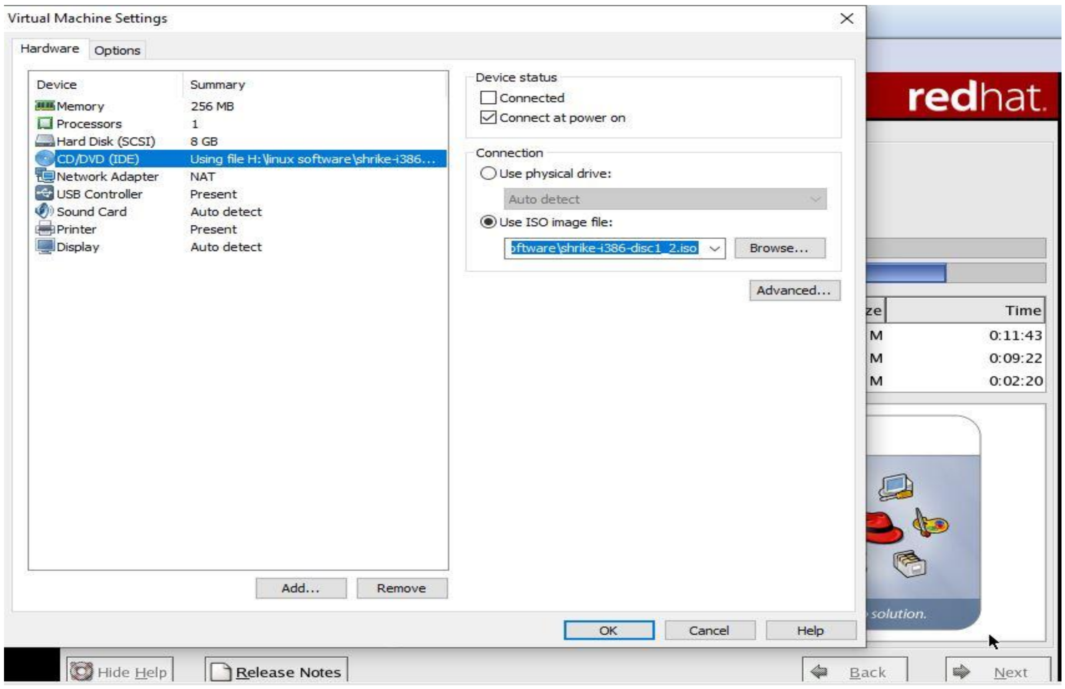
Size	Time
1645 M	0:11:43
1317 M	0:09:22
328 M	0:02:20

Change CDROM

Please insert disc 2 to continue.

OK

luecurve
Looking for an easy to use open source desktop productivity solution, without all the overwhelming complexity?
We have it all built-in right here...
Get back to the basics with the Red Hat Linux desktop solution.
redhat
Hide Help
Release Notes
Back
Next



Virtual Machine Settings

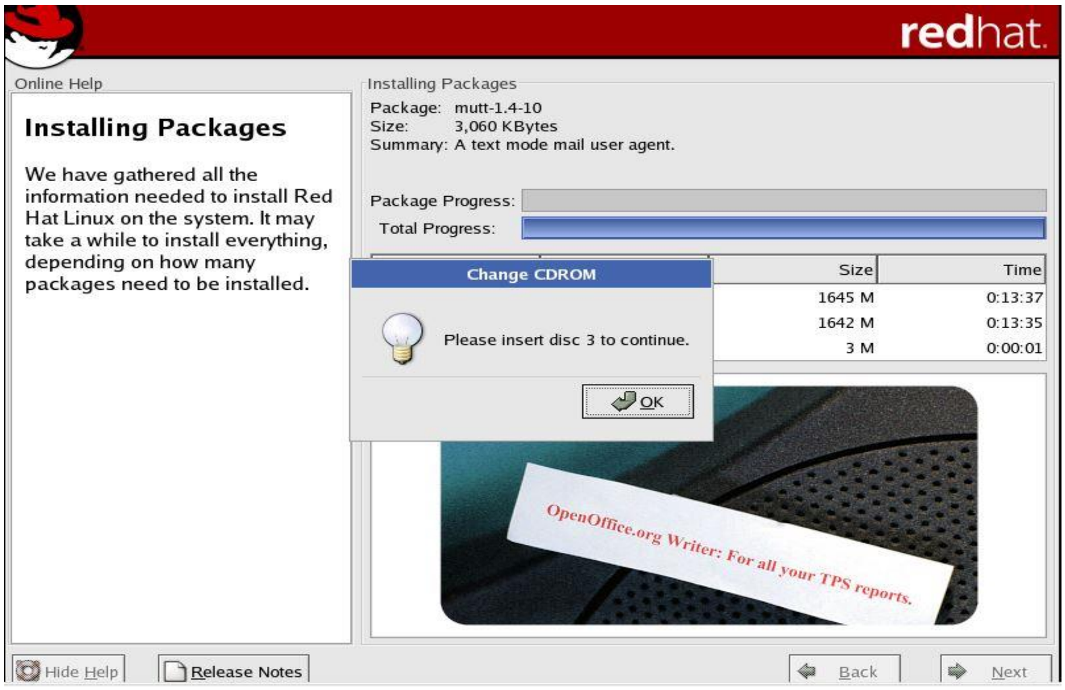
Hardware | Options

Device	Summary
Memory	256 MB
Processors	1
Hard Disk (SCSI)	8 GB
CD/DVD (IDE)	Using file H:\linux software\shrike-i386...
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Device status
☐ Connected
☑ Connect at power on

Connection
○ Use physical drive: Auto detect
● Use ISO image file: oftware\shrike-i386-disc1_2.iso

Insert Disc 3



Online Help

Installing Packages

We have gathered all the information needed to install Red Hat Linux on the system. It may take a while to install everything, depending on how many packages need to be installed.

Installing Packages
Package: mutt-1.4-10
Size: 3,060 KBytes
Summary: A text mode mail user agent.

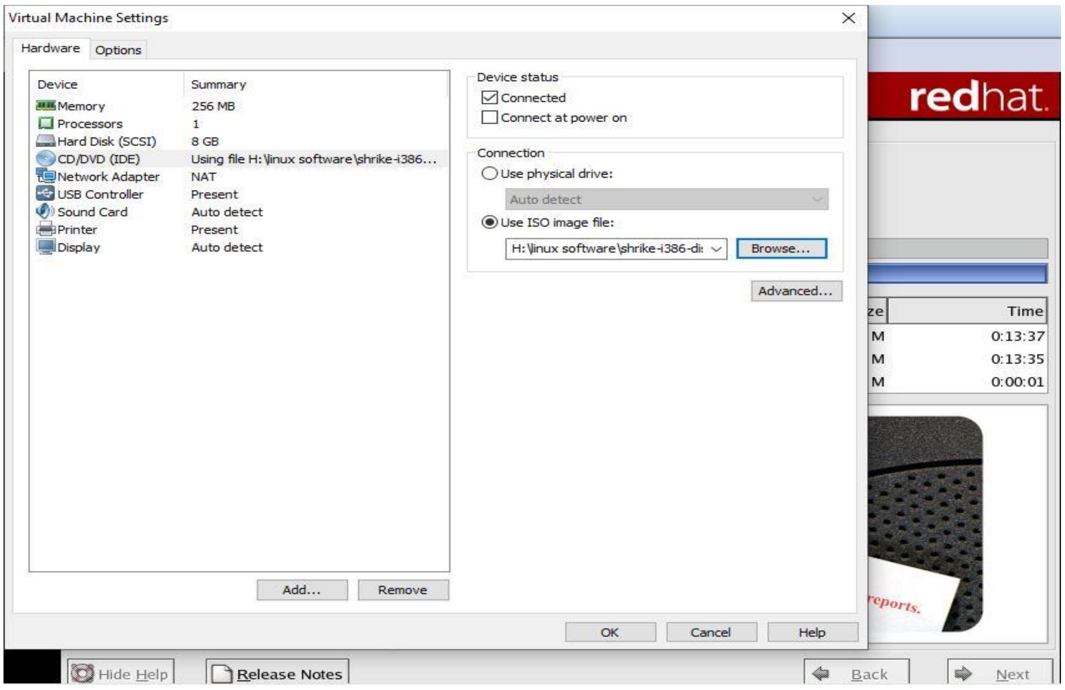
Package Progress:
Total Progress:

Size	Time
1645 M	0:13:37
1642 M	0:13:35
3 M	0:00:01

Change CDROM

Please insert disc 3 to continue.

Proper Settings



Virtual Machine Settings

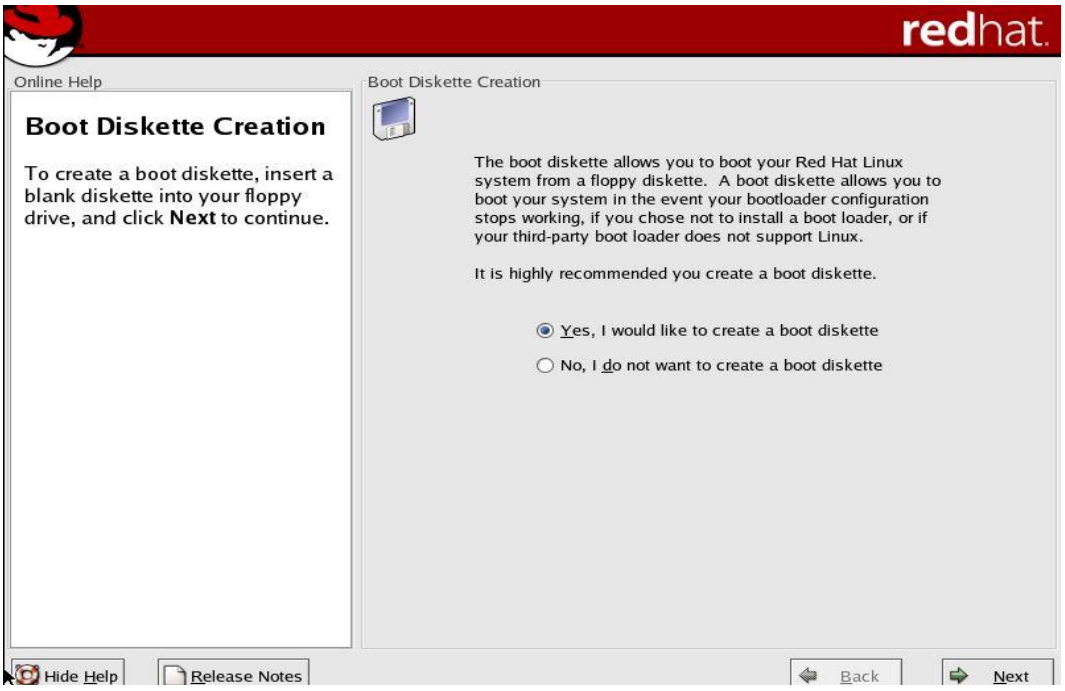
Hardware | Options

Device	Summary
Memory	256 MB
Processors	1
Hard Disk (SCSI)	8 GB
CD/DVD (IDE)	Using file H:\linux software\shrike-i386...
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Device status
[✓ Connected]
Connect at power on

Connection
Use physical drive: Auto detect
[✓ Use ISO image file:] H:\linux software\shrike-i386-di:

Boot Diskette Creation



Online Help

Boot Diskette Creation

To create a boot diskette, insert a blank diskette into your floppy drive, and click **Next** to continue.

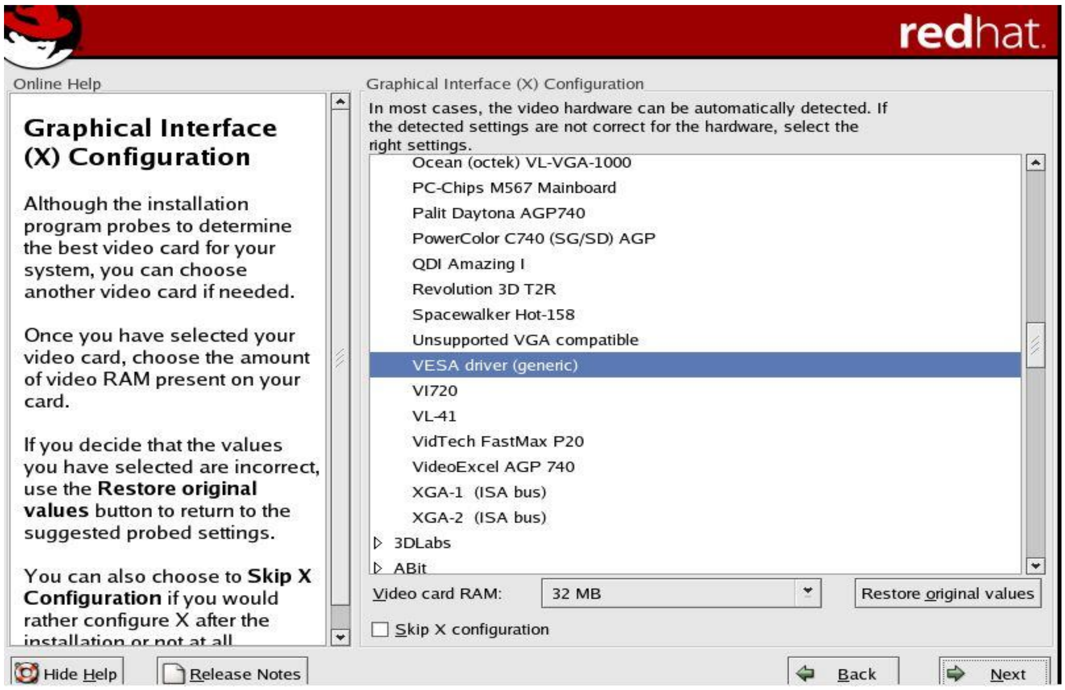
Boot Diskette Creation

The boot diskette allows you to boot your Red Hat Linux system from a floppy diskette. A boot diskette allows you to boot your system in the event your bootloader configuration stops working, if you chose not to install a boot loader, or if your third-party boot loader does not support Linux.

It is highly recommended you create a boot diskette.

[✓ Yes, I would like to create a boot diskette]
No, I do not want to create a boot diskette

Graphical Interface (X) Configuration



Online Help

Graphical Interface (X) Configuration

Although the installation program probes to determine the best video card for your system, you can choose another video card if needed.

Once you have selected your video card, choose the amount of video RAM present on your card.

If you decide that the values you have selected are incorrect, use the **Restore original values** button to return to the suggested probed settings.

You can also choose to **Skip X Configuration** if you would rather configure X after the installation or not at all

Graphical Interface (X) Configuration

In most cases, the video hardware can be automatically detected. If the detected settings are not correct for the hardware, select the right settings.

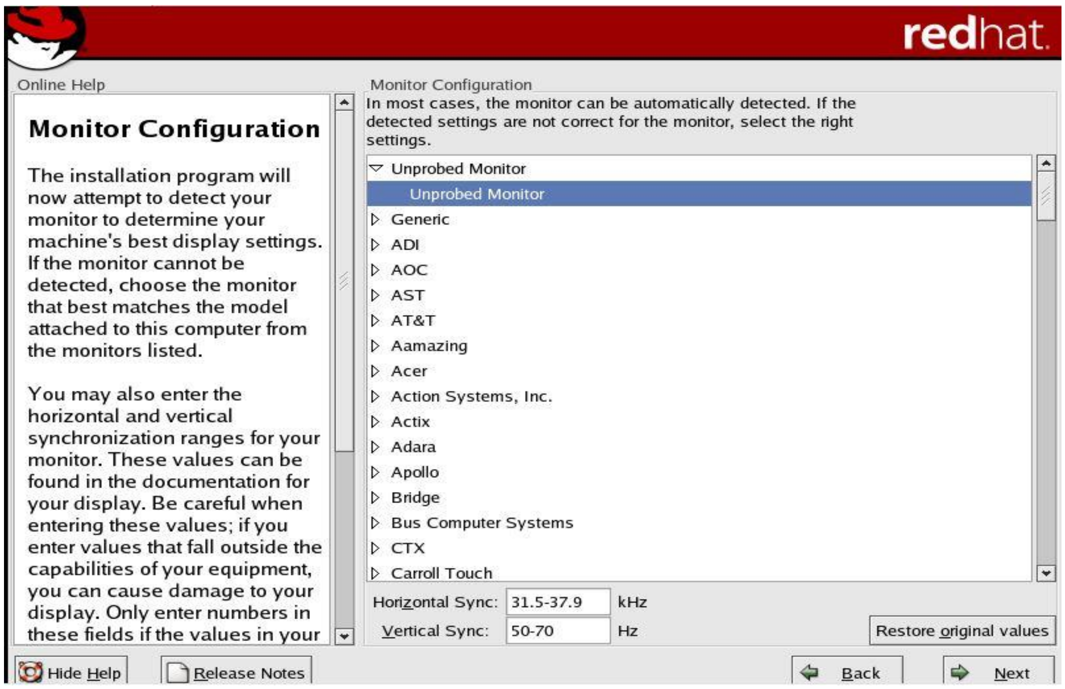
Ocean (octek) VL-VGA-1000
PC-Chips M567 Mainboard
Palit Daytona AGP740
PowerColor C740 (SG/SD) AGP
QDI Amazing I
Revolution 3D T2R
Spacewalker Hot-158
Unsupported VGA compatible
VESA driver (generic)
VI720
VL-41
VidTech FastMax P20
VideoExcel AGP 740
XGA-1 (ISA bus)
XGA-2 (ISA bus)
▷ 3DLabs
▷ ABit

Video card RAM: 32 MB

Restore original values

☐ Skip X configuration

Monitor Configuration



Online Help

Monitor Configuration

The installation program will now attempt to detect your monitor to determine your machine's best display settings. If the monitor cannot be detected, choose the monitor that best matches the model attached to this computer from the monitors listed.

You may also enter the horizontal and vertical synchronization ranges for your monitor. These values can be found in the documentation for your display. Be careful when entering these values; if you enter values that fall outside the capabilities of your equipment, you can cause damage to your display. Only enter numbers in these fields if the values in your

Monitor Configuration

In most cases, the monitor can be automatically detected. If the detected settings are not correct for the monitor, select the right settings.

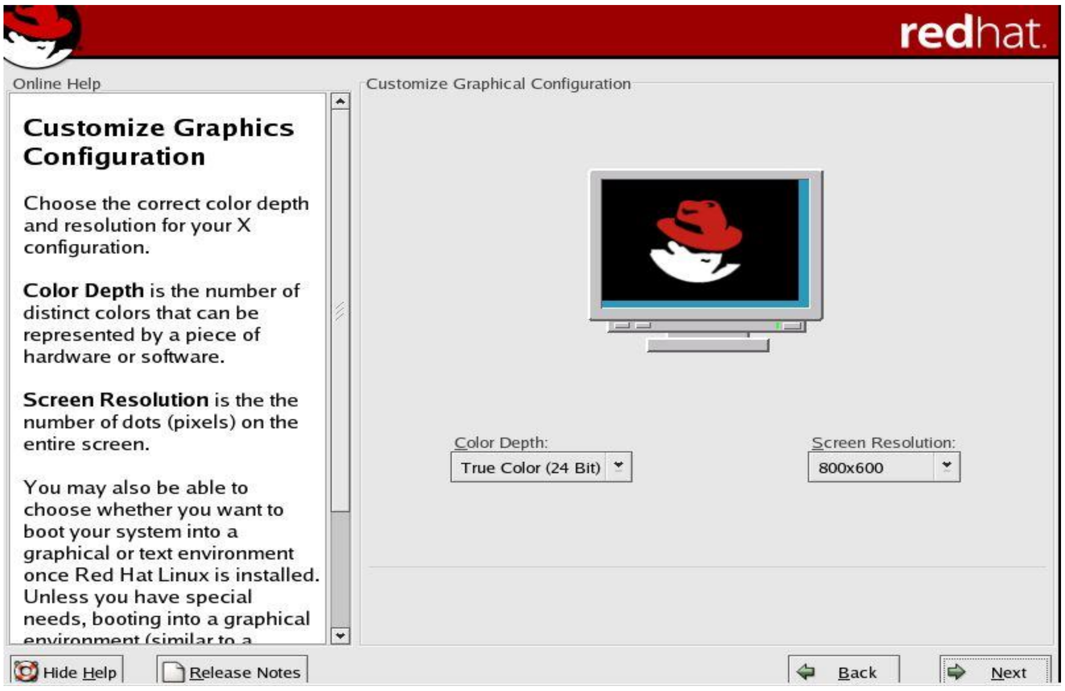
▽ Unprobed Monitor
Unprobed Monitor
▷ Generic
▷ ADI
▷ AOC
▷ AST
▷ AT&T
▷ Aamazing
▷ Acer
▷ Action Systems, Inc.
▷ Actix
▷ Adara
▷ Apollo
▷ Bridge
▷ Bus Computer Systems
▷ CTX
▷ Carroll Touch

Horizontal Sync: 31.5-37.9 kHz

Vertical Sync: 50-70 Hz

Restore original values

Customize Graphics Configuration



Customize Graphics Configuration

Choose the correct color depth and resolution for your X configuration.

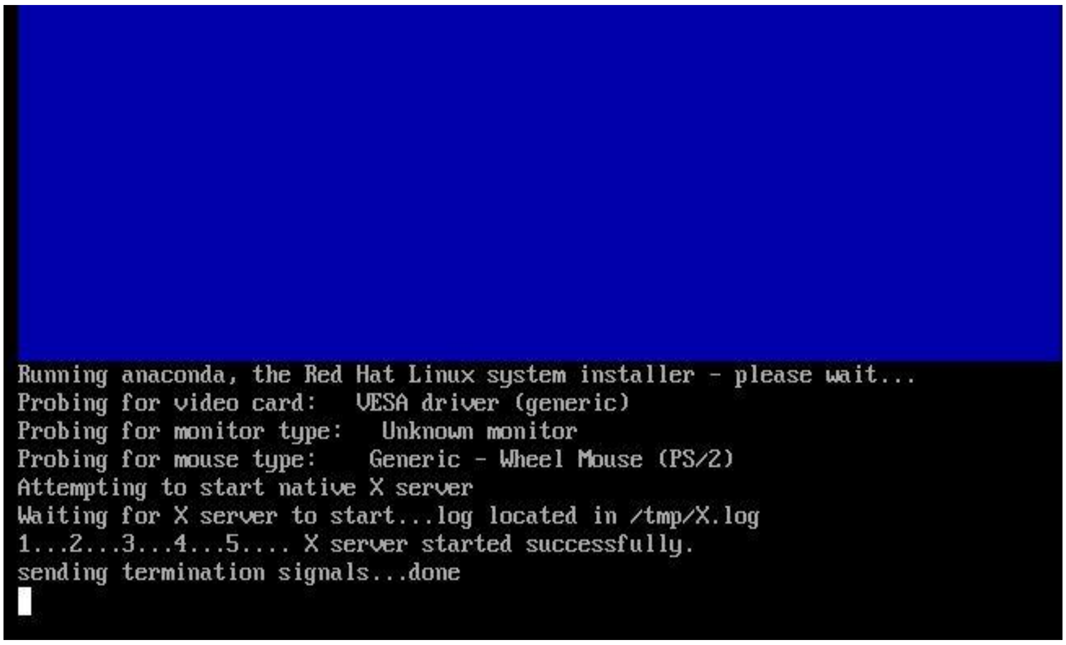
Color Depth is the number of distinct colors that can be represented by a piece of hardware or software.

Screen Resolution is the the number of dots (pixels) on the entire screen.

You may also be able to choose whether you want to boot your system into a graphical or text environment once Red Hat Linux is installed. Unless you have special needs, booting into a graphical environment (similar to a

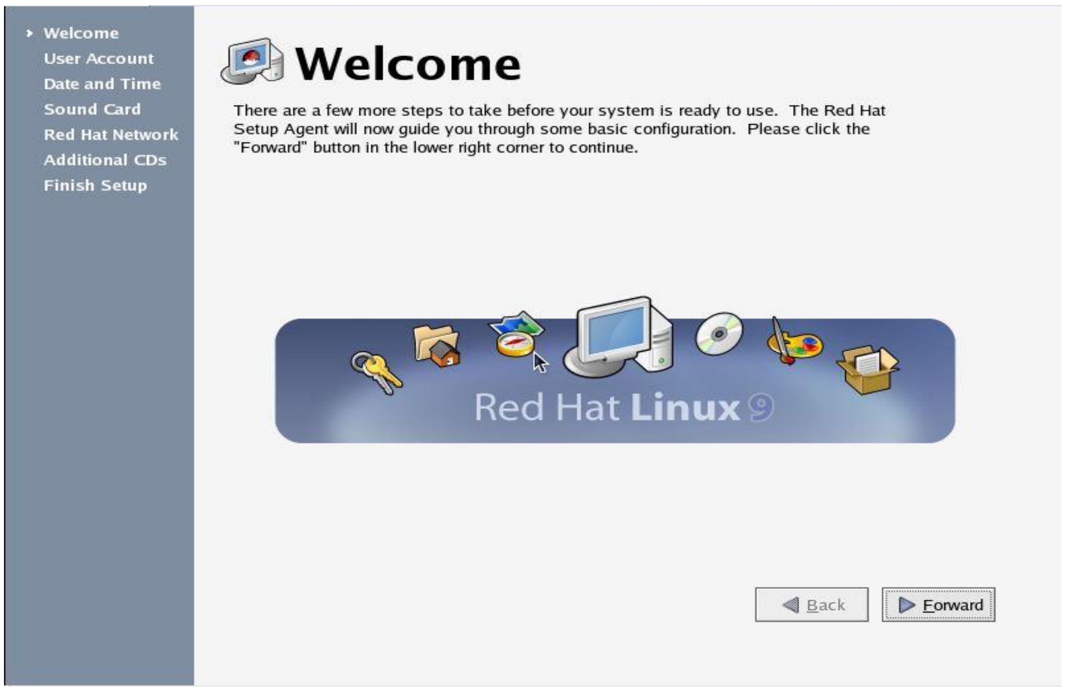
Color Depth: True Color (24 Bit)

Screen Resolution: 800x600



```
Running anaconda, the Red Hat Linux system installer - please wait...
Probing for video card:   VESA driver (generic)
Probing for monitor type:   Unknown monitor
Probing for mouse type:   Generic - Wheel Mouse (PS/2)
Attempting to start native X server
Waiting for X server to start...log located in /tmp/X.log
1...2...3...4...5.... X server started successfully.
sending termination signals...done
█
```

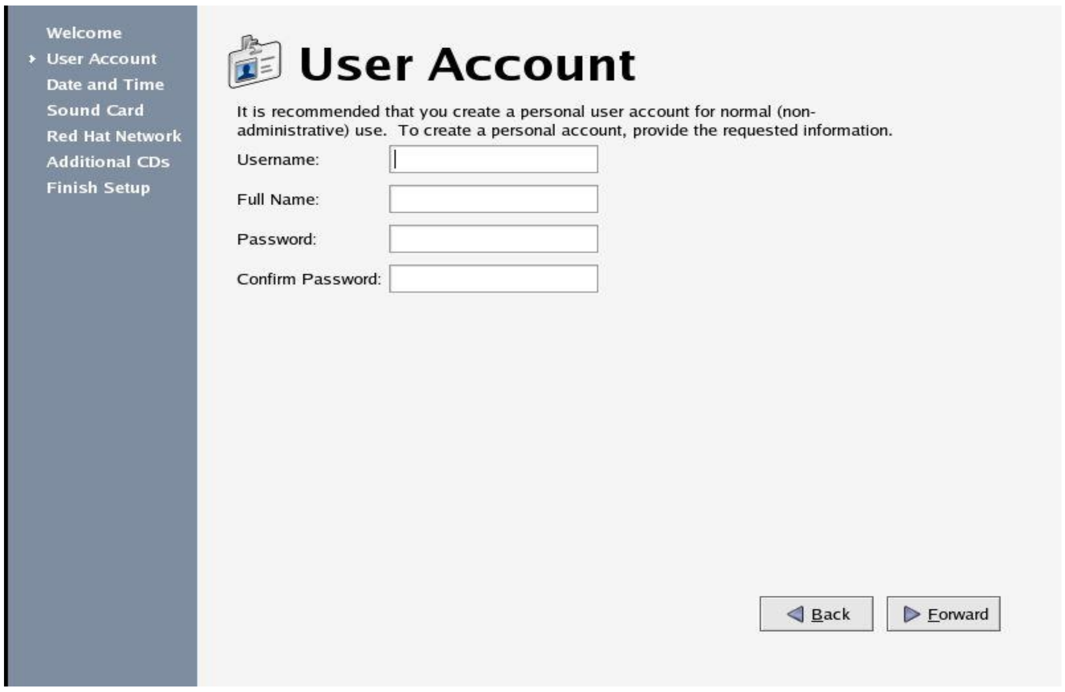
Welcome Screen



Welcome

There are a few more steps to take before your system is ready to use. The Red Hat Setup Agent will now guide you through some basic configuration. Please click the "Forward" button in the lower right corner to continue.

Creation of User Account

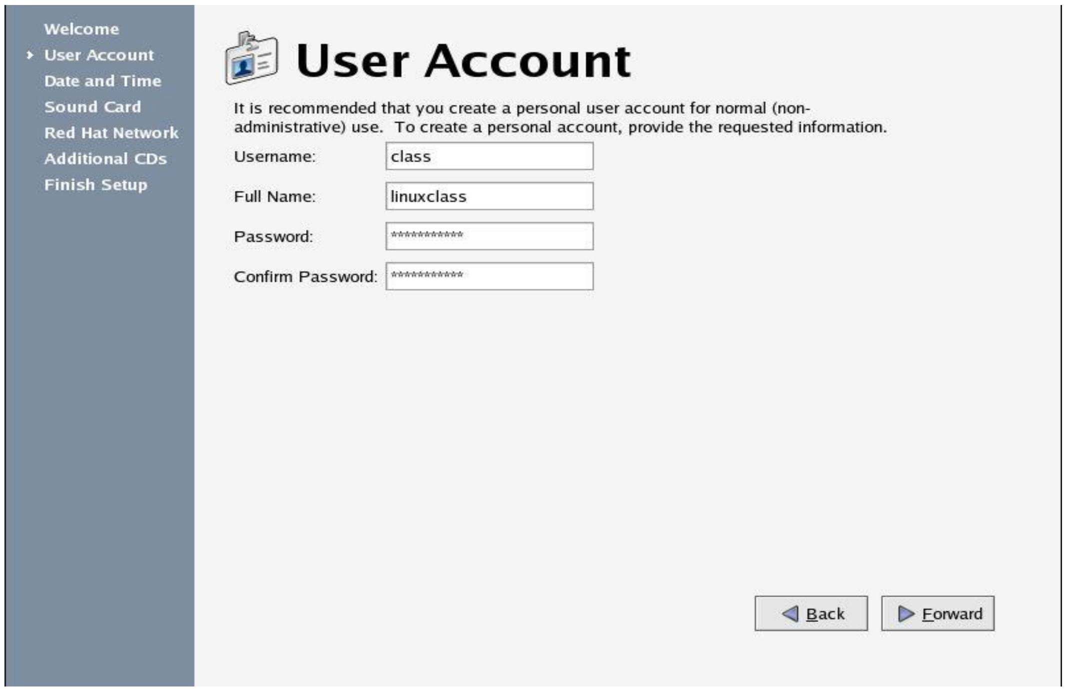


User Account

It is recommended that you create a personal user account for normal (non-administrative) use. To create a personal account, provide the requested information.

Username:	
Full Name:	
Password:	
Confirm Password:	

Enter Credentials

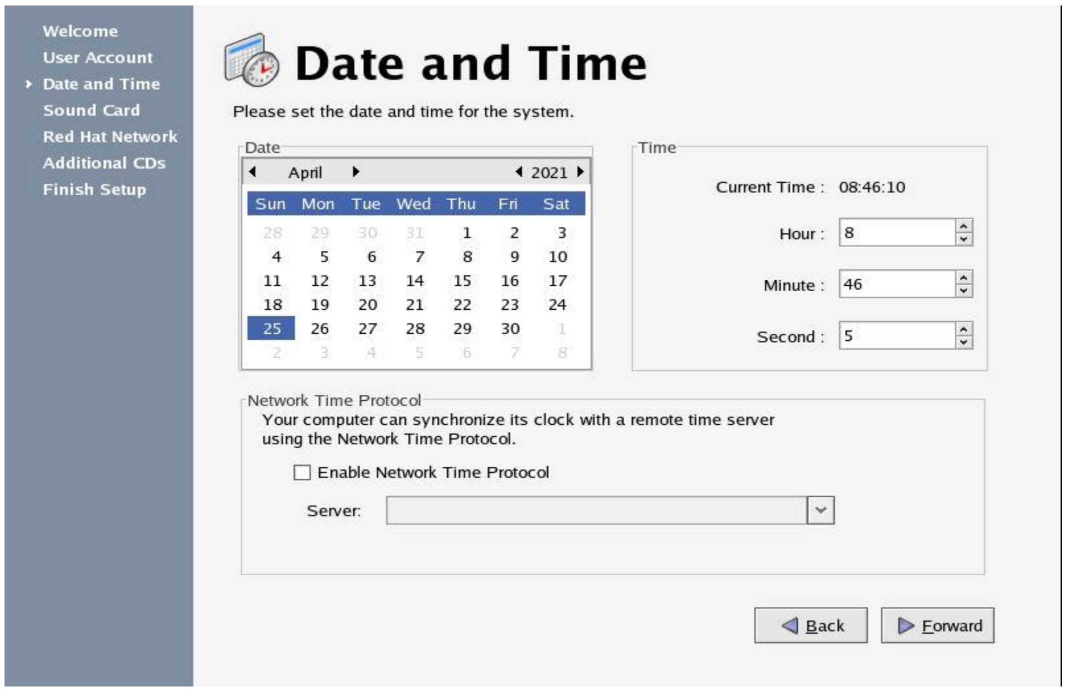


User Account

It is recommended that you create a personal user account for normal (non-administrative) use. To create a personal account, provide the requested information.

Username:	class
Full Name:	linuxclass
Password:	************
Confirm Password:	************

Enter Date and Time



Date and Time

Please set the date and time for the system.

Date

April 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

Time

Current Time :	08:46:10
Hour :	8
Minute :	46
Second :	5

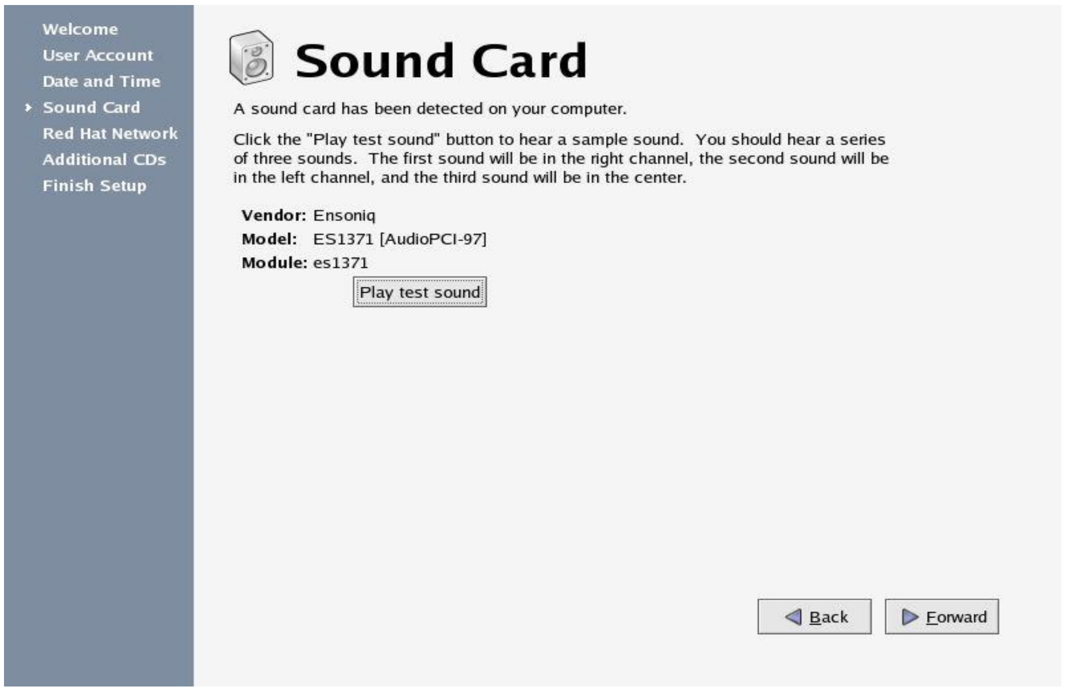
Network Time Protocol

Your computer can synchronize its clock with a remote time server using the Network Time Protocol.

☐ Enable Network Time Protocol

Server:

Sound Card



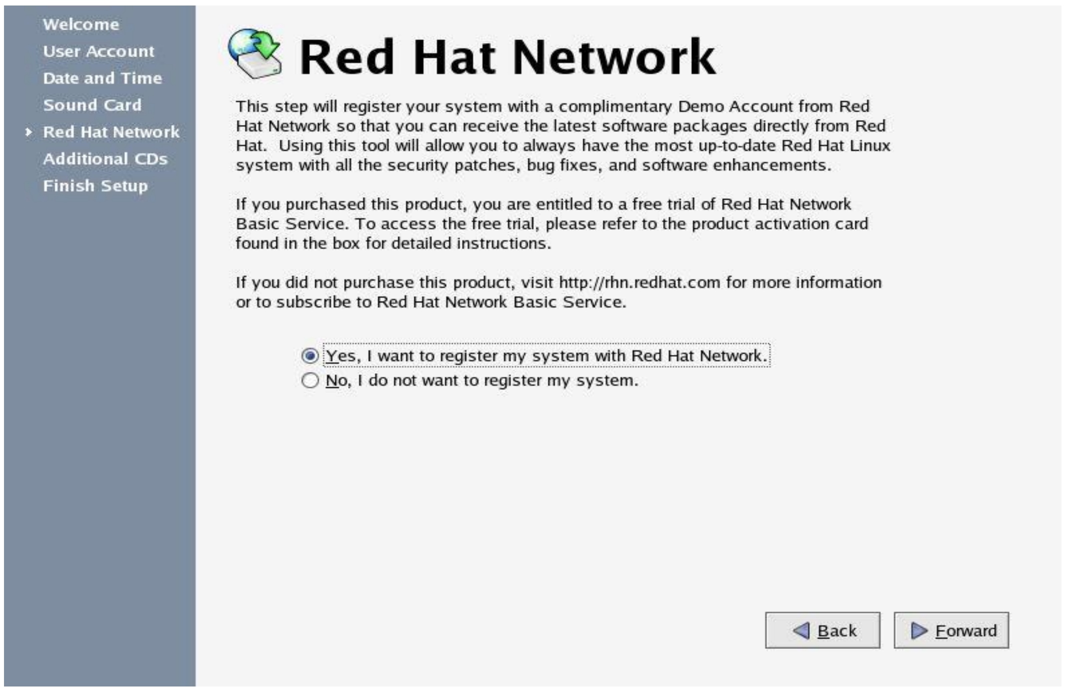
Sound Card

A sound card has been detected on your computer.

Click the "Play test sound" button to hear a sample sound. You should hear a series of three sounds. The first sound will be in the right channel, the second sound will be in the left channel, and the third sound will be in the center.

Vendor: Ensoniq
Model: ES1371 [AudioPCI-97]
Module: es1371

Registration with Red Hat Network



Red Hat Network

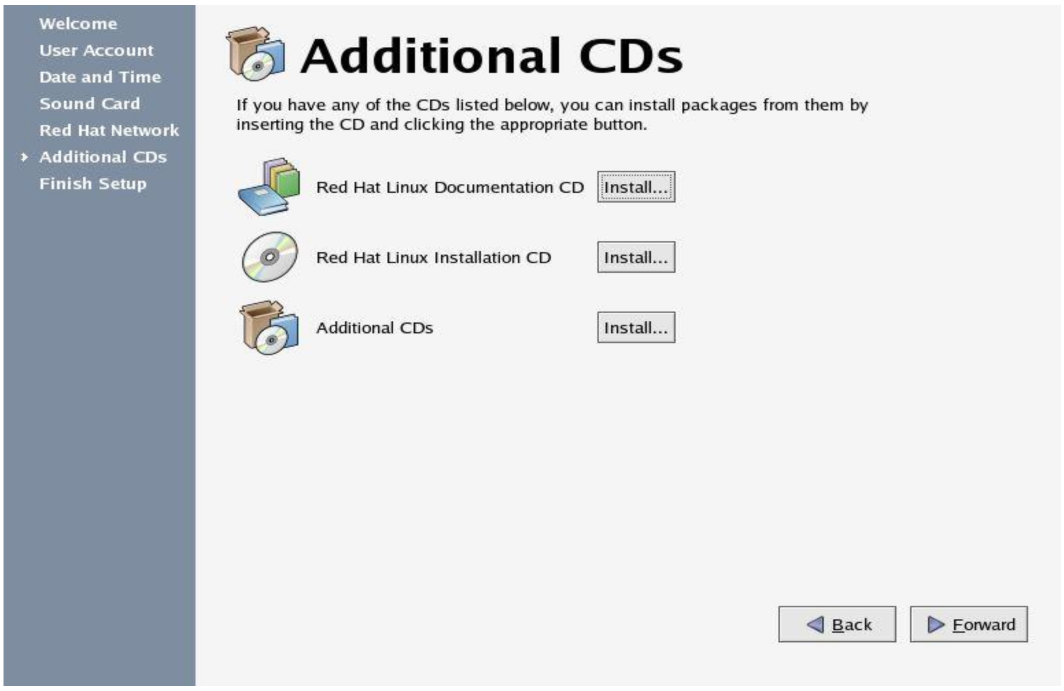
This step will register your system with a complimentary Demo Account from Red Hat Network so that you can receive the latest software packages directly from Red Hat. Using this tool will allow you to always have the most up-to-date Red Hat Linux system with all the security patches, bug fixes, and software enhancements.

If you purchased this product, you are entitled to a free trial of Red Hat Network Basic Service. To access the free trial, please refer to the product activation card found in the box for detailed instructions.

If you did not purchase this product, visit http://rhn.redhat.com for more information or to subscribe to Red Hat Network Basic Service.

[✓ Yes, I want to register my system with Red Hat Network.]
No, I do not want to register my system.

Installation of packages from additional CDs



Additional CDs

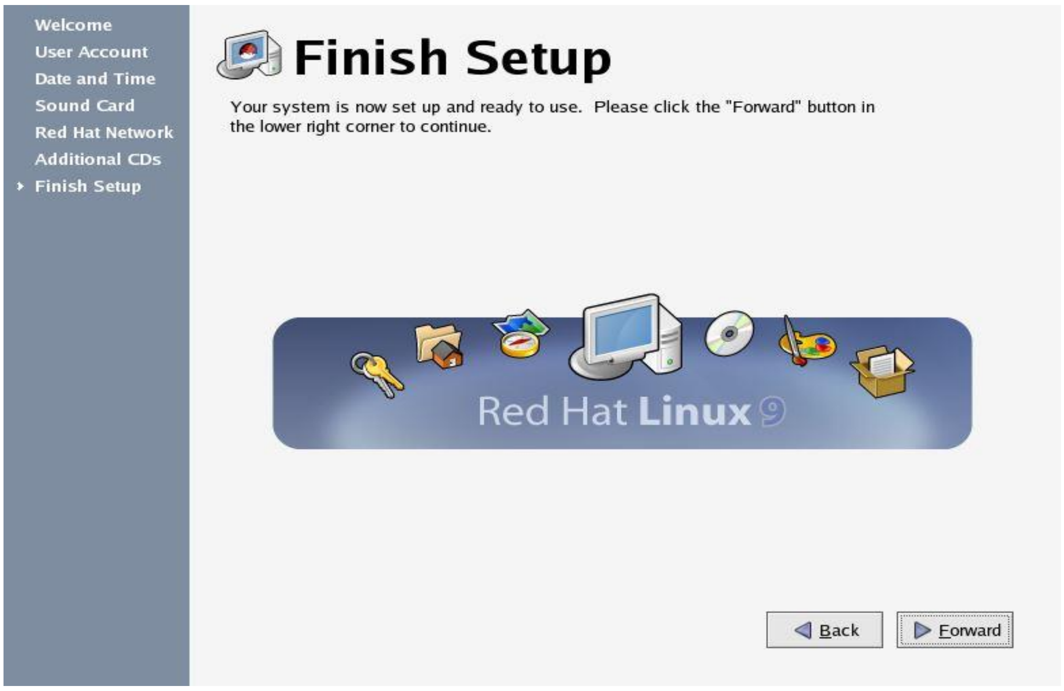
If you have any of the CDs listed below, you can install packages from them by inserting the CD and clicking the appropriate button.

Red Hat Linux Documentation CD Install...

Red Hat Linux Installation CD Install...

Additional CDs Install...

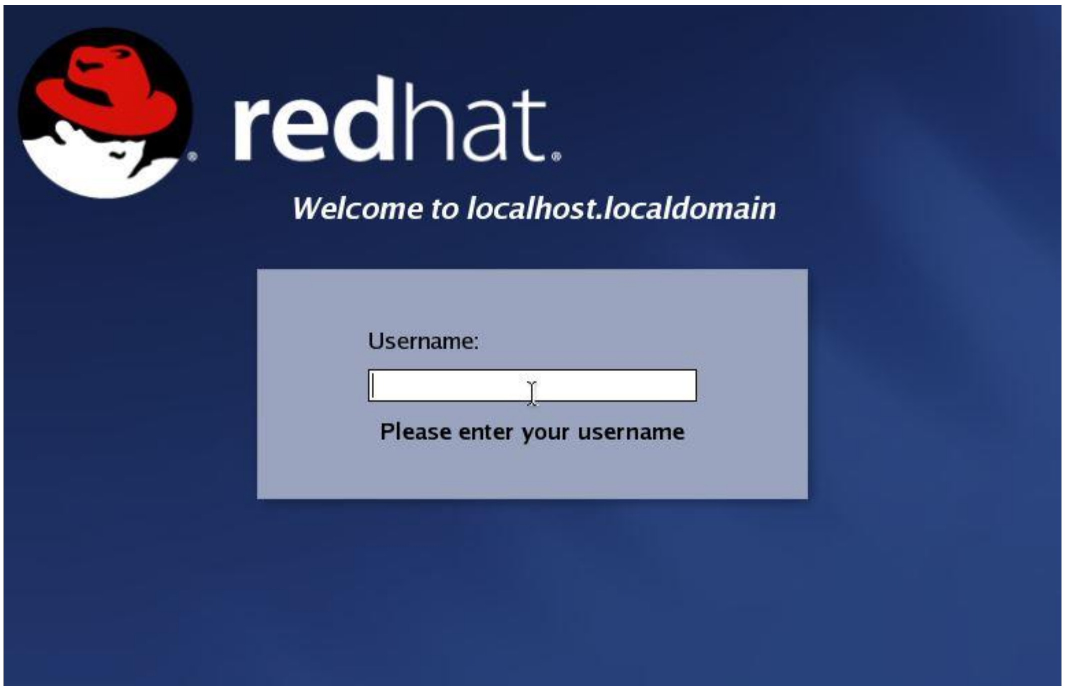
Finish Setup



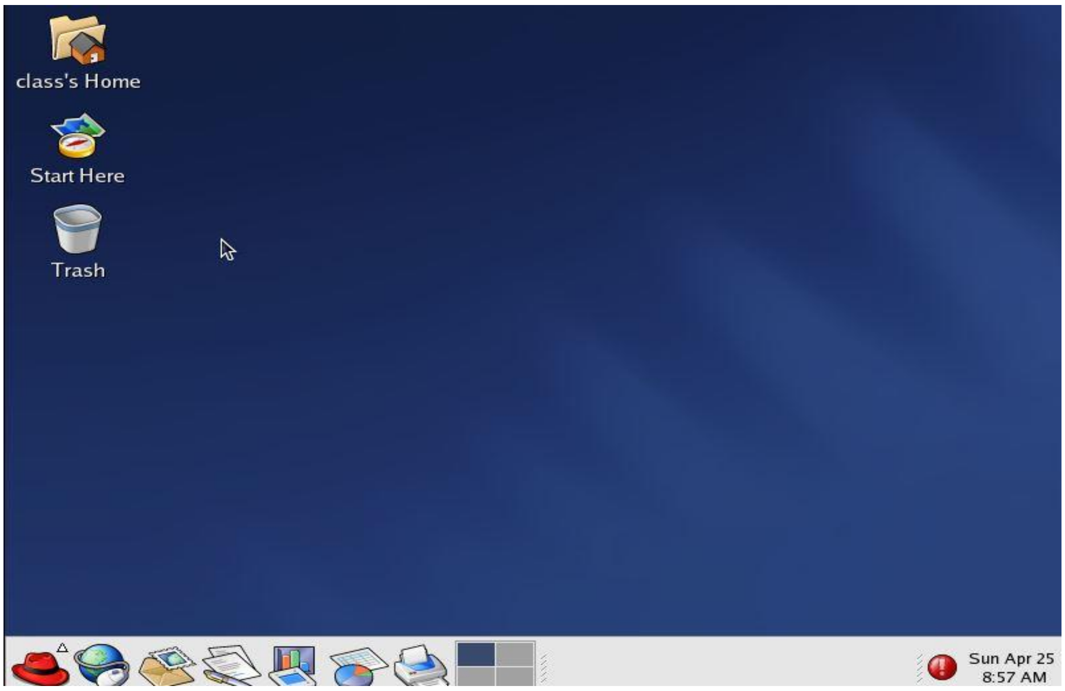
Finish Setup

Your system is now set up and ready to use. Please click the "Forward" button in the lower right corner to continue.

Logging in



Home Screen



2.4 Moving around the Desktop

- **GNOME**: The default desktop interface of Red Hat Linux 9. GNOME represents a presentation layer that provides a graphical user interface as well as the focused working environment, which enables you to access all your work from one place.
- **KDE**: KDE desktop is included in Rec Hat Linux 9 distribution but not installed by default.KDE is a desktop environment for an integrated set of cross-platform applications designed to run on Linux, FreeBSD, Microsoft Windows, Solaris and Mac OS, designed by the KDE Community.

2.5 Components of Desktop

- Panel: The panel stretches across the bottom of the desktop. By default, it contains the main menu icon and quick-launch icons for logging out, opening a terminal window, and other common applications and utilities. The panel is highly configurable. You can add and remove buttons that launch applications easily.

- Workspace: Workspaces refer to the grouping of windows on your desktop. You can create multiple workspaces, which act like virtual desktops. Workspaces are meant to reduce clutter and make the desktop easier to navigate.Workspaces can be used to organize your work. For example, you could have all your communication windows, such as e-mail and your chat program, on one workspace, and the work you are doing on a different workspace. Your music manager could be on a third workspace.

Summary:

- A boot sequence is the set of operations the computer performs when it is switched on that load an operating system.
- The primary function of BIOS is code program embedded on a chip that recognizes and controls various devices that make up the computer.
- GRUB and LILO are the most popular Linux boot loader.
- The first thing the kernel does is to execute init program.
- There are four types of installation available in Linux: personal desktop, workstation, server and custom.
- There are two ways the disk can be partitioned: manually and automatically.

Keywords:

- Booting: It is a bootstrapping process that starts operating systems when the user turns on a computer system.
- BIOS: It refers to the software code run by a computer when first powered on.
- Partitioning: It is a means to divide a single hard drive into many logical drives.A partition is a contiguous set of blocks on a drive that are treated as an independent disk.
- Swap: Swap partitions are used to support virtual memory. In other words, data is written to a swap partition when there is not enough RAM to store the data your system is processing.
- GNOME: The default desktop interface of Red Hat Linux is GNOME.
- Run-level: A run-level is a software configuration of the system which allows only a selected group of processes to exist

Self Assessment

1. Which of these executes kernel?

A. MBR

B. BIOS

C. GRUB

D. Init

2. The first thing a Kernel does is __________.

A. Execute GRUB

B. Execute LILO

C. Execute init program

D. None of the above

3. Which of these tasks are handled by Kernel?

A. System Call

B. Process Management

C. Device Management

D. All: System call, process and device management

4. While installation of Red Hat Linux in the system, it asks for __________.

A. Language Selection

B. Keyboard Configuration

C. Mouse Configuration

D. All: Language selection, keyboard and mouse configuration

5. In partition field, i.e., SIZE, the measurement unit is _____

A. TB

B. MB

C. GB

D. KB

6. Which of these defines the runlevels?

A. 0-6

B. 1-7

C. 2-8

D. 3-9

7. Which of these partitions is used to support virtual memory?

A. /

B. /var

C. /boot

D. swap

8. Which of these is a Red Hat Linux installer?
A. Anaconda
B. GRUB
C. LILO
D. Emulator

9. Which of these programs allows us to partition the disk?
A. Anaconda
B. GRUB
C. Disk Druid
D. Disk Help

10. MBR is executed by ___________ .
A. BIOS
B. GRUB
C. Kernel
D. Init

11. In which mode, it is possible to install and upgrade Red Hat Linux?
A. Graphical
B. Text
C. Both graphical and text
D. None of the above

12. What does BIOS mean?
A. Basic Input/ Output Service
B. Basic Input/ Output System
C. Buffer Input/ Output System
D. Buffer Input/ Output Service

13. MBR is executed by ___________ .
A. BIOS
B. GRUB
C. Kernel
D. Init

14. In which mode, it is possible to install and upgrade Red Hat Linux?
A. Graphical
B. Text
C. Both graphical and text
D. None of the above

15. What type can be installed in Red Hat Linux?

A. Personal Desktop
B. Server
C. Workstation
D. All: personal desktop, server and workstation

Answers for Self Assessment

1.	C	2.	C	3.	D	4.	D	5.	B
6.	A	7.	D	8.	A	9.	C	10.	A
11.	C	12.	B	13.	A	14.	C	15.	D

Review Questions:

1. What is booting? Explain the booting sequence in detail.
2. What is a kernel? Explain the tasks of a kernel in detail.
3. What is a partition? Write the partition fields. What is the recommended partition scheme?
4. What is a file system? Explain its types in detail.
5. What is a run-level? Explain about the run-level of in it.

Further Readings

Mark G. Sobell, A Practical Guide to Fedora andRed Hat Enterprise Linux, Fifth Edition, Pearson Education, Inc.

Web Links

https://www.educba.com/install-linux/

https://phoenixnap.com/kb/linux-create-partition

Unit 03: Connecting to Internet

Objectives

After studying this unit, you will be able to understand

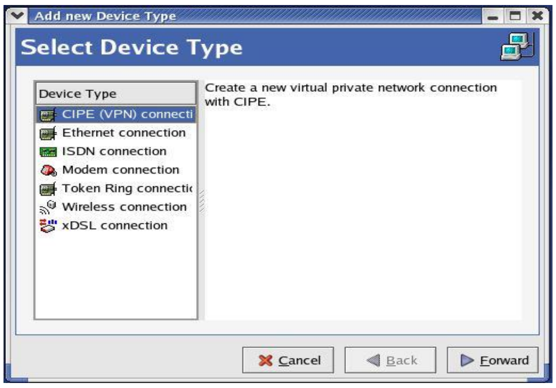
- Understand the Network Interfacing Tool
- Connect to LAN using static and dynamic addresses
- Understand DNS
- Know useful commands for configuration of system for internet
- Understand the internet connectivity in Linux

Introduction

When the installation of operating system is completed on the computer system, the next task we do is to connect it to the internet. The surfing of websites, playing online games, sending, and receiving of emails etc. is possible only through internet. For this, first we need to configure our computer system so that the connection can take place.

3.1 Internet Configuration Wizard

The internet configuration wizard can be opened by following this path: Main Menu | System Tools | Internet Configuration Wizard. Network interfacing tool is also known as the network administration tool or network configuration tool.

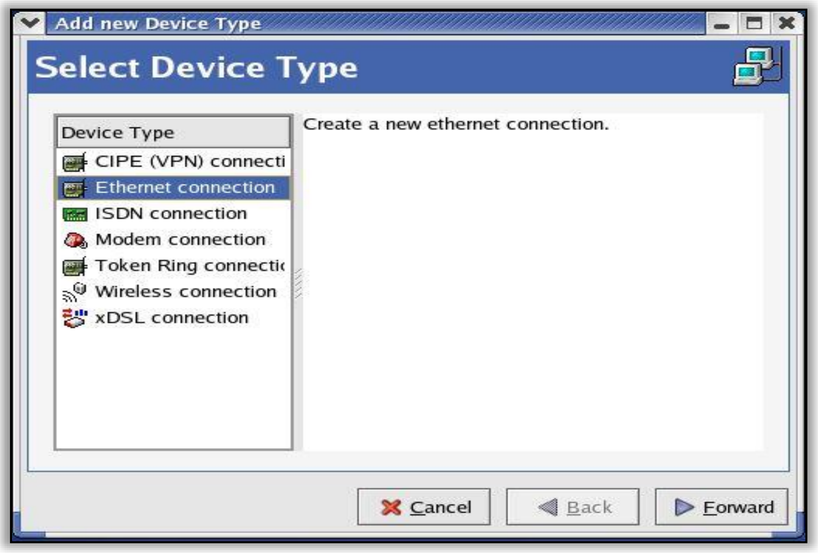


Create a new virtual private network connection with CIPE.

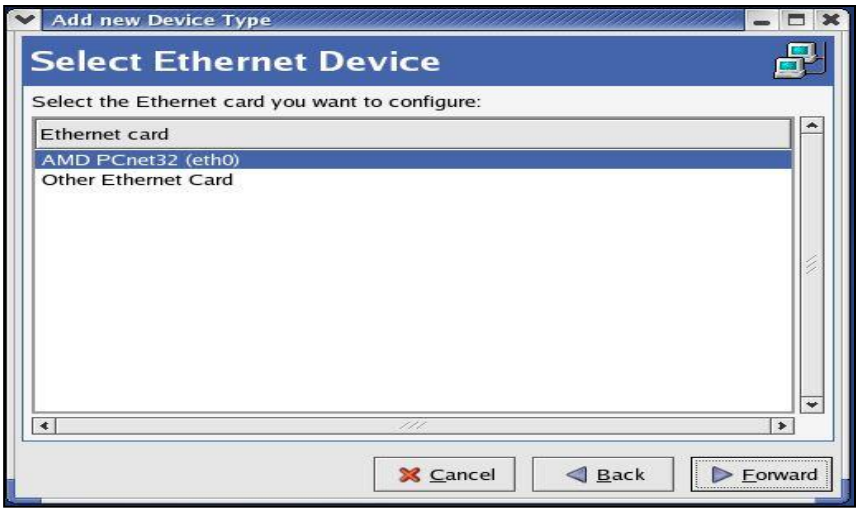
3.2 Connecting to LAN

A local area network (LAN) is a collection of devices connected in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school. A network device connected to a TCP/IP network has an IP address associated with it, such as 192.168.100.20. Using the IP address of a machine, other machines on the network can address it uniquely.A LAN comprises cables, access points, switches, routers, and other components that enable devices to connect to internal servers, web servers, and other LANs via wide area networks.

Once the network interfacing wizard is opened, then we need to select the device type. The available options are VPN connection, ethernet connection, ISDN connection, modem connection, token ring connection, wireless connection and xDSL connection. As we going to connect to LAN, so for that we need to click on 'Ethernet connection', it will create a new ethernet connection. Once selected, click on forward.



Next it will ask for the selection of Ethernet card. The options are AMD PCnet32 and other ethernet cards. Click on AMD PCnet32 (eth0) and then click forward.

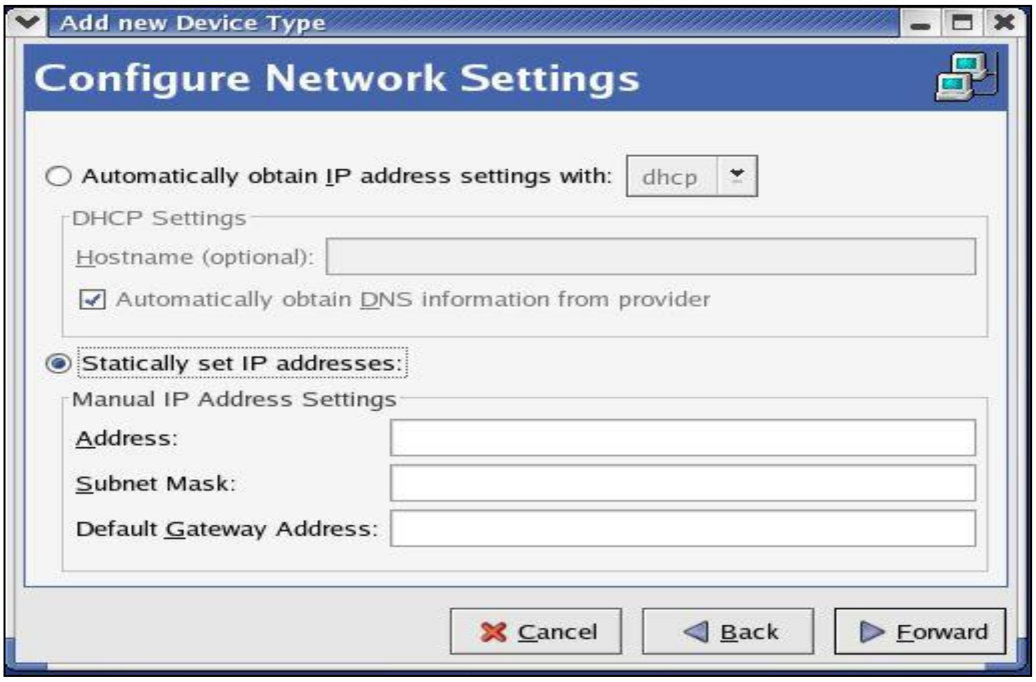


Next it will ask for configuration of network settings, and these networks are provided by the means of IP addresses. An IP address is a unique address that identifies a device on the internet or a local network. IP stands for "Internet Protocol," which is the set of rules governing the format of data sent via the internet or local network. In essence, IP addresses are the identifier that allows information to be sent between devices on a network: they contain location information and make devices accessible for communication. The internet needs a way to differentiate between different computers, routers, and websites. IP addresses provide a way of doing so and form an essential part of how the internet works. These are of two types: static IP address and dynamic IP address.

Static IP address: A static IP address is simply an address that doesn't change. Once your device is assigned a static IP address, that number typically stays the same until the device is decommissioned or your network architecture changes. Static IP addresses generally are used by servers or other important equipment. Static IP addresses are assigned by Internet Service Providers (ISPs). Your ISP may or may not allocate you a static IP address depending on the nature of your service agreement. We describe your options a little later, but for now assume that a static IP address adds to the cost of your ISP contract. A static IP address may be IPv4 or IPv6; in this case the important quality is static. Some day, every bit of networked gear we have might have a unique static IPv6 address. We're not there yet. For now, we usually use static IPv4 addresses for permanent addresses.

Dynamic IP address: As the name suggests, dynamic IP addresses are subject to change, sometimes at a moment's notice. Dynamic addresses are assigned, as needed, by Dynamic Host Configuration Protocol (DHCP) servers. We use dynamic addresses because IPv4 doesn't provide enough static IP addresses to go around. So, for example, a hotel probably has a static IP address, but each individual device within its rooms would have a dynamic IP address. On the internet, your home or office may be assigned a dynamic IP address by your ISP's DHCP server. Within your home or business network, the dynamic IP address for your devices -- whether they are personal computers, Smartphone, streaming media devices, tablet, what have you -- are probably assigned by your network router. Dynamic IP is the standard used by and for consumer equipment.

The next step is to select static or dynamic IP address. So when you go further, the next screen will show this.


Add new Device Type

Configure Network Settings

( ) Automatically obtain IP address settings with: dhcp

DHCP Settings

Hostname (optional):

[✓] Automatically obtain DNS information from provider

[● Statically set IP addresses:]

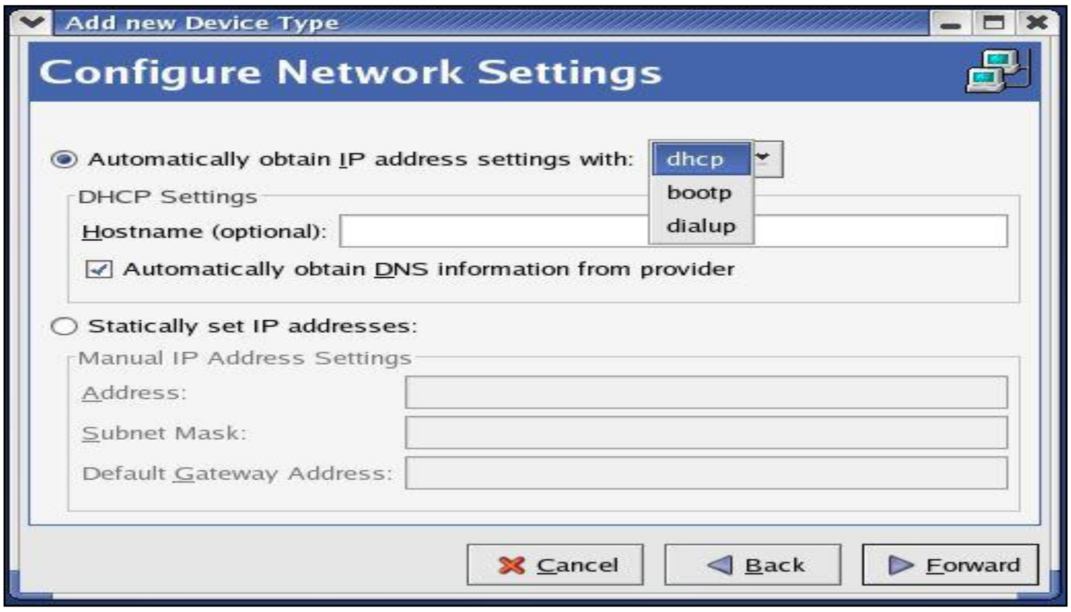
Manual IP Address Settings

Address:

Subnet Mask:

Default Gateway Address:

Here we need to select the dynamic IP address. Dynamic IP addresses are distributed and managed via dynamic address allocation protocols. In the case of a machine connected to a LAN using a dynamic IP address, the address is allocated either using the DHCP or BOOTP. For ISPs that use PPPoE, the address is allotted by the PPPoE protocol, in which case we need to choose dialup.


Add new Device Type

Configure Network Settings

[✓ Automatically obtain IP address settings with: dhcp] (bootp, dialup)

DHCP Settings

Hostname (optional):

[✓] Automatically obtain DNS information from provider

Statically set IP addresses:

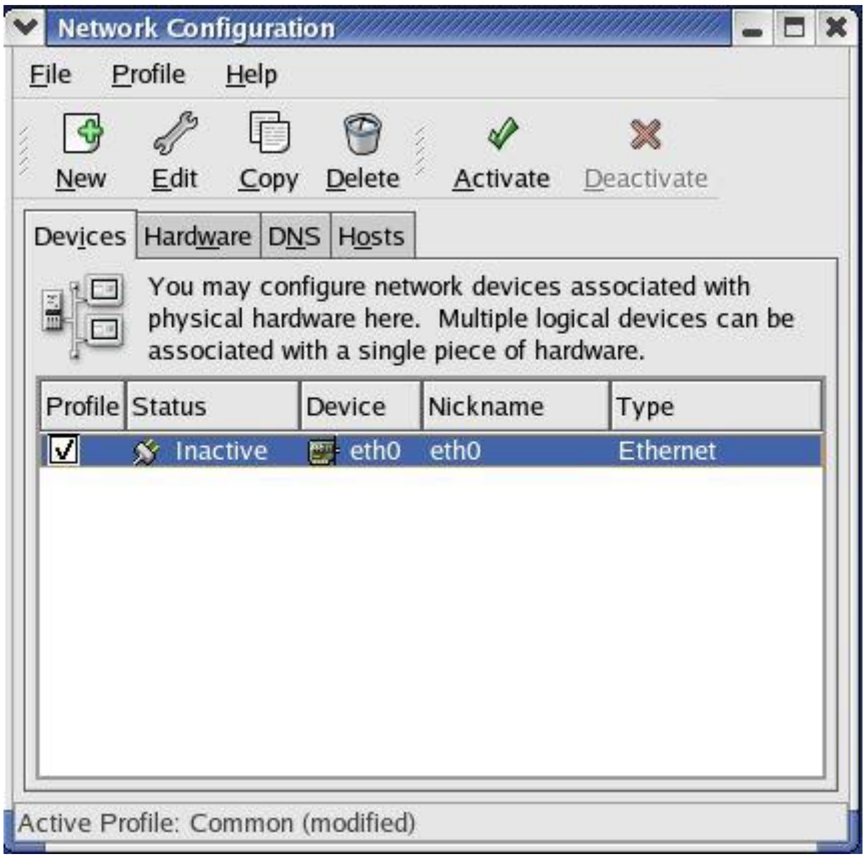
Manual IP Address Settings

Address:

Subnet Mask:

Default Gateway Address:

When everything is done, it will create an Ethernet device.


Network Configuration

You may configure network devices associated with physical hardware here. Multiple logical devices can be associated with a single piece of hardware.

Profile	Status	Device	Nickname	Type
✓	Inactive	eth0	eth0	Ethernet

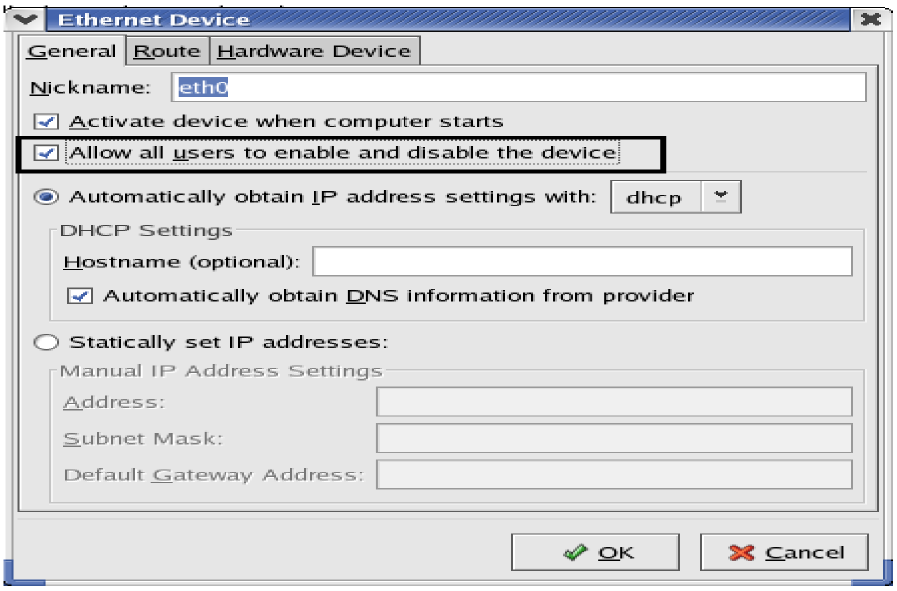
Active Profile: Common (modified)

The network interfacing tool will be opened. It has four tabs:

- Devices Tab: Lists the device connections that we have available on our machine.
- Hardware Tab: Allows us to manage the various network devices on the system, such as Ethernet cards, internal modems, and wireless cards.
- DNS Tab: Allows us to specify DNS server information.

- Hosts Tab: Allows us to modify the hostname of the machine and add aliases to the same host.

When you double click on the selected device, the next screen will look like this.



Ethernet Device

General | Route | Hardware Device

Nickname: eth0

[✓] Activate device when computer starts

[✓] Allow all users to enable and disable the device

(●) Automatically obtain IP address settings with: dhcp

DHCP Settings

Hostname (optional):

[✓] Automatically obtain DNS information from provider

( ) Statically set IP addresses:

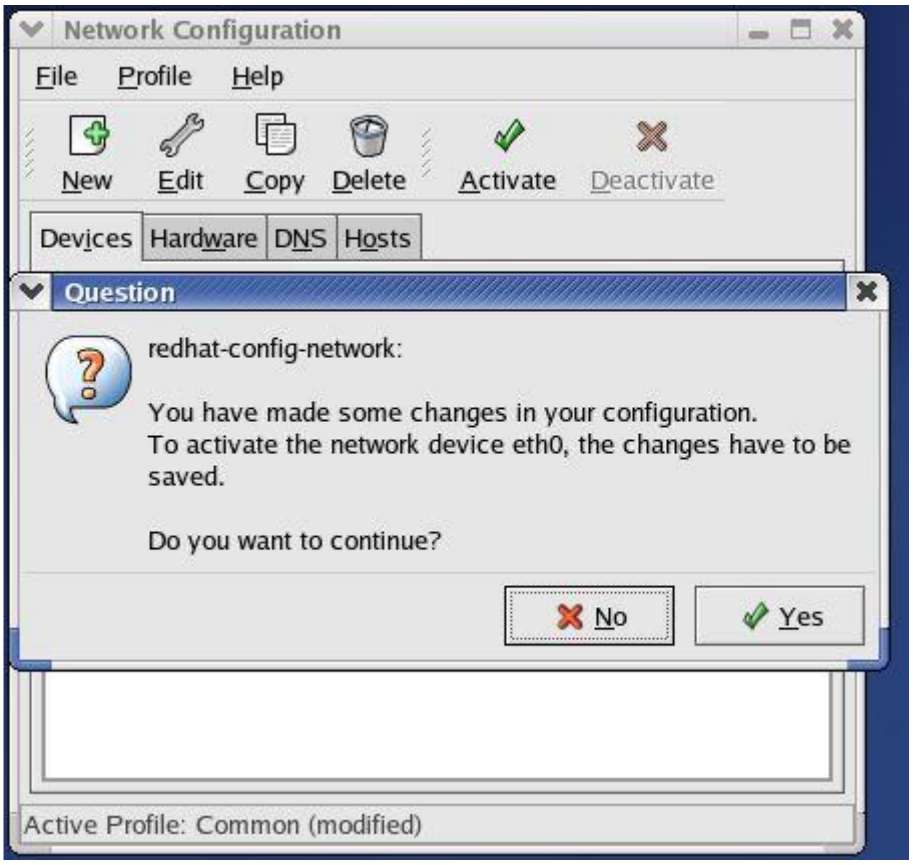
Manual IP Address Settings

Address:

Subnet Mask:

Default Gateway Address:

Click on allow all users to enable and disable the device. The next screen is:



Question

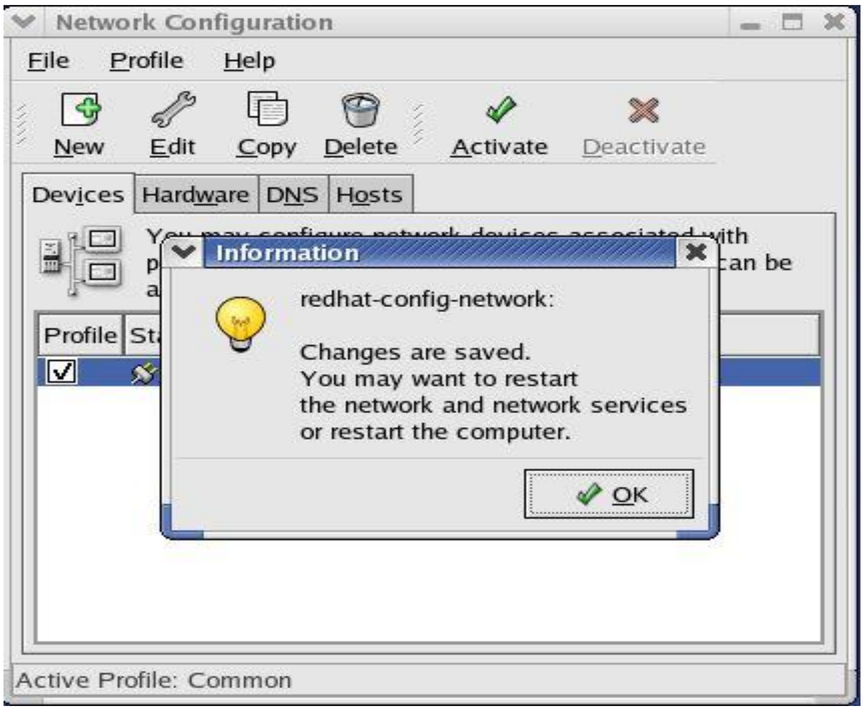
redhat-config-network:

You have made some changes in your configuration.
To activate the network device eth0, the changes have to be saved.

Do you want to continue?

Active Profile: Common (modified)

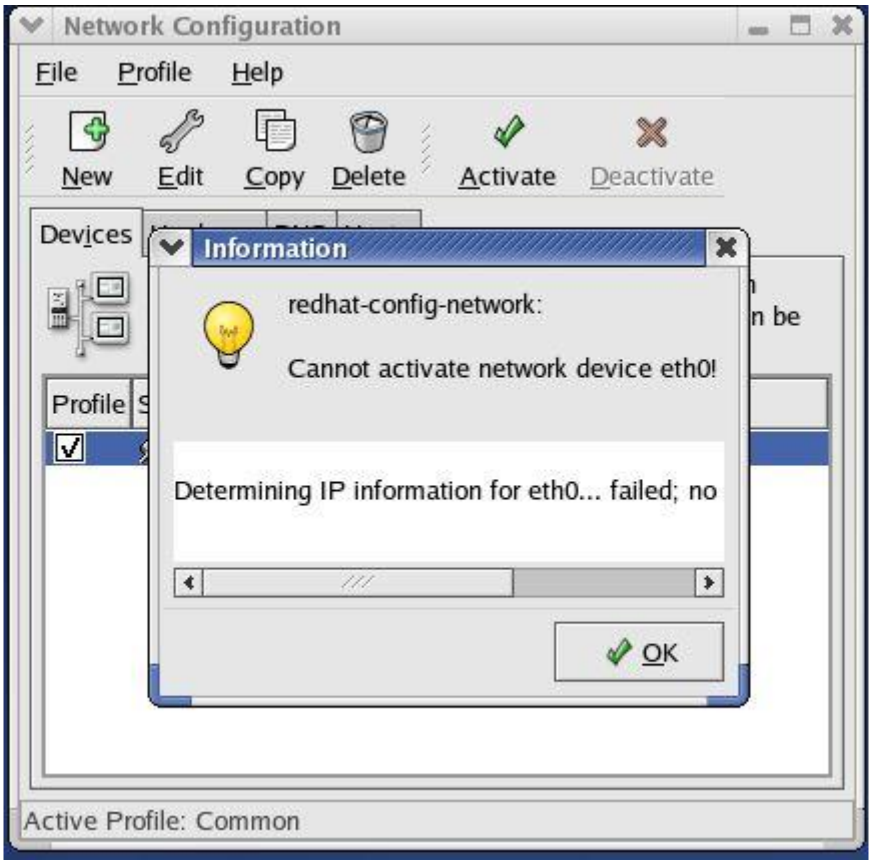
Click on OK.



redhat-config-network:

Changes are saved.
You may want to restart
the network and network services
or restart the computer.

To activate the internet connection, click on ok.

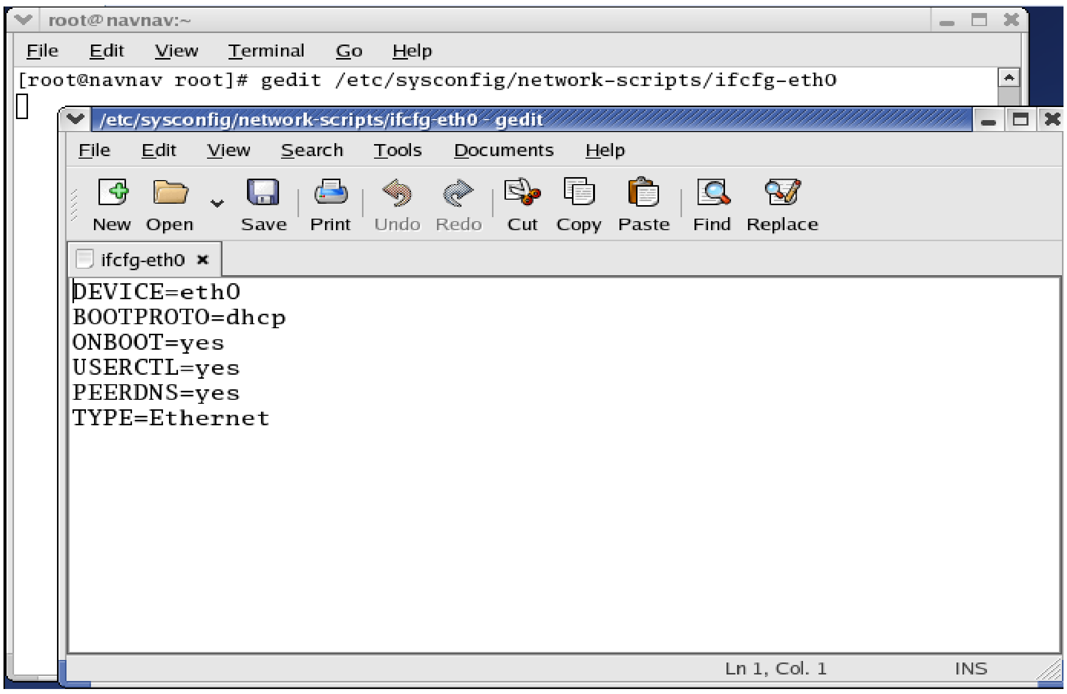


redhat-config-network:

Cannot activate network device eth0!

```
Determining IP information for eth0... failed; no
```

When this is done, our next task is to modify three files so that the computer system can be connected to the internet. These files are opened using the editors. For the opening of first file, we need to write #gedit /etc/syscon fig/network-scripts/ifcfg-eth0 in terminal and then press ENTER. When this file is opened, it will show


root@navnav:~
File Edit View Terminal Go Help
/etc/sysconfig/network-scripts/ifcfg-eth0 - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
ifcfg-eth0
Ln 1, Col. 1 INS

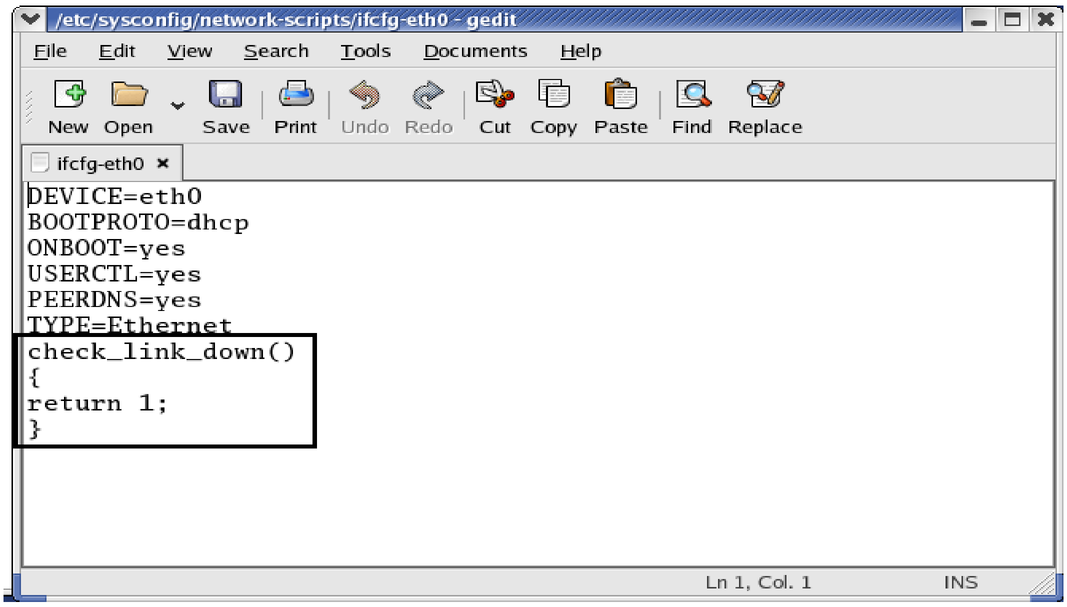
```
[root@navnav root]# gedit /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
USERCTL=yes
PEERDNS=yes
TYPE=Ethernet
```

Here at the bottom of it, we need to add some content, i.e.,

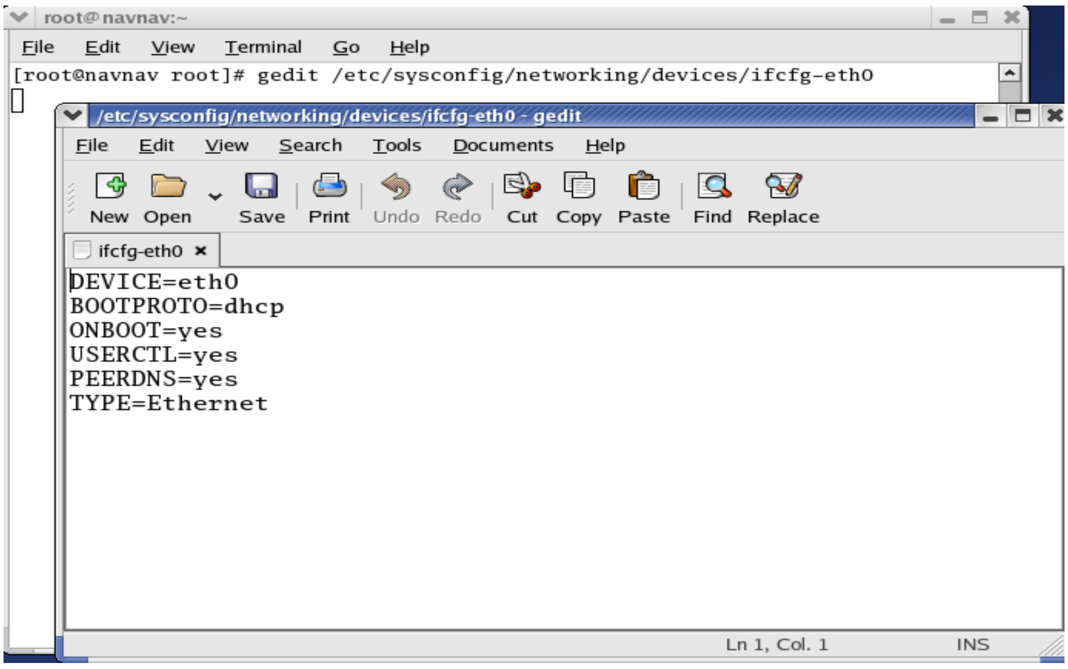
```
Check_link_down()
{
return 1;
}
```

So, after modification, it will look like


/etc/sysconfig/network-scripts/ifcfg-eth0 - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
ifcfg-eth0
Ln 1, Col. 1 INS

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
USERCTL=yes
PEERDNS=yes
TYPE=Ethernet
check_link_down()
{
return 1;
}
```

Next we need to open and modify second file: #gedit /etc/sysconfig/networking/devices/ifcfg-eth0.


root@navnav:~
File Edit View Terminal Go Help
/etc/sysconfig/networking/devices/ifcfg-eth0 - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
ifcfg-eth0
Ln 1, Col. 1 INS

```
[root@navnav root]# gedit /etc/sysconfig/networking/devices/ifcfg-eth0
```

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
USERCTL=yes
PEERDNS=yes
TYPE=Ethernet
```

Here at the bottom of it, we need to add some content, i.e.,

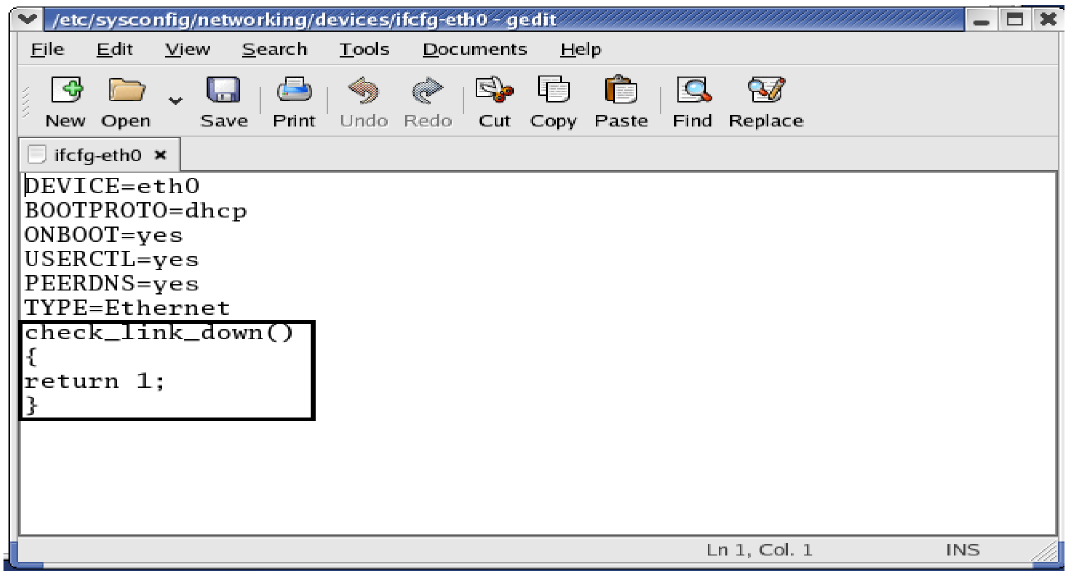
Check_link_down()

{

return 1;

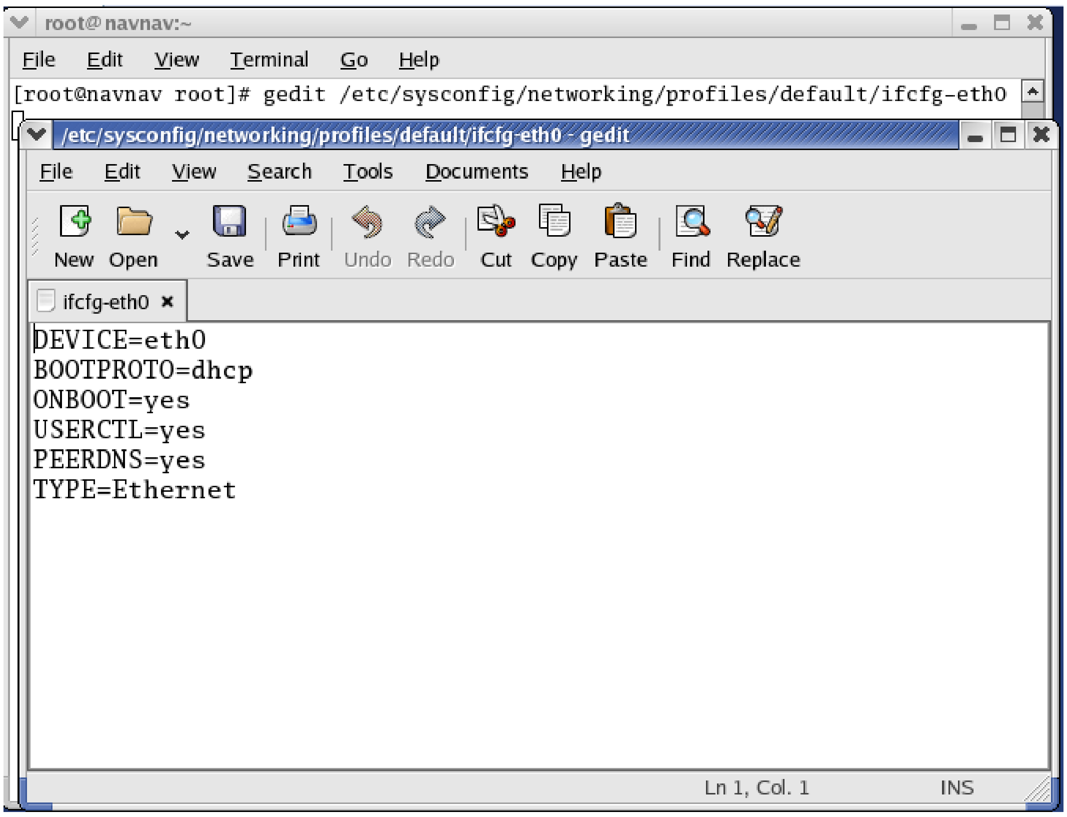
}

So, after modification, it will look like


/etc/sysconfig/networking/devices/ifcfg-eth0 - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
ifcfg-eth0
Ln 1, Col. 1 INS

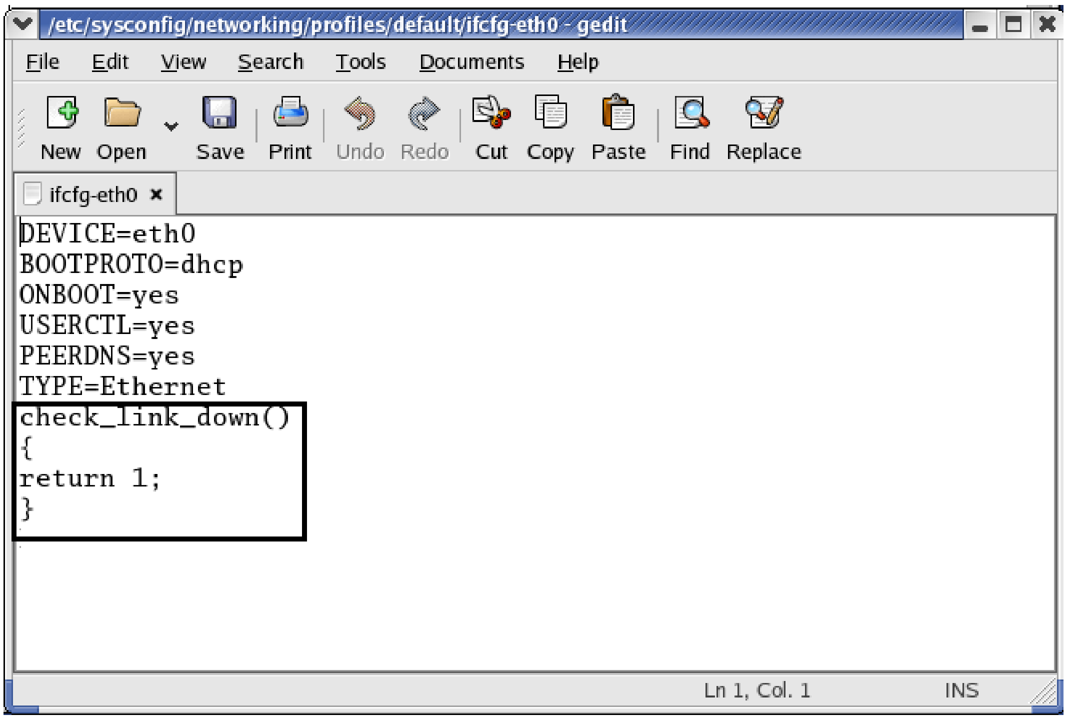
```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
USERCTL=yes
PEERDNS=yes
TYPE=Ethernet
check_link_down()
{
return 1;
}
```

Next, we will open the third file: #gedit/etc/sysconfig/networking/profiles/default/ifcfg-eth0


root@ navnav:~
File Edit View Terminal Go Help
/etc/sysconfig/networking/profiles/default/ifcfg-eth0 - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
ifcfg-eth0
Ln 1, Col. 1 INS

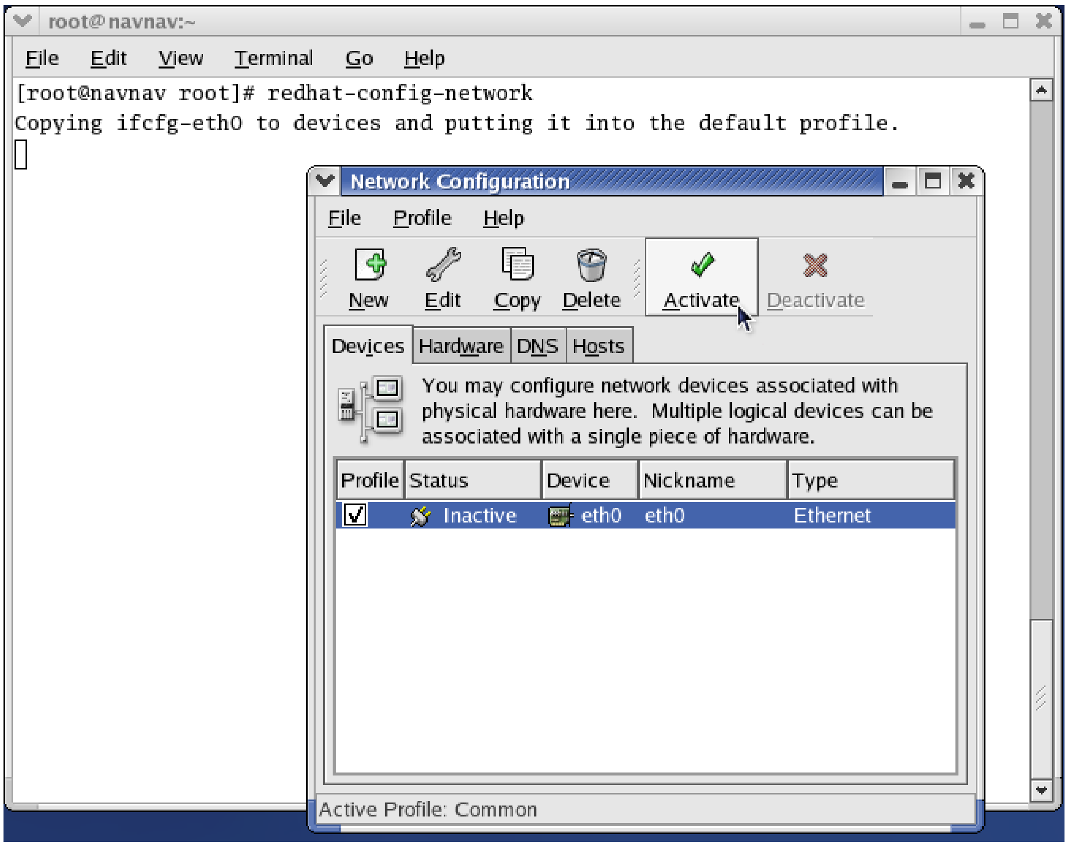
```
[root@navnav root]# gedit /etc/sysconfig/networking/profiles/default/ifcfg-eth0
```

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
USERCTL=yes
PEERDNS=yes
TYPE=Ethernet
```


/etc/sysconfig/networking/profiles/default/ifcfg-eth0 - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
ifcfg-eth0
Ln 1, Col. 1 INS

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
USERCTL=yes
PEERDNS=yes
TYPE=Ethernet
check_link_down()
{
return 1;
}
```

After modification of these three files, we ned to activate the internet.



```
[root@navnav root]# redhat-config-network
Copying ifcfg-eth0 to devices and putting it into the default profile.
```

You may configure network devices associated with physical hardware here. Multiple logical devices can be associated with a single piece of hardware.

Profile	Status	Device	Nickname	Type
✓	Inactive	eth0	eth0	Ethernet

Active Profile: Common

root@navnav:~
File Edit View Terminal Go Help
Network Configuration
File Profile Help
New Edit Copy Delete Activate Deactivate
Devices Hardware DNS Hosts

Once it is activated, we can browse the internet through web browser.

3.3 Domain Name System

- Rather than remember the IP address of the Wrox web site, it is easier for us to remember www.wrox.com.
- Domain Name System (DNS) servers provide the mapping between human−readable addresses (such as www.wrox.com) and the IP addresses of the machines acting as the web servers for the corresponding web service.
- Applications such as web browsers and e−mail clients require the IP address to connect to a web site or a mail server respectively.
- In order to get this from the human−readable input that we provide them with, they query a DNS server for the corresponding IP address information.
- Obviously, this also means that the browser and other clients on the machine need to know the IP address of the DNS server.
- For machines that use DHCP, the information about the DNS server is automatically available when the machine is configured.

Some useful Commands for connecting to internet:

- #redhat-config-network //to open network configuration manager
- ifconfig // to get information about active network
- ping //used to test the reachability of a host

Keywords

- Network Interfacing Tool is also known as the Network Administration Tool or Network Configuration Tool.
- A network device connected to a TCP/IP network has an IP address associated with it, such as 192.168.100.20. Using the IP address of a machine, other machines on the network can address it uniquely.
- Dynamic IP addresses are distributed and managed via dynamic address allocation protocols.
- Applications such as web browsers and e−mail clients require the IP address to connect to a web site or a mail server respectively.
- For machines that use DHCP, the information about the DNS server is automatically available when the machine is configured.

Summary

- **Static address**: These are allotted to machines indefinitely and do not change. Typically, static addresses are allocated to servers.
- **Dynamic address**: These are allotted to machines for a specific period with no guarantee that the same address will be available next time the machine connects to the network.
- **Devices Tab**: This tab lists the device connections that we have available on our machine.
- **Hardware Tab**: It allows us to manage the various network devices on the system, such as Ethernet cards, internal modems, and wireless cards.
- **DNS Tab**: It allows us to specify DNS server information.
- **Hosts Tab**: It allows us to modify the hostname of the machine and add aliases to the same host.
- **Domain Name System (DNS)**: It provide the mapping between human−readable addresses (such as www.wrox.com) and the IP addresses of the machines acting as the web servers for the corresponding web service.

Self Assessment

1. If we want to make a connection to the LAN, then what kind of device will be chosen?
A. CIPC Connection
B. Ethernet Connection
C. ISDN Connection
D. None of the above

2. What is an internet?
A. A tool to write the text data
B. A network that connects the computer all over the world
C. A tool to convert the word doc to pdf
D. None of the above

3. The IP address can be assigned
A. Statically
B. Dynamically
C. Both statically and dynamically

D. None of the above

4. What is another name of Network Interfacing Tool?
A. Network Administration Tool
B. Network Configuration Tool
C. Both network administration and configuration tool
D. None of the above

5. For a machine connected to a LAN or ISP using a static IP address, we need to obtain the network details like
A. IP address
B. Subnet mask
C. Default gateway address
D. All IP address, subnet mask and default gateway address

6. Which of these tabs are available in Network Interfacing Tool?
A. Device tab
B. Hardware tab
C. DNS tab
D. All device, hardware, and DNS tabs

7. In the case of a machine connected to a LAN using a dynamic IP address, the address is allocated either using ____
A. DHCP
B. BOOTP
C. Either DHCP or BOOTP
D. None of the above

8. In the process of activating the internet, how many files were modified?
A. One
B. Two
C. Three
D. Four

9. Which of these is not a web browser?
A. Windows
B. Google Chrome
C. Mozilla Firefox
D. Microsoft Edge

10. How to open the network configuration manager through command line?
A. redhat-config-network
B. redhat-config-internet

C. redhat-config-mozillabrowser
D. None of the above

11. What is used by browsers to retrieve any published resource on the web?
A. URL
B. VRL
C. LRU
D. None of the above

12. What is the path to enter the Network Administration Wizard?
A. Main Menu | System Tools | Internet Configuration Wizard
B. Main Menu | Systems | Settings
C. Main Menu | System Tools | Web Browsers
D. None of the above

13. What is the format of IP address?
A. x.x
B. x.x.x
C. x.x.x.x
D. None of the above

14. What is required to access internet?
A. Pdf converter
B. Photoshop
C. Web browser
D. None of the above

15. What provides the interfacing between human-readable address and IP address of the machine?
A. DNS
B. ABC
C. Wrox
D. None of the above

Answers for Self Assessment

1.	C	2.	B	3.	C	4.	C	5.	D
6.	D	7.	C	8.	C	9.	A	10.	A
11.	A	12.	A	13.	C	14.	C	15.	A

Review Questions:

1. Is it necessary to configure the system before the internet connection take place? If yes, how can we configure it?
2. What is an IP address? Explain the difference between static and dynamic IP address.
3. What is a network interfacing tool? Explain its tabs.
4. What is DNS? Explain.
5. Explain the process of configuring the system for internet connection.

Further Readings

Mark G. Sobell, A Practical Guide to Fedora and Red Hat Enterprise Linux, Fifth Edition, Pearson Education, Inc.

Web Links

https://www.redhat.com/sysadmin/network-interface-linux

Unit 04: Installing Software

CONTENTS

Objectives

After studying this unit, you will be able to understand

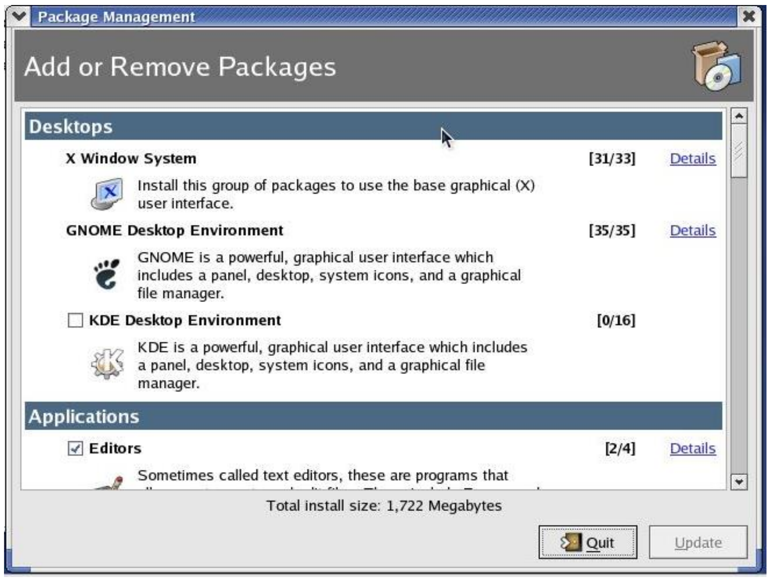
- Understand RPM
- See the RPM Package Management tool
- Add and remove the packages
- Query the RPM package
- Install package in TAR format

Introduction

RPM stands for Redhat Package Manager. The RPM package manager is an open-source packaging system distributed under the GNU GPL. It runs on most Linux distributions and makes it easy for you to install, uninstall, and upgrade the software on your machine. RPM files can be easily recognized by their .rpm file extension and the 'package' icon that appears in your navigation window.The RPM package management is shown below.

Notes

4.1 RPM Package Manager


Package Management

Add or Remove Packages

Desktops

X Window System [31/33] Details

Install this group of packages to use the base graphical (X) user interface.

GNOME Desktop Environment [35/35] Details

GNOME is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

☐ KDE Desktop Environment [0/16]

KDE is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

Applications

☑ Editors [2/4] Details

Sometimes called text editors, these are programs that

Total install size: 1,722 Megabytes

Benefits of using RPM: There are few reasons to use RPM.

- **Simplicity:** RPM is quite simple to use. The interface of RPM is very clear. The packages and the groups in RPM are very easy to locate. So, this is the remarkable feature of RPM.
- **Upgradability**: RPM interface is easy to upgrade. If a new package comes, it can be easily upgraded.
- **Manageability**: RPM interface is easily manageable. If we want to add or delete some packages using RPM, then it can be easily done. So, the manageability is one of the greatest feature of RPM interface.
- **Package Queries**: The packages are easily queried in RPM. By querying the packages, we can see which all packages are installed in the computer system.
- **Uninstalling**: It is very easy to uninstall a package or a group. If we don't need any package or its related extra group at some time, then it can be deleted at that time.
- **System Verification**: System verification can be easily done using RPM.
- **Security**: The RPM way of installing and installing packages is secure.

Ways to use RPM: RPM can be used in two different, yet complementary ways –

- From the desktop, using the GUI interface,
- From the command line.

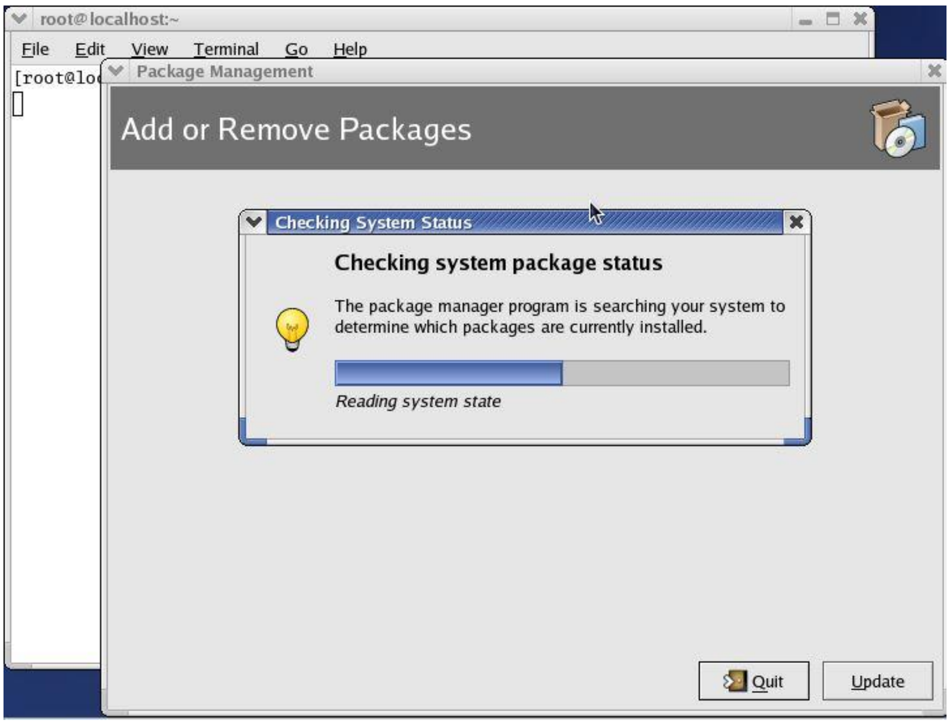
The RPM package management (GUI) tool

This tool is a graphical user interface (GUI) designed for the management of package installation and removal. The GUI allows us to add and remove packages at the click of a mouse.

Starting the RPM Package Management Tool: There are two ways to start RPM.

- Main Menu, select Main Menu | System Settings | Add/Remove Applications.
- $ redhat–config–packages

When you use either of the way, the next moment your screen will look like this. It checks the system package status.



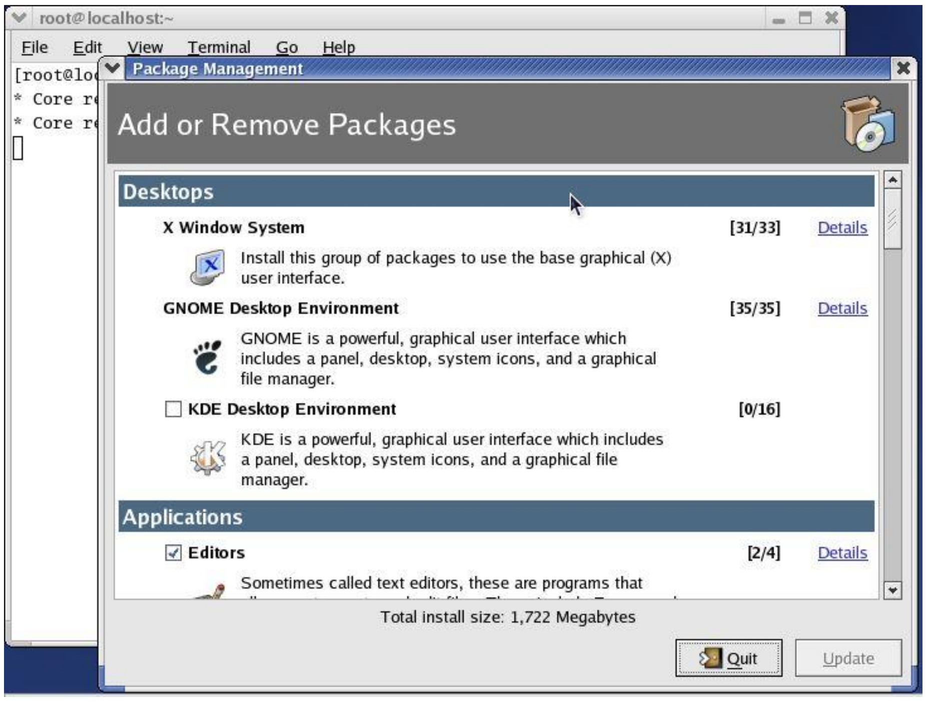
Checking system package status

The package manager program is searching your system to determine which packages are currently installed.

Reading system state

The package management window shows all the packages and its group. The interface contains:

- Package category
- Package group
- Details link
- Number of packages installed/Out of total number of packages
- Summary of disk space required to install the package
- Update button
- Quit button



X Window System [31/33]

Install this group of packages to use the base graphical (X) user interface.

GNOME Desktop Environment [35/35]

GNOME is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

☐ KDE Desktop Environment [0/16]

KDE is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

☑ Editors [2/4]

Sometimes called text editors, these are programs that

Total install size: 1,722 Megabytes

Package categories and groups: There are various package categories and groups available.

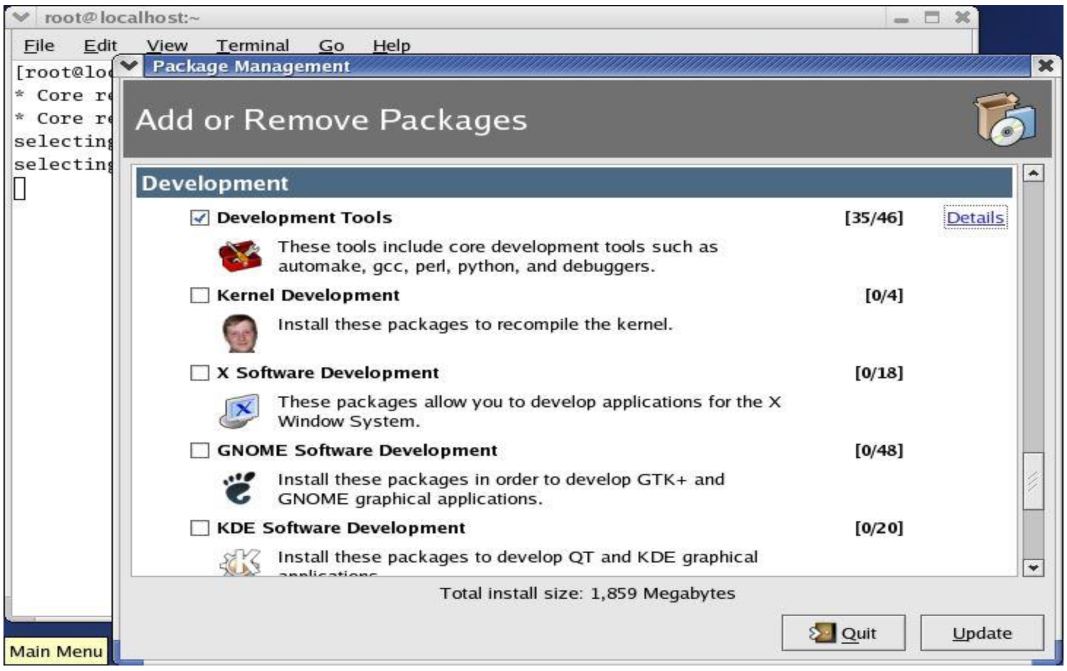
Standard and extra packages

Each group may have standard packages and extra packages, or just extra packages. Standard packages are always available when a package group is installed − so you can't add or remove them explicitly unless the entire group is removed. However, the extra packages are optional so they can be individually selected for installation or removal at any time.

Package Category	Package Groups
Desktop	X Window System GNOME Desktop Environment KDE Desktop Environment
Applications	Editors Engineering and Scientific Graphical Internet Text−based Internet Office/Productivity Sound and Video Authoring and Publishing Graphics Games and Entertainment
Servers	Server Configuration Tools Web Server Mail Server Windows File Server DNS Name Server FTP Server SQL Database Server News Server Network Servers
Development	Development Tools Kernel Development X Software Development GNOME Software Development KDE Software Development
System	Administration Tools System Tools Printing Support

4.2 Adding and Removing Packages

Installing new software from the package management tool is very simple. When we select any group using the RPM package management tool interface, it automatically selects the standard packages (if any) that are needed for the category as well as any dependent packages that it may have.We can customize the packages to be installed by clicking on the Details button. Once you've made your selections, click on the Update button on the main window. The package management tool will then calculate the disk space required for installing packages, as well as any dependencies, before displaying the following dialog:



```
[root@lo
* Core r
* Core r
selecting
selecting
```

Add or Remove Packages

Development

[✓] Development Tools [35/46]
These tools include core development tools such as automake, gcc, perl, python, and debuggers.

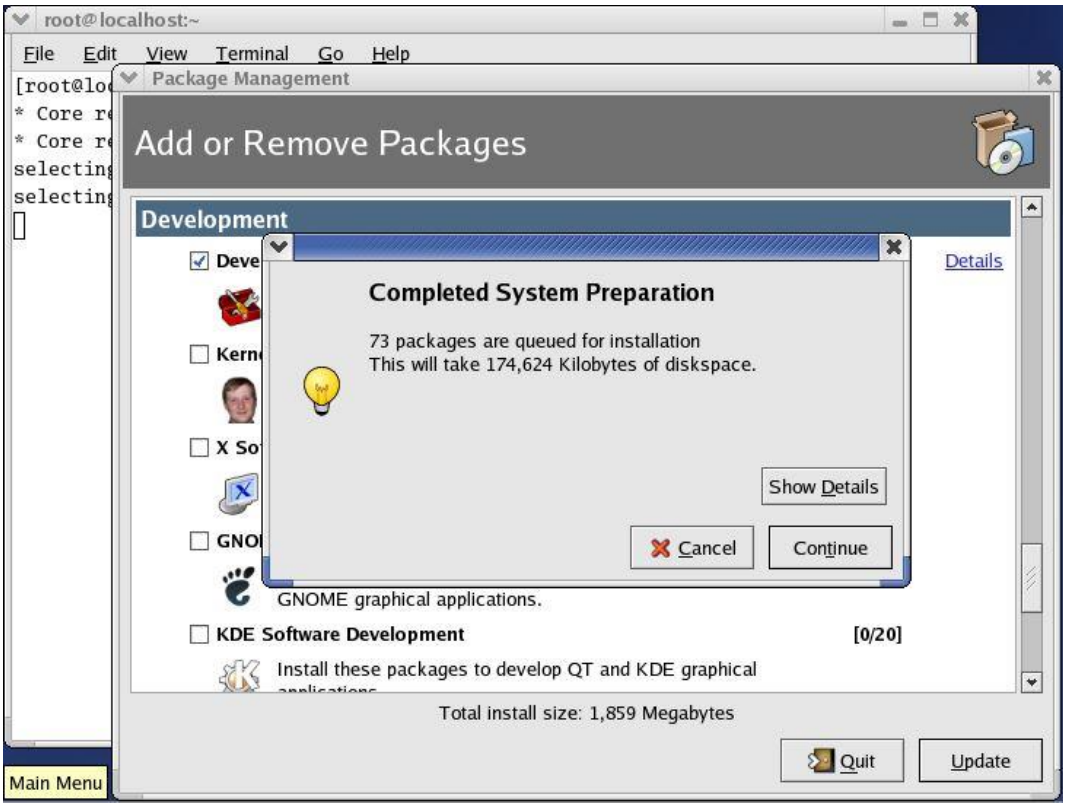
[ ] Kernel Development [0/4]
Install these packages to recompile the kernel.

[ ] X Software Development [0/18]
These packages allow you to develop applications for the X Window System.

[ ] GNOME Software Development [0/48]
Install these packages in order to develop GTK+ and GNOME graphical applications.

[ ] KDE Software Development [0/20]
Install these packages to develop QT and KDE graphical

Total install size: 1,859 Megabytes



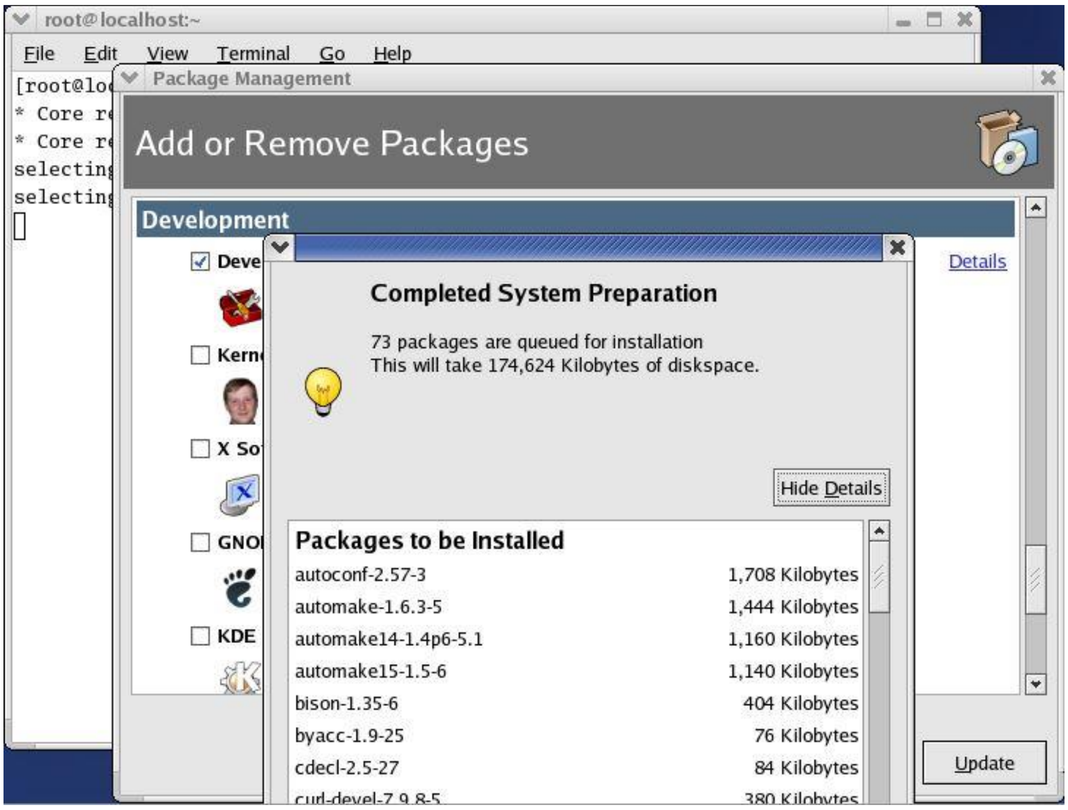
Completed System Preparation

73 packages are queued for installation
This will take 174,624 Kilobytes of diskspace.

GNOME graphical applications.

[ ] KDE Software Development [0/20]
Install these packages to develop QT and KDE graphical

Total install size: 1,859 Megabytes



```
[root@lo
* Core r
* Core r
selectin
selectin
```

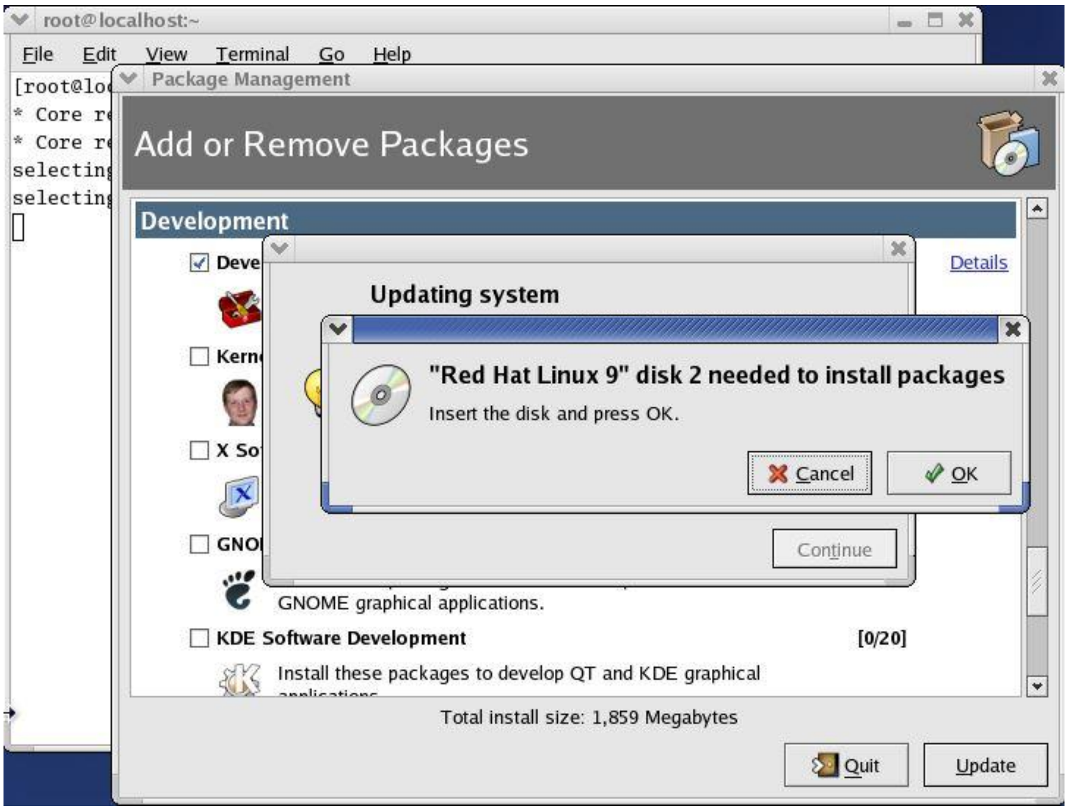
Completed System Preparation

73 packages are queued for installation
This will take 174,624 Kilobytes of diskspace.

Packages to be Installed

```
autoconf-2.57-3                1,708 Kilobytes
automake-1.6.3-5               1,444 Kilobytes
automake14-1.4p6-5.1           1,160 Kilobytes
automake15-1.5-6               1,140 Kilobytes
bison-1.35-6                     404 Kilobytes
byacc-1.9-25                      76 Kilobytes
cdecl-2.5-27                      84 Kilobytes
curl-devel-7.9.8-5               380 Kilobytes
```

Use the required disk for updation of system.



```
[root@lo
* Core r
* Core r
selectin
selectin
```

Updating system

"Red Hat Linux 9" disk 2 needed to install packages

Insert the disk and press OK.

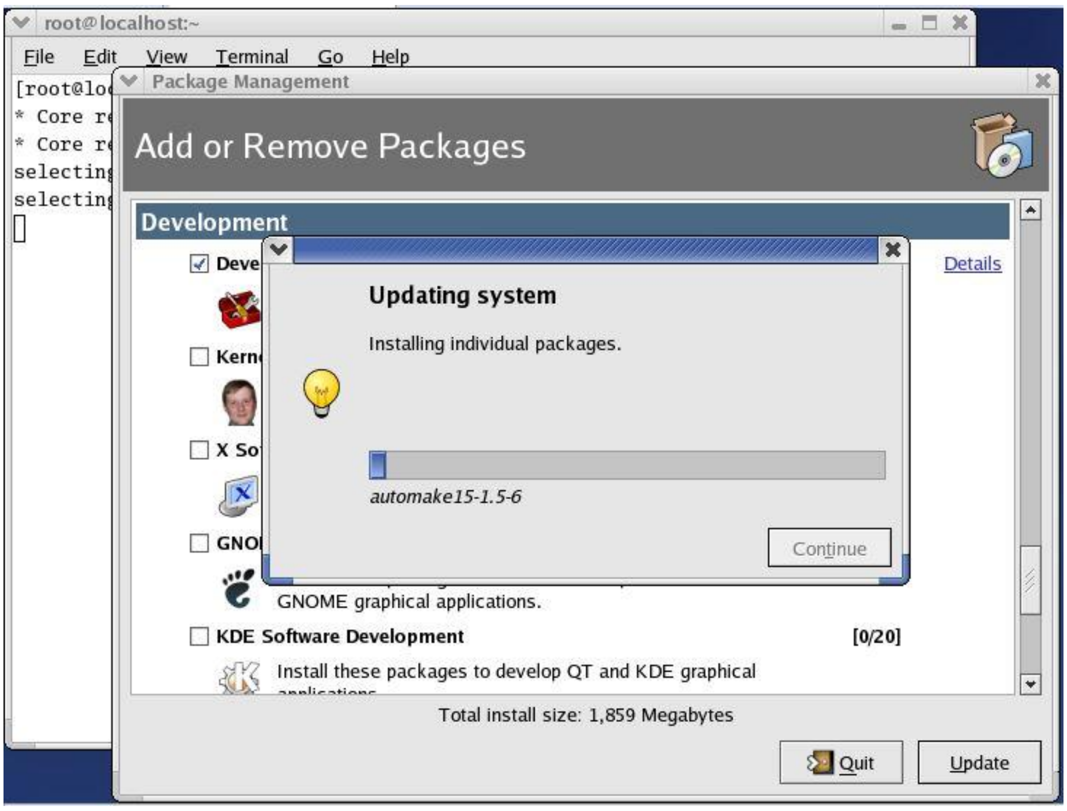
GNOME graphical applications.

KDE Software Development [0/20]

Install these packages to develop QT and KDE graphical

Total install size: 1,859 Megabytes

It will start updating the system.



```
[root@lo
* Core r
* Core r
selectin
selectin
```

Updating system

Installing individual packages.

automake15-1.5-6

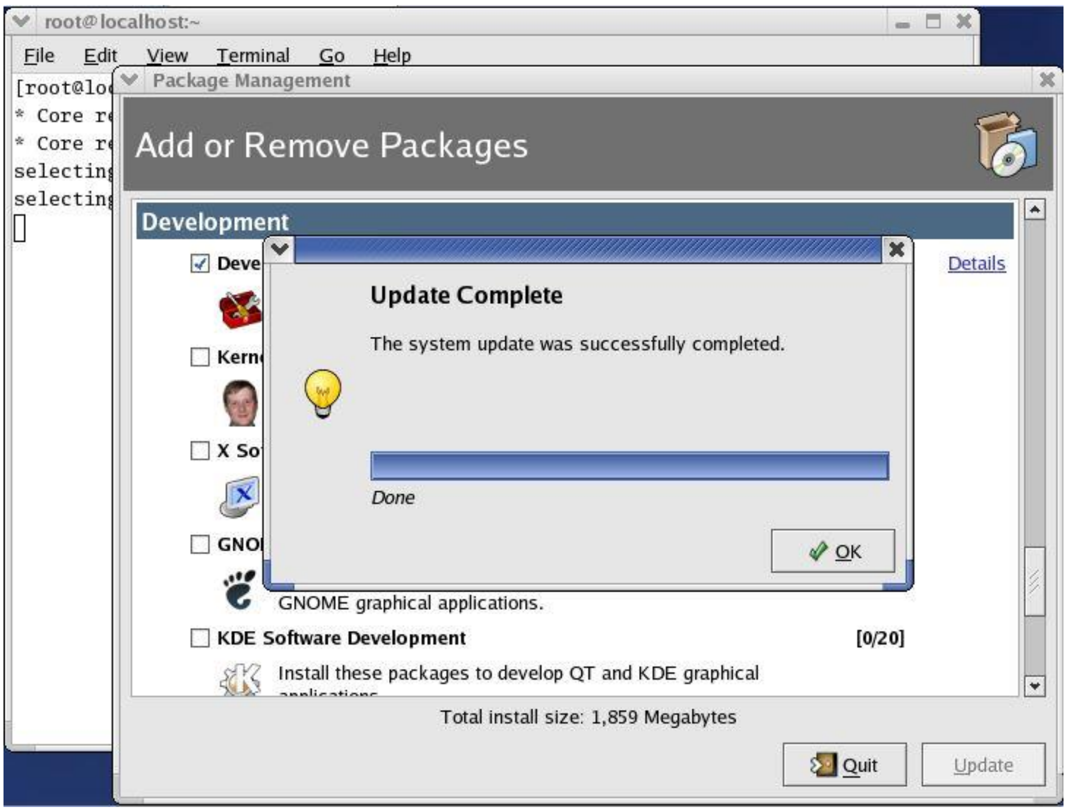
GNOME graphical applications.

KDE Software Development [0/20]

Install these packages to develop QT and KDE graphical

Total install size: 1,859 Megabytes

When the update is complete, it will show the updation complete message.



```
[root@lo
* Core r
* Core r
selectin
selectin
```

Update Complete

The system update was successfully completed.

Done

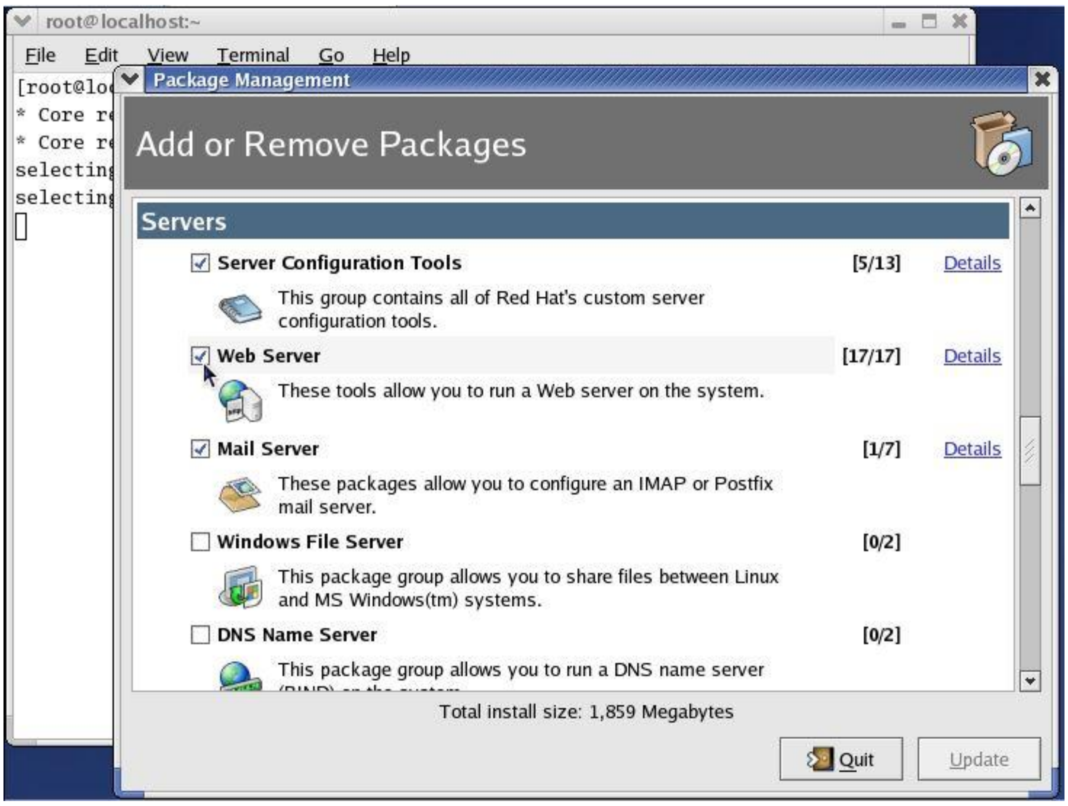
GNOME graphical applications.

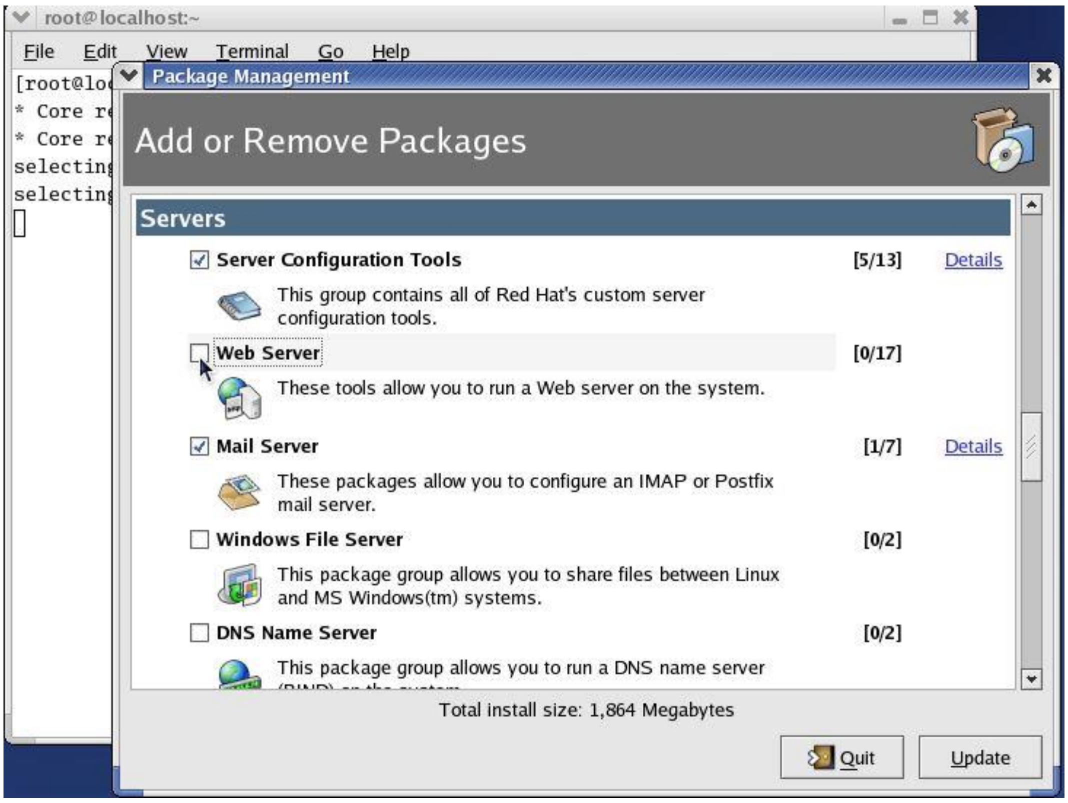
KDE Software Development [0/20]

Install these packages to develop QT and KDE graphical

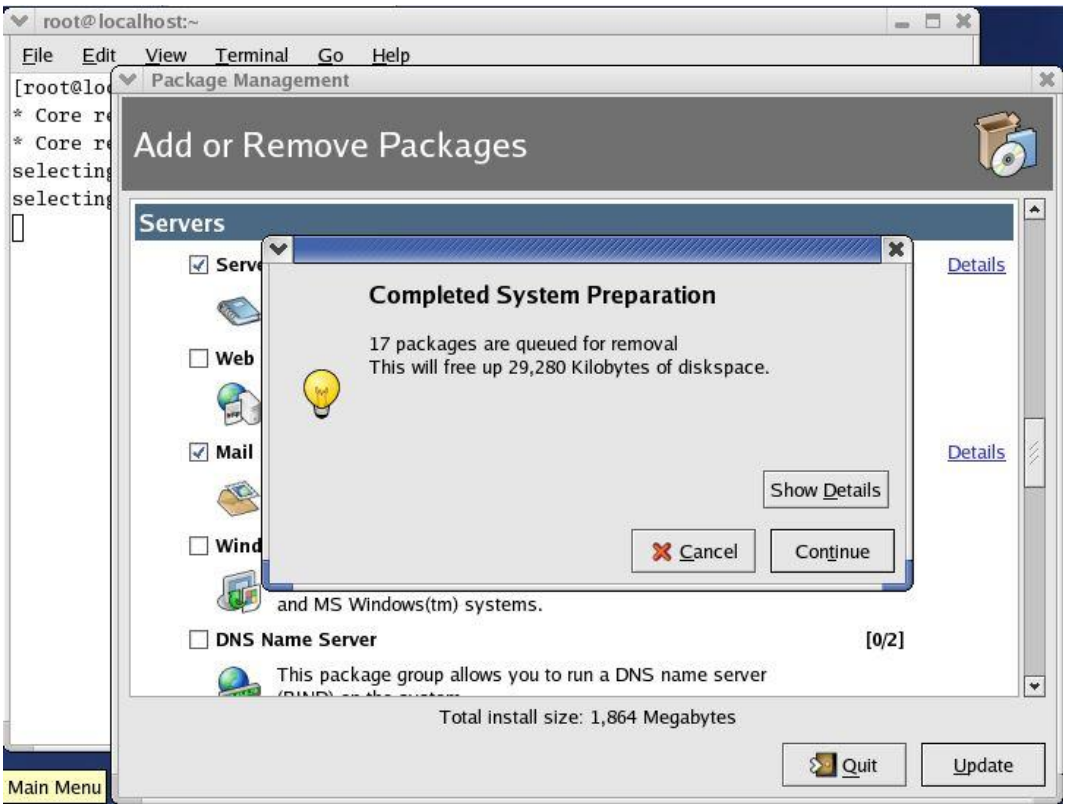
Total install size: 1,859 Megabytes

We can also delete the package if some package is no longer required. We just need to uncheck that package and click on update.


```
[root@loc
* Core re
* Core re
selecting
selecting
```
root@localhost:~
File Edit View Terminal Go Help
Package Management
Add or Remove Packages
Servers
Server Configuration Tools [5/13] Details
This group contains all of Red Hat's custom server configuration tools.
Web Server [17/17] Details
These tools allow you to run a Web server on the system.
Mail Server [1/7] Details
These packages allow you to configure an IMAP or Postfix mail server.
Windows File Server [0/2]
This package group allows you to share files between Linux and MS Windows(tm) systems.
DNS Name Server [0/2]
This package group allows you to run a DNS name server
Total install size: 1,859 Megabytes
Quit
Update


```
[root@loc
* Core re
* Core re
selecting
selecting
```
root@localhost:~
File Edit View Terminal Go Help
Package Management
Add or Remove Packages
Servers
Server Configuration Tools [5/13] Details
This group contains all of Red Hat's custom server configuration tools.
Web Server [0/17]
These tools allow you to run a Web server on the system.
Mail Server [1/7] Details
These packages allow you to configure an IMAP or Postfix mail server.
Windows File Server [0/2]
This package group allows you to share files between Linux and MS Windows(tm) systems.
DNS Name Server [0/2]
This package group allows you to run a DNS name server
Total install size: 1,864 Megabytes
Quit
Update

Then the packages will be placed in queue for deletion. Here the 17 packages are queued for removal. It will also show how much space will be freed after removal of packages. If the packages are correct for deletion (click on show details for confirmation), click on continue. It will remove the packages from the system.



```
[root@lo
* Core r
* Core r
selecting
selecting
```

Serv

Web

Mail

Wind

Completed System Preparation

17 packages are queued for removal
This will free up 29,280 Kilobytes of diskspace.

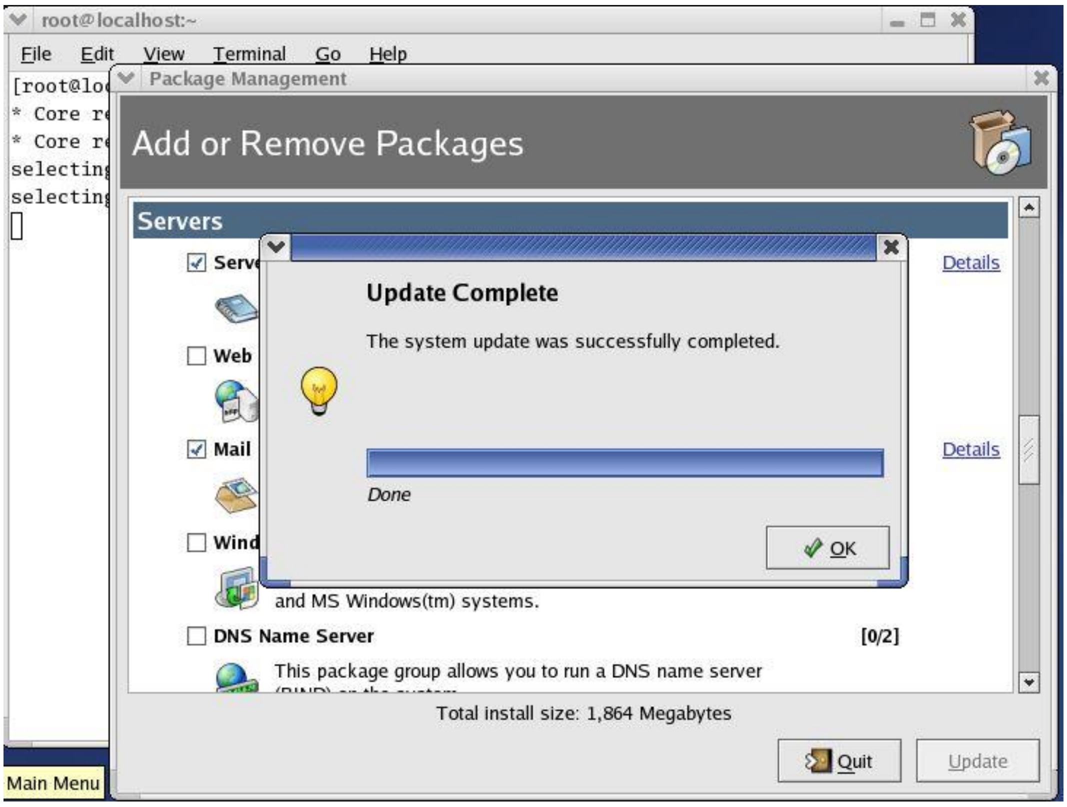
and MS Windows(tm) systems.

DNS Name Server [0/2]

This package group allows you to run a DNS name server

Total install size: 1,864 Megabytes

When the update is complete. Click on ok.



```
[root@lo
* Core r
* Core r
selecting
selecting
```

Serv

Web

Mail

Wind

Update Complete

The system update was successfully completed.

Done

and MS Windows(tm) systems.

DNS Name Server [0/2]

This package group allows you to run a DNS name server

Total install size: 1,864 Megabytes

4.3 RPM Command Line Tool

Up to now, we've talked about how to install and remove packages using Red Hat's graphical package management tool. While this tool is simple to use, it's lacking in functionality. For example:

- It cannot install packages using network, FTP, or HTTP connections.
- It does not show the location the files in a package are installed to.
- It lacks the capability to query for specific packages installed on the system.
- It does not provide full details of the RPM package − such as the vendor, build date, signature, description, and so on.
- It does not have a function to verify a package. That means it cannot compare information about files like size, MD5 sum, permissions, type, owner, and group installed from a package with the same information from the original package.
- It does not show all the packages available in a product. So you won't always know if you've got the whole thing.

4.4 Querying Packages

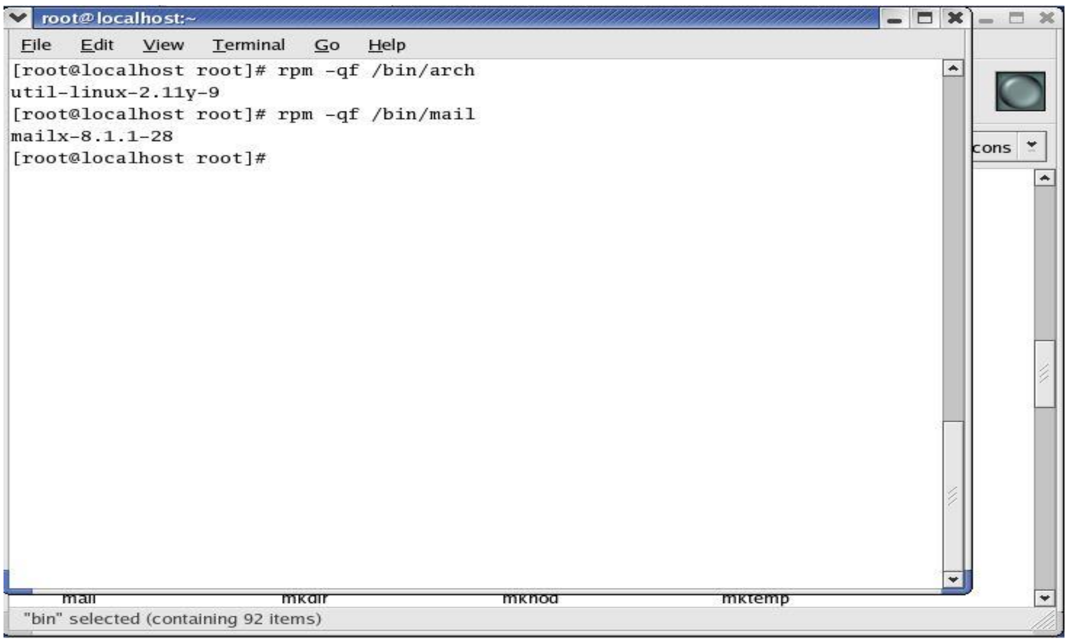
RPM keeps a record of all the packages installed on your system in a database. By querying the database, you obtain a complete list of all the packages that you've installed on your system. Should you want to, you can then go further and query each individual package for more details about itself.

The syntax for a basic query is as follows:

- rpm −q [options] <filename>

You have list of files, and you would like to find out which package belongs to these files.The syntax is:

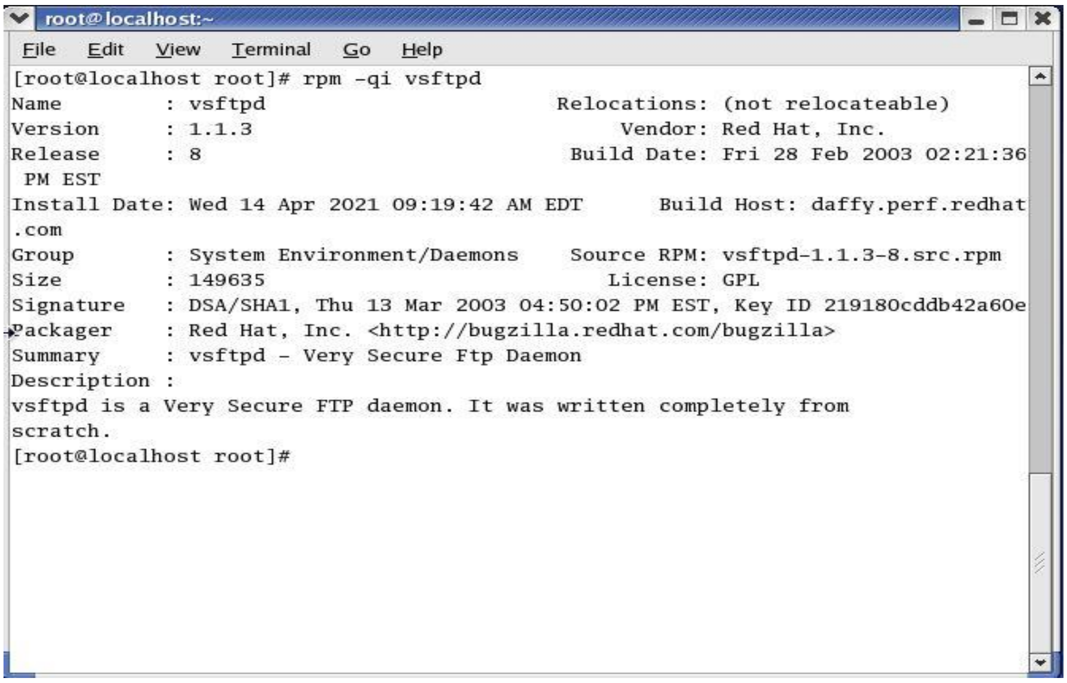
- rpm –qf<filename>


root@localhost:~
File Edit View Terminal Go Help
cons
mail mkdir mknod mktemp
"bin" selected (containing 92 items)

```
[root@localhost root]# rpm -qf /bin/arch
util-linux-2.11y-9
[root@localhost root]# rpm -qf /bin/mail
mailx-8.1.1-28
[root@localhost root]#
```

You have installed an rpm package and want to know the information about the package. The syntax is:

- rpm –qi <package _name>



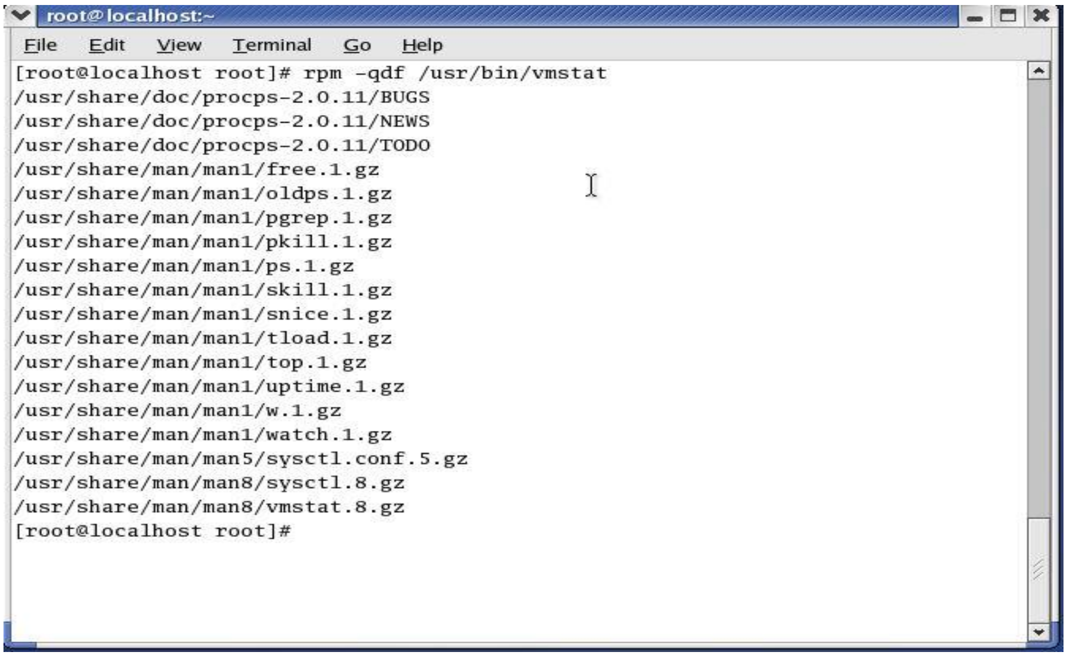
```
[root@localhost root]# rpm -qi vsftpd
Name        : vsftpd                    Relocations: (not relocateable)
Version     : 1.1.3                          Vendor: Red Hat, Inc.
Release     : 8                          Build Date: Fri 28 Feb 2003 02:21:36
 PM EST
Install Date: Wed 14 Apr 2021 09:19:42 AM EDT      Build Host: daffy.perf.redhat
.com
Group       : System Environment/Daemons    Source RPM: vsftpd-1.1.3-8.src.rpm
Size        : 149635                           License: GPL
Signature   : DSA/SHA1, Thu 13 Mar 2003 04:50:02 PM EST, Key ID 219180cddb42a60e
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : vsftpd - Very Secure Ftp Daemon
Description :
vsftpd is a Very Secure FTP daemon. It was written completely from
scratch.
[root@localhost root]#
```

You have downloaded a package from the internet and want to know the information of a package before installing.The syntax is:

- rpm –qip<package_name>

To get the list of available documentation of an installed package.The syntax is:

- rpm –qdf<package_name>



```
[root@localhost root]# rpm -qdf /usr/bin/vmstat
/usr/share/doc/procps-2.0.11/BUGS
/usr/share/doc/procps-2.0.11/NEWS
/usr/share/doc/procps-2.0.11/TODO
/usr/share/man/man1/free.1.gz
/usr/share/man/man1/oldps.1.gz
/usr/share/man/man1/pgrep.1.gz
/usr/share/man/man1/pkill.1.gz
/usr/share/man/man1/ps.1.gz
/usr/share/man/man1/skill.1.gz
/usr/share/man/man1/snice.1.gz
/usr/share/man/man1/tload.1.gz
/usr/share/man/man1/top.1.gz
/usr/share/man/man1/uptime.1.gz
/usr/share/man/man1/w.1.gz
/usr/share/man/man1/watch.1.gz
/usr/share/man/man5/sysctl.conf.5.gz
/usr/share/man/man8/sysctl.8.gz
/usr/share/man/man8/vmstat.8.gz
[root@localhost root]#
```

4.5 Package Installation in TAR Format

TAR stands for tape archive.An archive is nothing, but you bundle (put) many files together into a single file on a single tape or disk.The tar program combines multiple files into a single large file. It is separate from the compression tool, so it allows you to select which compression tool to use or whether you even want compression.A tar ball is a (usually compressed) archive of files, similar to a Zip file on Windows or a Sit on the Mac. Tar balls come in files that end in .tar, .tar.gz, .tgz, or something along these lines.

- Structure of the tar command

  [root@localhost /root]# tar [commands and options] filename

- Option for tar

-c	Create a new archive.
-t	View the contents of an archive.
-x	Extract the contents of an archive.
-f	Specify the name of the file (or device) in which the archive is located.
-v	Be verbose during operations.
-z	Use gzip to compress or decompress the file.
-u	To upgrade the tar file

To create a tar file

[root@localhostmohit]# tar -cvf filename.tar directory [path of files]

In this example, filename.tar represents the file you are creating and directory/file represents the directory and file you want to put in the archived file.

Where,

-c : Create a tar ball.

-v : Verbose output (show progress).

-f : Output tar ball archive file name.

-x : Extract all files from archive.tar.

-t : Display the contents (file list) of an archive.

To View a Tar Ball

- It Contains (list file inside a tar ball)
- Type the following command:

  tar -tvf/tmp/data.tar **ar Ball**

To Extract a Tar Ball

Type the following command to extract /tmp/data.tar in a current directory, enter:

tar -xvf/tmp/data.tar

Summary

- RPM files can be easily recognized by their .rpm file extension and the 'package' icon that appears in your navigation window.
- There are various benefits of using RPM for package installation and deletion: simplicity, upgradability, manageability, easy uninstallation, system verification and high security.
- The RPM package management tool contains: the button for package category, package group, details link, number of packages installed/Out of total number of packages, summary of disk space required to install the package, update button, and quit button.
- Each group may have standard packages and extra packages, or just extra packages.
- We can customize the packages to be installed by clicking on the Details button. Once you've made your selections, click on the Update button on the main window.
- RPM keeps a record of all the packages installed on your system in a database.

- The tar program combines multiple files into a single large file. It is separate from the compression tool, so it allows you to select which compression tool to use or whether you even want compression.

Keywords

- **RPM:**The RPM package manager is an open-source packaging system distributed under the GNU GPL.
- **Standard packages**: These are always available when a package group is installed − so you can't add or remove them explicitly unless the entire group is removed.
- **Extra packages**:These are optional so they can be individually selected for installation or removal at any time.
- **Archive:**An archive is nothing, but you bundle (put) many files together into a single file on a single tape or disk.
- **Tar ball**: A tar ball is a (usually compressed) archive of files, like a Zip file on Windows or a Sit on the Mac. Tar balls come in files that end in .tar, .tar.gz, .tgz, or something along these lines.

Self Assessment

1. TAR stands for

A. Tour Archive
B. Tape Archive
C. Tape Assistance
D. Tour Assistance

2. Under development tools, what can be installed using RPM package management tool?

A. KDE Software development
B. GNOME Software development
C. X Software development
D. All of the above

3. While graphical interface of RPM package management tool can install/remove/update the packages, but it still lacks which functionality.

A. It cannot install packages using network, FTP, or HTTP connections.
B. It does not show the location the files in a package are installed to.
C. Both above
D. None of the above

4. The RPM package management tool is a ______

A. Graphical interface
B. Textual interface
C. Not an interface
D. None of the above

5. Using RPM package management tool, we can install
A. Web server
B. Mail server
C. DNS name server
D. All of the above

6. We can which packages are installed/ not installed by clicking on
A. Update button
B. Quit button
C. Details link
D. None of the above

7. Which of these packages are always available when a package group is installed?
A. Standard packages
B. Extra packages
C. Grouped packages
D. None of the above

8. Which of the buttons are available on the tool interface?
A. Update button
B. Quit button
C. Both update and quit buttons
D. None of the above

9. The slash (/) in the interface of RPM package management tool represents:
A. Total number of packages/ Number of packages installed
B. Number of packages installed/ total number of packages
C. Package category/ package group
D. Package group/ package category

10. What is the extension of the RPM file
A. .txt
B. .doc
C. .rpm
D. .pdf

11. How can we start RPM package management tool?
A. $redhat-config-services
B. $redhat-config-packages
C. $redhat-config-management
D. None of the above

12. What are the benefits of using RPM?

A. Package queries
B. System verification
C. Security
D. All of the above

13. With RPM, it is easy to ______ softwares on the computer system.
A. Install
B. Uninstall
C. Upgrade
D. All: install, uninstall and upgrade

14. From ___________________, it is possible to install packages network, FTP or HTTP connections.
A. Command line
B. Graphical interface
C. Both command line and graphical interface
D. None of the above

15. Which of these commands is used for querying the information about a package after installation?
A. rpm –qi<filename>
B. rpm –qu<filename>
C. rpm –qr<filename>
D. None of the above

Answers for Self Assessment

1.	B	2.	D	3.	C	4.	A	5.	D
6.	C	7.	A	8.	C	9.	B	10.	C
11.	B	12.	D	13.	D	14.	A	15.	A

Review Questions:

1. What is RPM? Write the ways and benefits of using RPM.
2. What is RPM package management tool? How can we start it? Explain some details about its interface.
3. How can we add and remove the packages? Explain.
4. What is RPM command line tool? Write its benefits.
5. How can we query a package? Write syntax.
6. Explain the package installation in TAR format. How can we create, view and extract a tar ball?

Further Readings

Mark G. Sobell, A Practical Guide to Fedora and Red Hat Enterprise Linux, Fifth Edition, Pearson Education, Inc.

Web Links

https://www.redhat.com/sysadmin/create-rpm-package

Unit 05: Utilities

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand the basic utilities
- Work with files
- Understand the Pipe
- Understand the compressing and archiving of files
- Understand the locating commands

Introduction

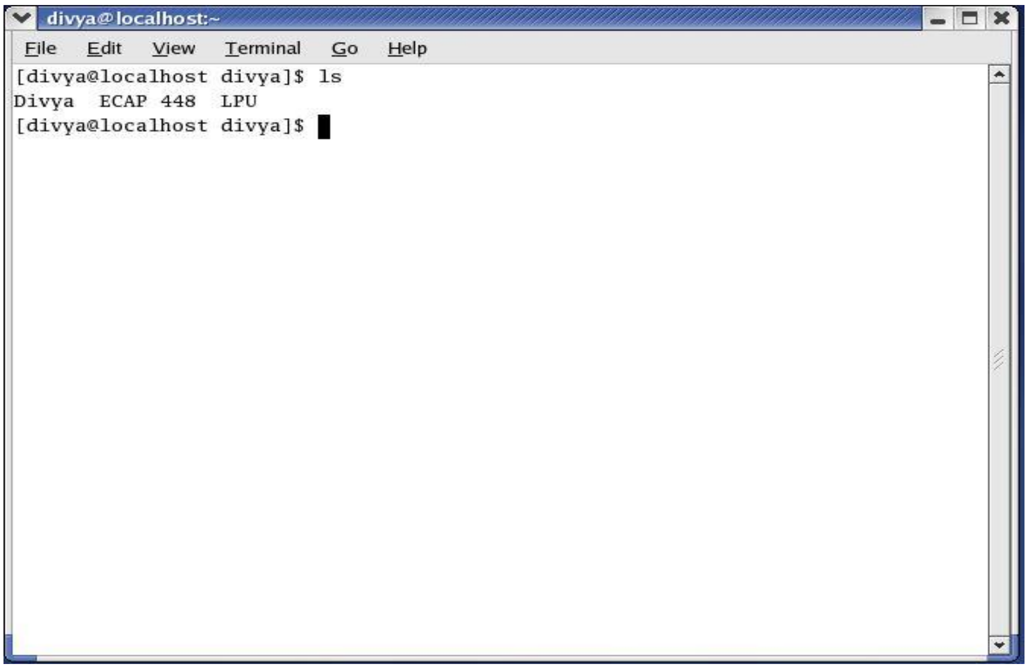
Command-line utilities are often faster, more powerful, or more complete than their GUI counterparts. When you work with a command-line interface, you are working with a shell. One of the important advantages of Linux is that it comes with thousands of utilities that perform innumerable functions.

- ls
- cat
- rm
- less
- more

5.1 Common Utilities

ls: Lists the Names of Files

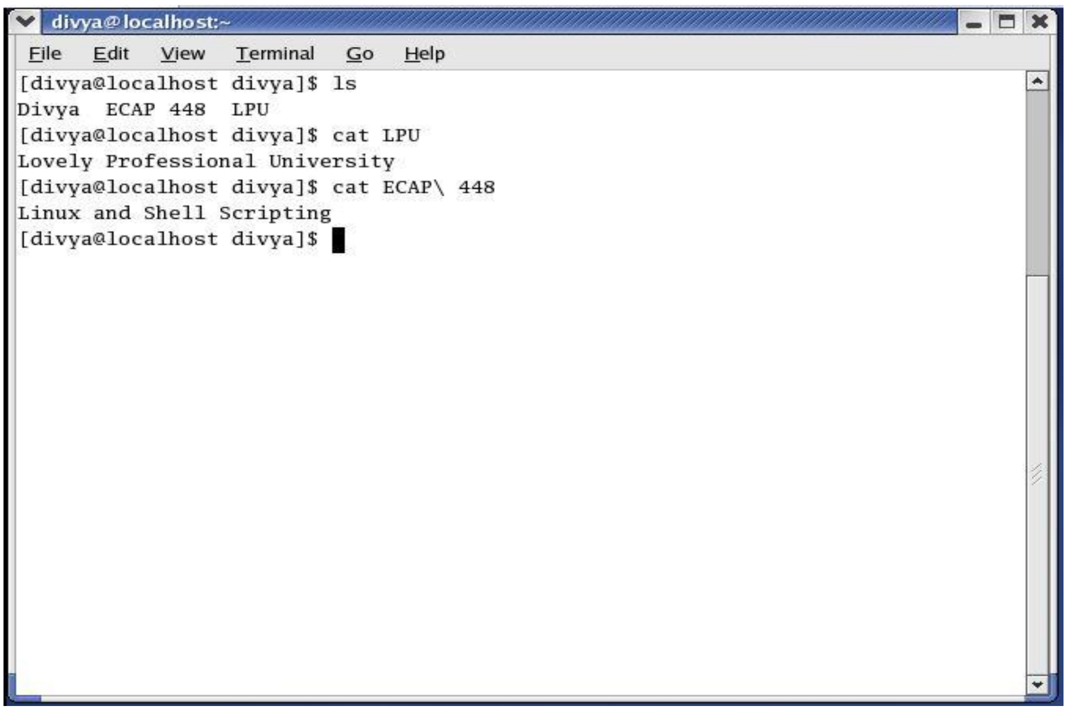
The ls utility lists the names of files which are available. **ls** is a Linux shell command that lists directory contents of files and directories.



```
[divya@localhost divya]$ ls
Divya  ECAP 448  LPU
[divya@localhost divya]$ 
```

cat: Displays the Text of File

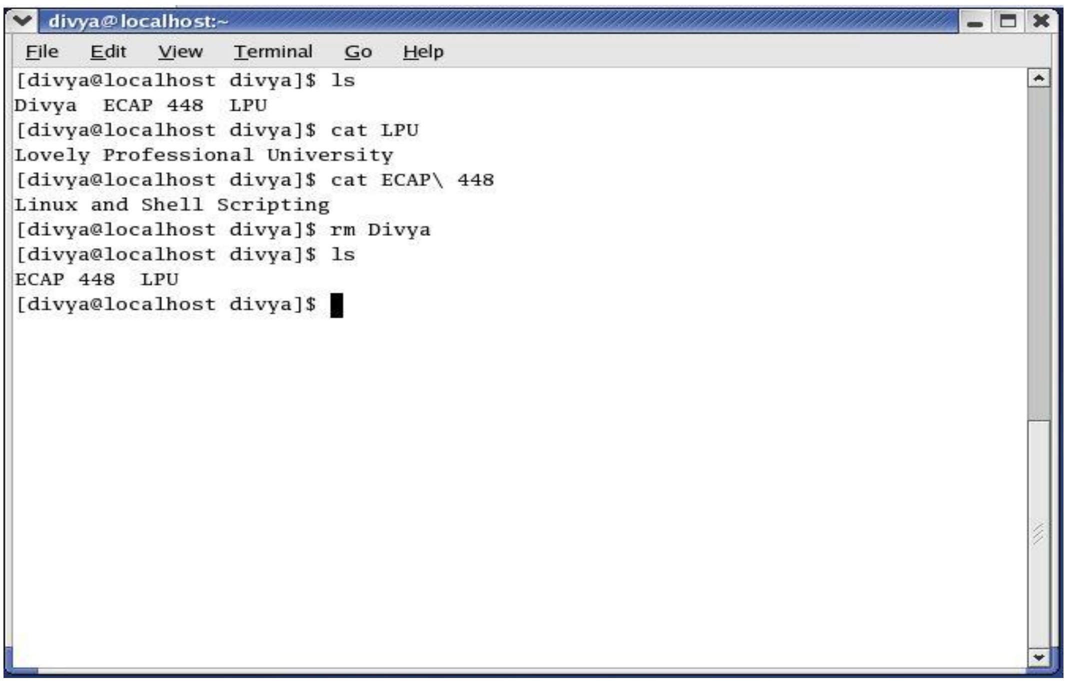
The cat utility displays the contents of a text file. The name of the command is derived from catenate, which means to join, one after the other.



```
[divya@localhost divya]$ ls
Divya  ECAP 448  LPU
[divya@localhost divya]$ cat LPU
Lovely Professional University
[divya@localhost divya]$ cat ECAP\ 448
Linux and Shell Scripting
[divya@localhost divya]$ 
```

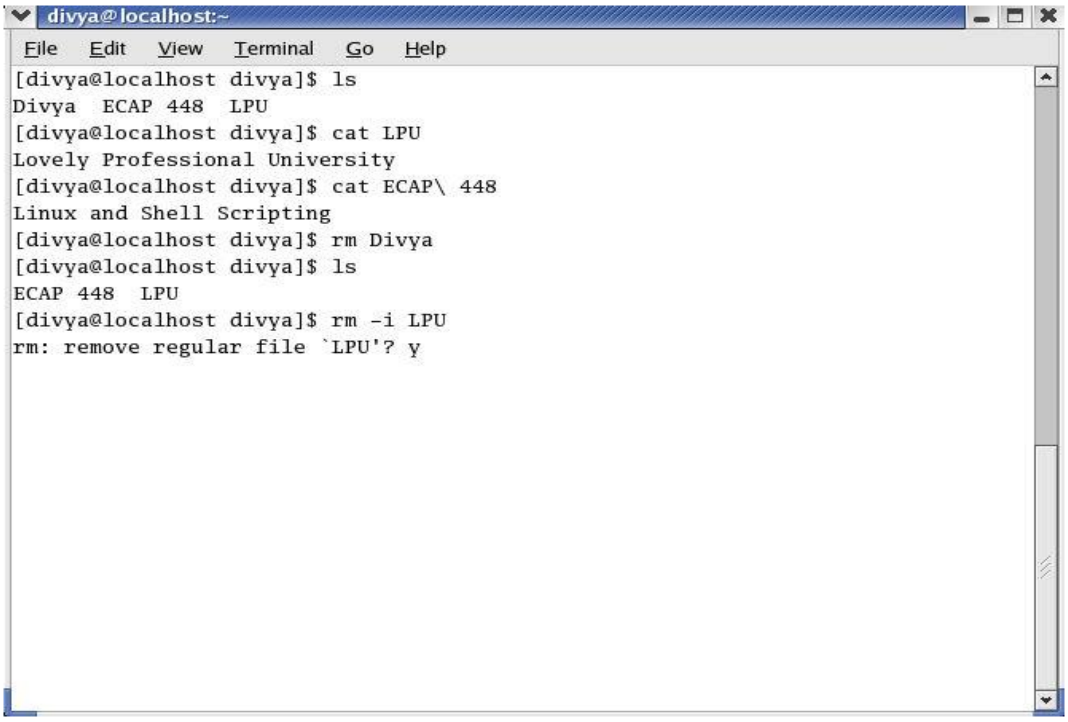
rm: Deletes a File

The rm (remove) utility deletes a file.

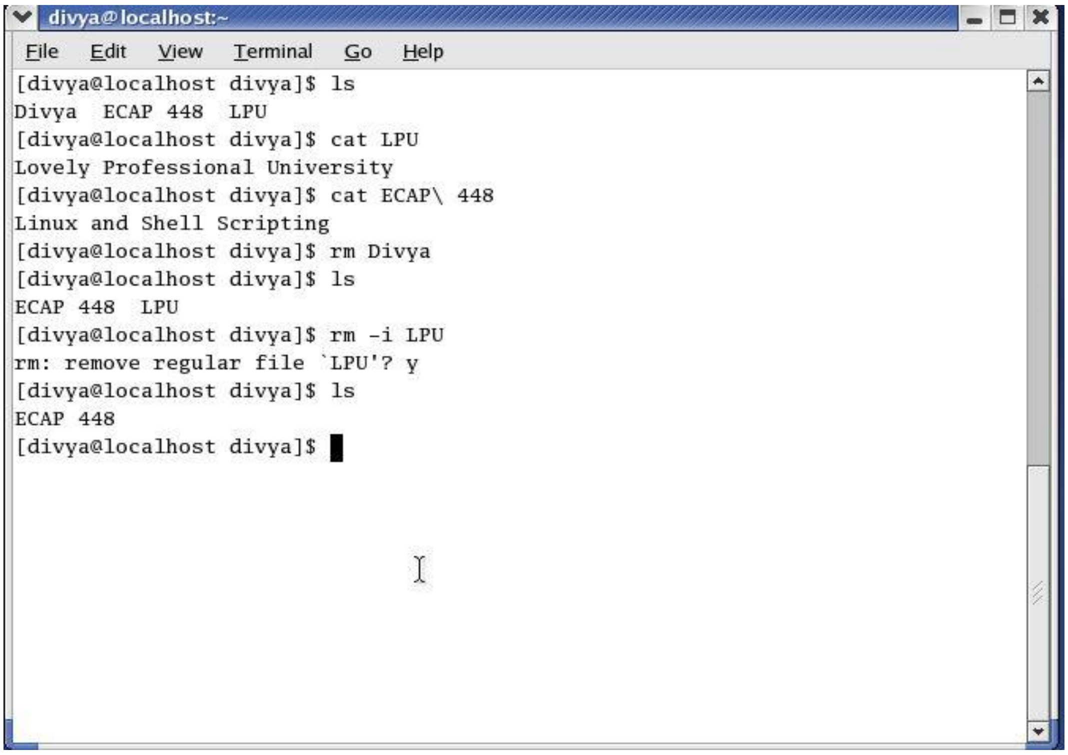


```
[divya@localhost divya]$ ls
Divya  ECAP 448  LPU
[divya@localhost divya]$ cat LPU
Lovely Professional University
[divya@localhost divya]$ cat ECAP\ 448
Linux and Shell Scripting
[divya@localhost divya]$ rm Divya
[divya@localhost divya]$ ls
ECAP 448  LPU
[divya@localhost divya]$ 
```

When you follow rm with the –i option and the name of the file you want to delete, rm displays the name of the file and then waits for you to respond with y (yes) before it deletes the file. It does not delete the file if you respond with a string that begins with a character other than y.



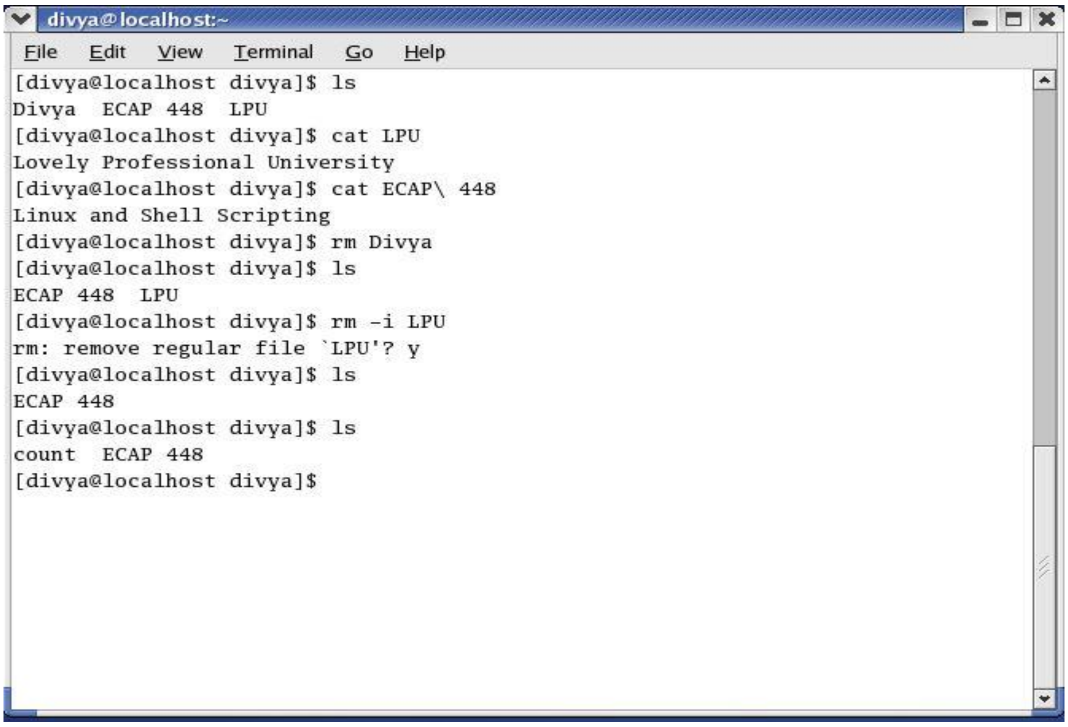
```
[divya@localhost divya]$ ls
Divya  ECAP 448  LPU
[divya@localhost divya]$ cat LPU
Lovely Professional University
[divya@localhost divya]$ cat ECAP\ 448
Linux and Shell Scripting
[divya@localhost divya]$ rm Divya
[divya@localhost divya]$ ls
ECAP 448  LPU
[divya@localhost divya]$ rm -i LPU
rm: remove regular file `LPU'? y
```



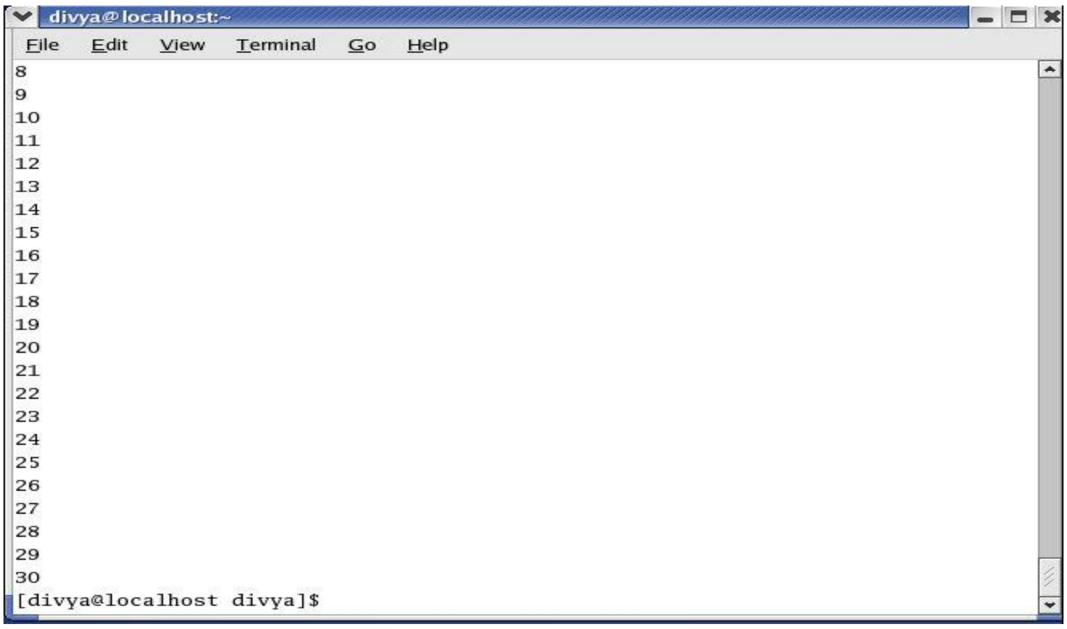
```
[divya@localhost divya]$ ls
Divya  ECAP 448  LPU
[divya@localhost divya]$ cat LPU
Lovely Professional University
[divya@localhost divya]$ cat ECAP\ 448
Linux and Shell Scripting
[divya@localhost divya]$ rm Divya
[divya@localhost divya]$ ls
ECAP 448  LPU
[divya@localhost divya]$ rm -i LPU
rm: remove regular file `LPU'? y
[divya@localhost divya]$ ls
ECAP 448
[divya@localhost divya]$
```

less Is more: Display a Text File One Screen at a Time

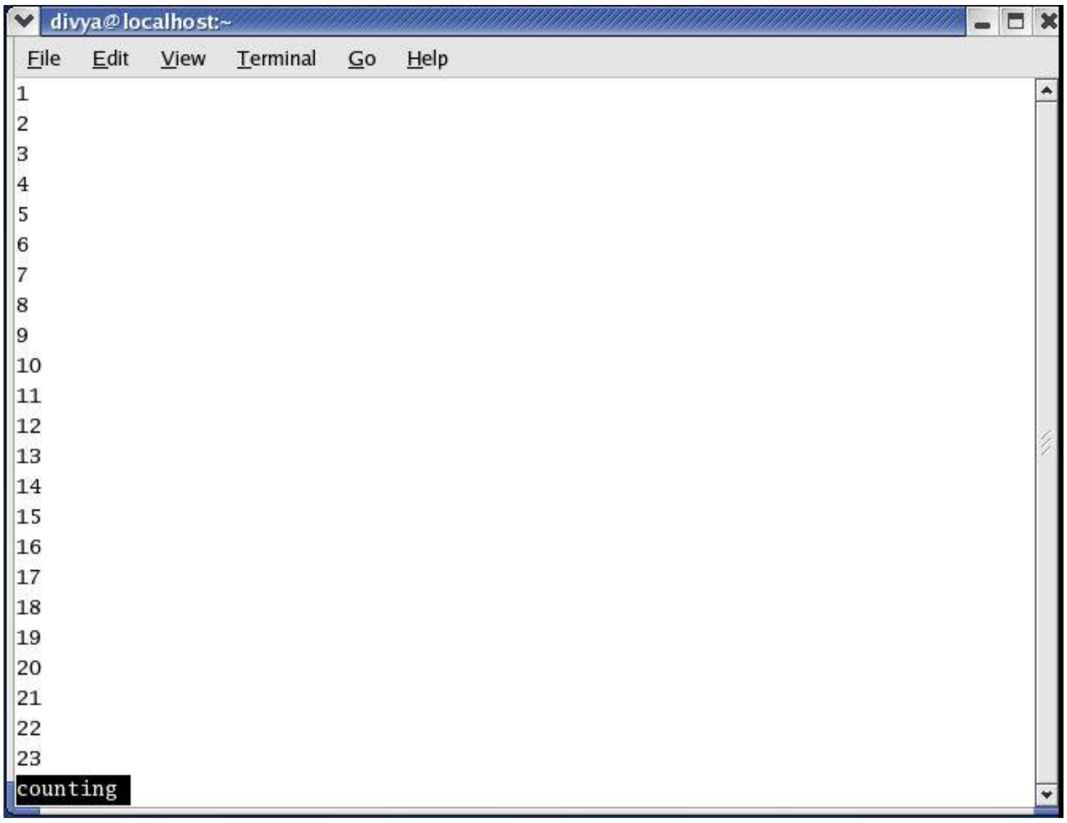
It displays a text file one screen at a time. When you want to view a file that is longer than one screen, you can use either the less utility or the more utility. Each of these utilities pauses after displaying a screen of text; press the SPACE bar to display the next screen of text. Because these utilities show one page at a time, they are called pagers. Although less and more are very similar, they have subtle differences. At the end of the file, for example, less displays an END message and waits for you to press q before returning you to the shell. In contrast, more returns you directly to the shell. While using both utilities you can press h to display a Help screen that lists commands you can use while paging through a file.



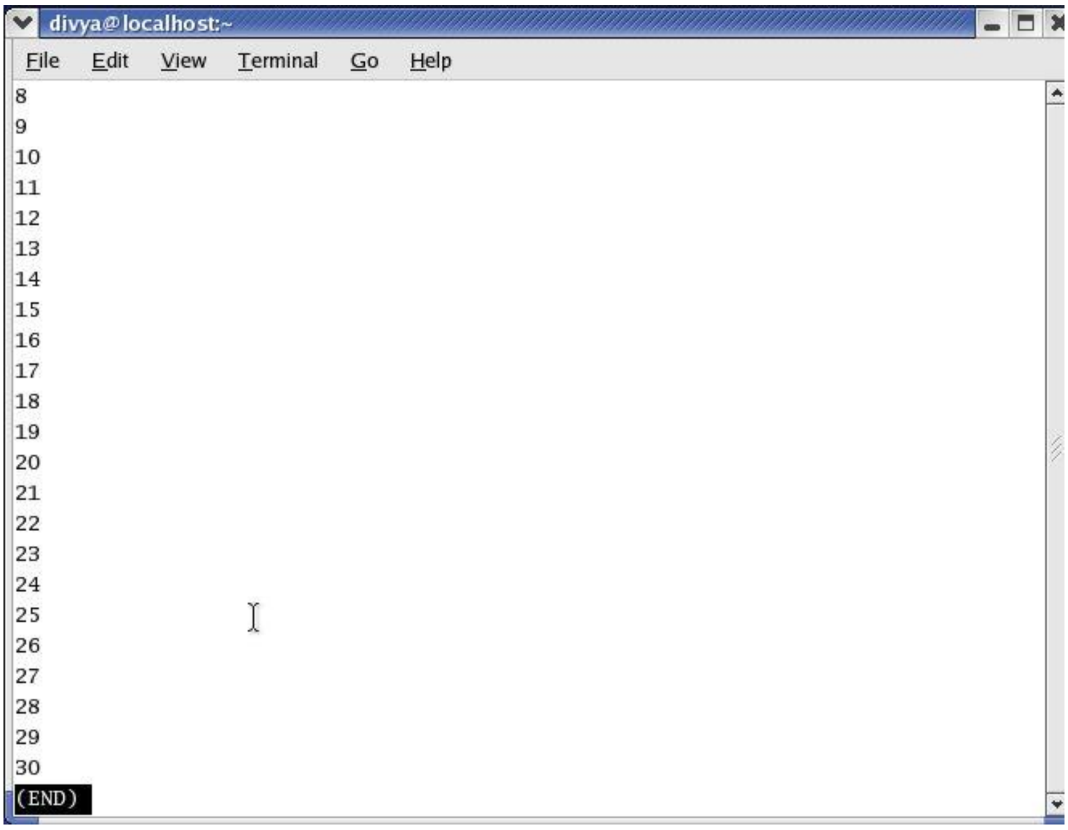
```
[divya@localhost divya]$ ls
Divya  ECAP 448  LPU
[divya@localhost divya]$ cat LPU
Lovely Professional University
[divya@localhost divya]$ cat ECAP\ 448
Linux and Shell Scripting
[divya@localhost divya]$ rm Divya
[divya@localhost divya]$ ls
ECAP 448  LPU
[divya@localhost divya]$ rm -i LPU
rm: remove regular file `LPU'? y
[divya@localhost divya]$ ls
ECAP 448
[divya@localhost divya]$ ls
count  ECAP 448
[divya@localhost divya]$
```



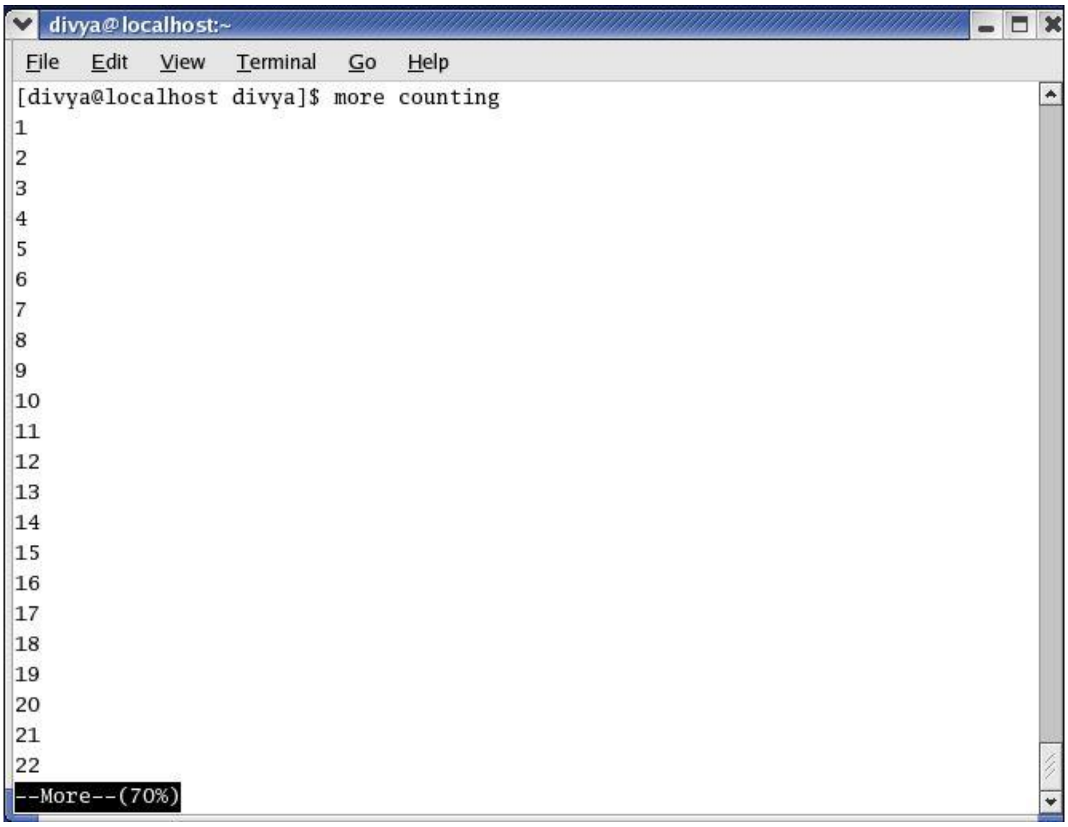
```
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
[divya@localhost divya]$
```



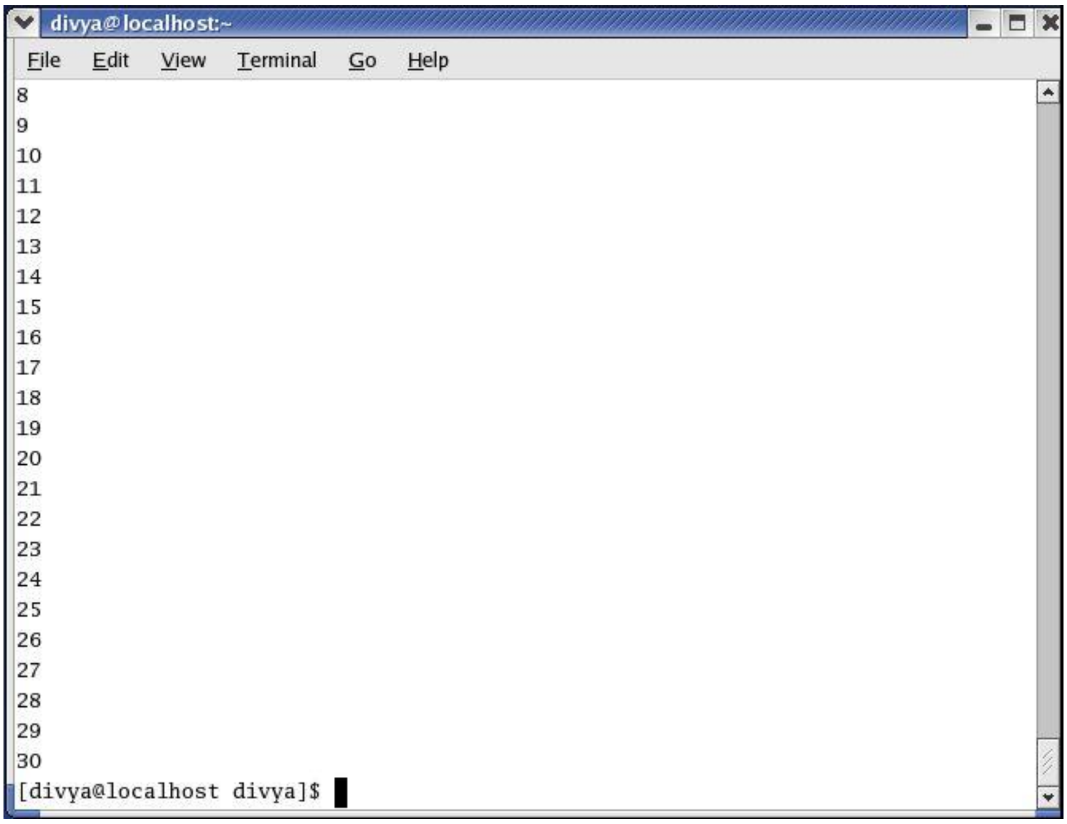
```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
counting
```



```
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
(END)
```



```
[divya@localhost divya]$ more counting
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
--More--(70%)
```



```
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
[divya@localhost divya]$
```

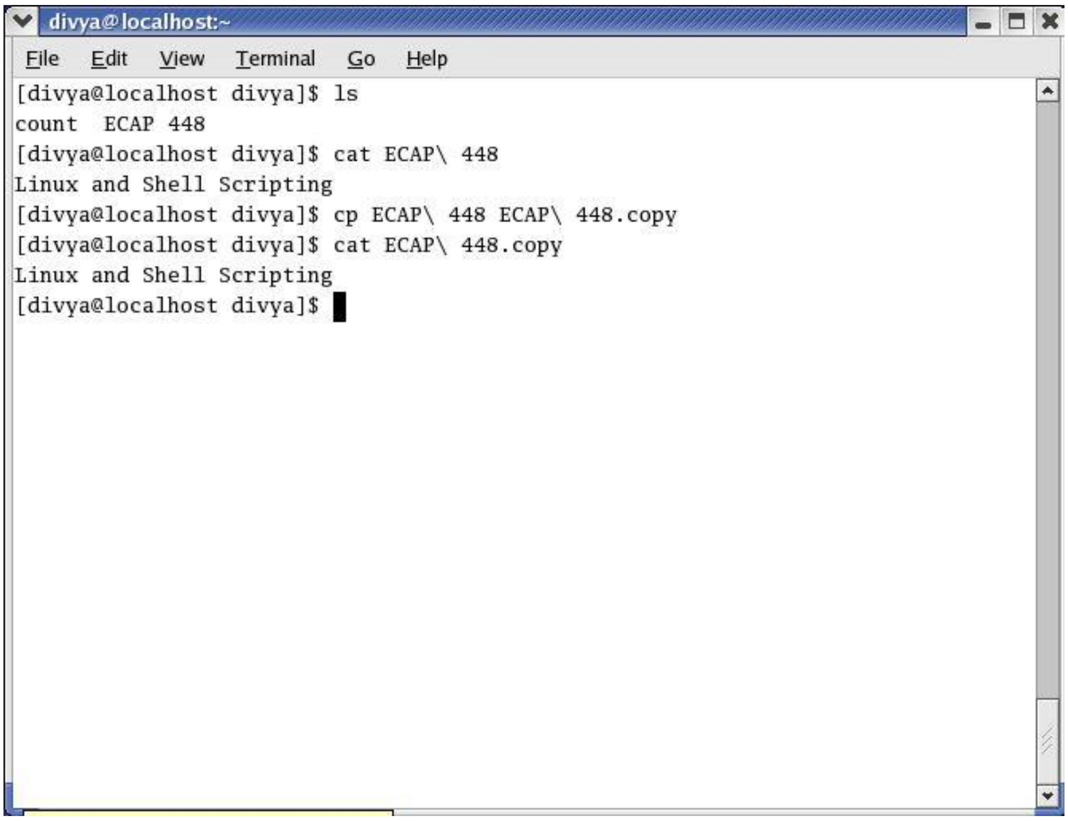
5.2 Working With Files

There are various utilities which we can use when we are working with files. These are:

- cp
- mv
- grep
- head
- tail
- sort
- uniq
- diff
- file

cp: Copies a File

The cp (copy) utility makes a copy of a file. This utility can copy any file, including text and executable program (binary) files. You can use cp to make a backup copy of a file or a copy to experiment with. The cp command line uses the following syntax to specify source and destination files: **cp source-file destination-file**

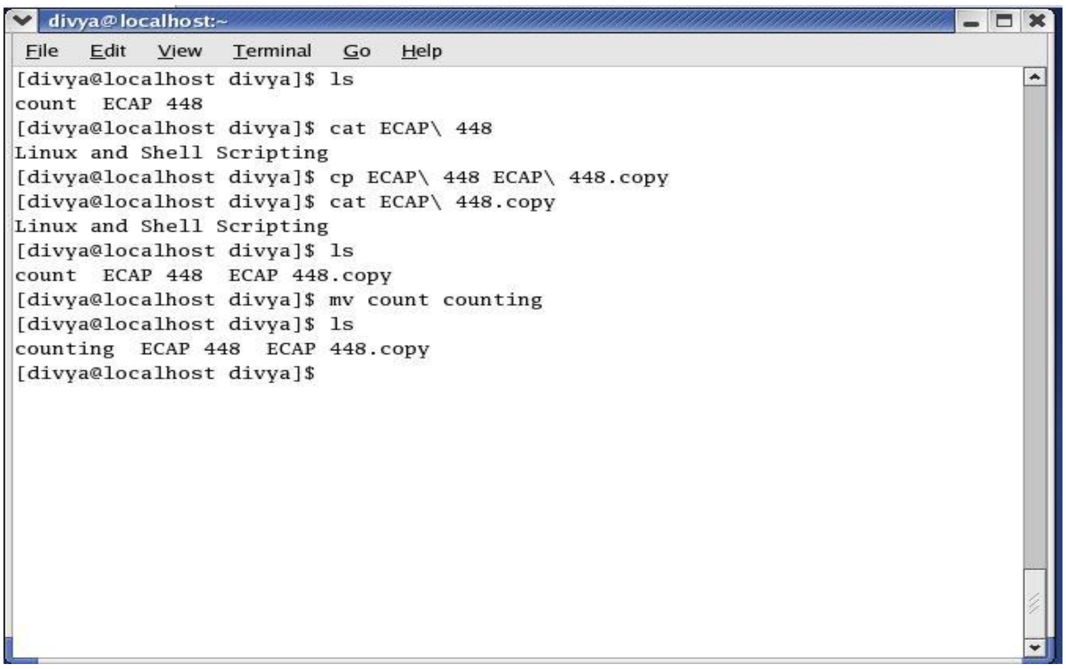


```
[divya@localhost divya]$ ls
count  ECAP 448
[divya@localhost divya]$ cat ECAP\ 448
Linux and Shell Scripting
[divya@localhost divya]$ cp ECAP\ 448 ECAP\ 448.copy
[divya@localhost divya]$ cat ECAP\ 448.copy
Linux and Shell Scripting
[divya@localhost divya]$
```

cp can destroy a file: if the destination-file exists before you give a cp command, cp overwrites it. The cp –i (interactive) option prompts you before it overwrites a file.

mv: Changes the Name of a File

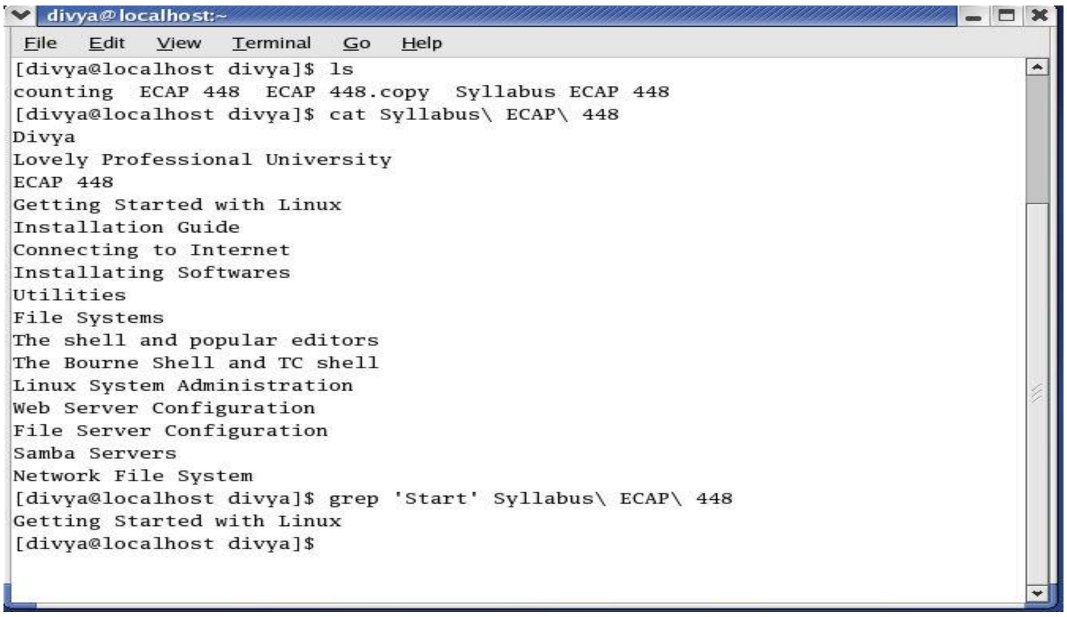
The mv (move) utility can rename a file without making a copy of it. The mv command line specifies an existing file and a new filename using the same syntax as cp: **mv existing-filename new-filename**



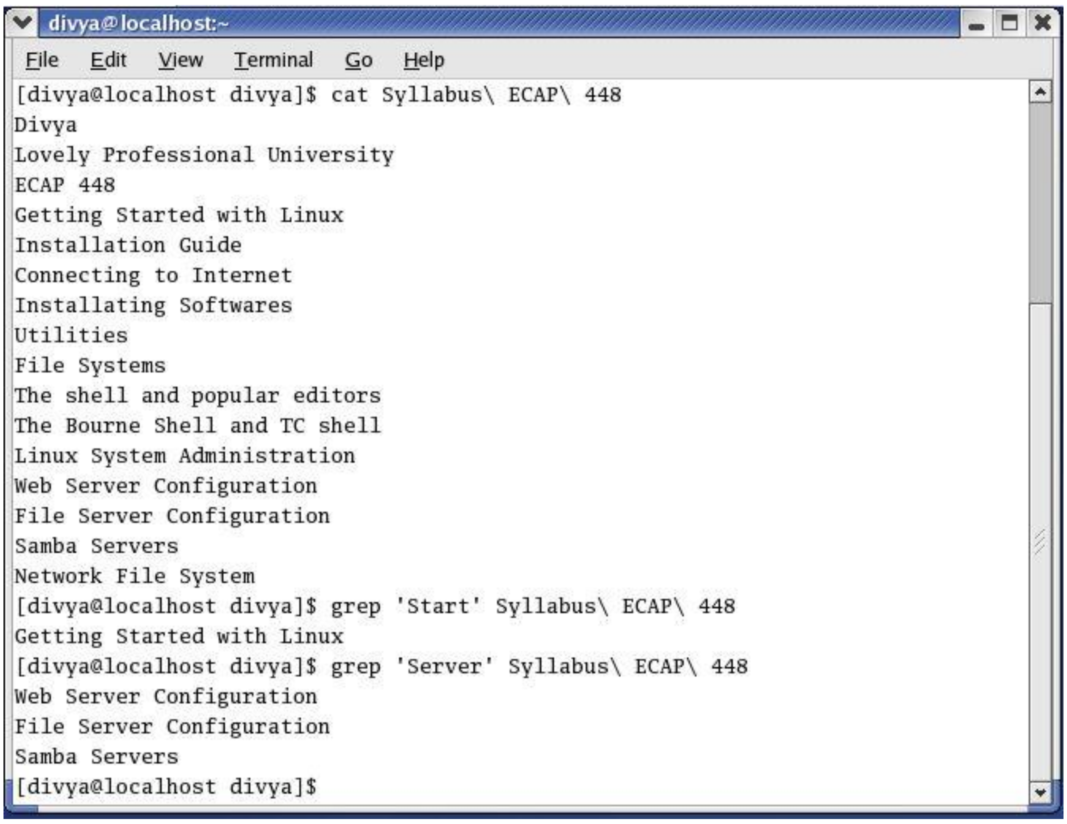
```
[divya@localhost divya]$ ls
count  ECAP 448
[divya@localhost divya]$ cat ECAP\ 448
Linux and Shell Scripting
[divya@localhost divya]$ cp ECAP\ 448 ECAP\ 448.copy
[divya@localhost divya]$ cat ECAP\ 448.copy
Linux and Shell Scripting
[divya@localhost divya]$ ls
count  ECAP 448  ECAP 448.copy
[divya@localhost divya]$ mv count counting
[divya@localhost divya]$ ls
counting  ECAP 448  ECAP 448.copy
[divya@localhost divya]$
```

grep: Searches for a String

The grep utility searches through one or more files to see whether any contain a specified string of characters. This utility does not change the file it searches but simply displays each line that contains the string.



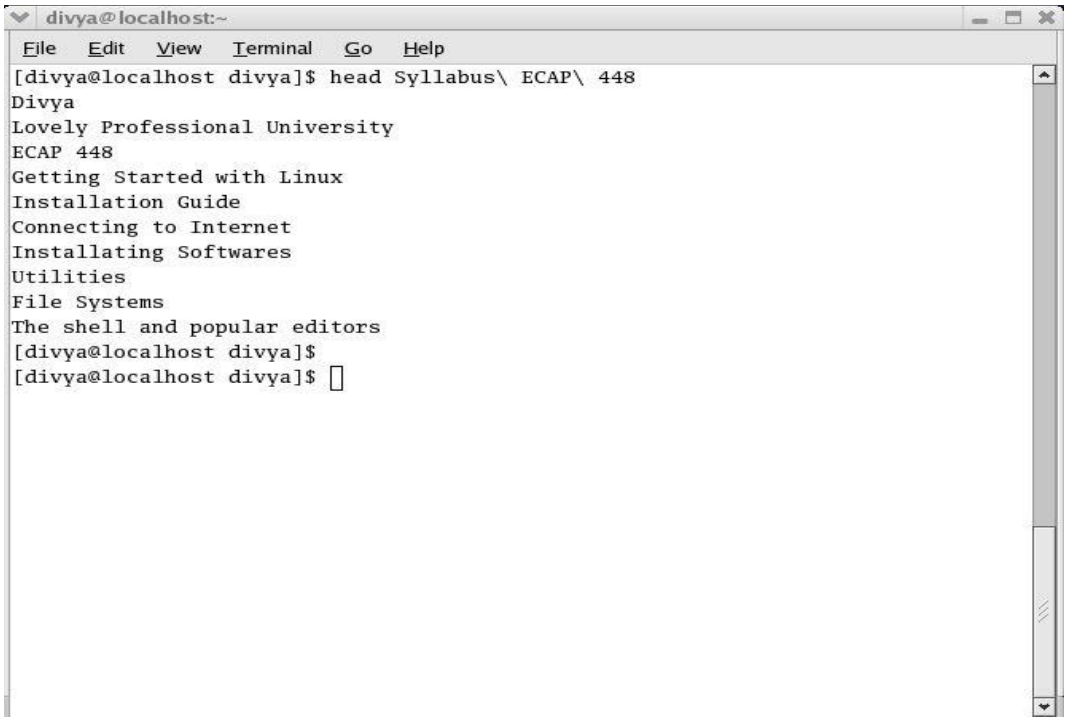
```
[divya@localhost divya]$ ls
counting  ECAP 448  ECAP 448.copy  Syllabus ECAP 448
[divya@localhost divya]$ cat Syllabus\ ECAP\ 448
Divya
Lovely Professional University
ECAP 448
Getting Started with Linux
Installation Guide
Connecting to Internet
Installating Softwares
Utilities
File Systems
The shell and popular editors
The Bourne Shell and TC shell
Linux System Administration
Web Server Configuration
File Server Configuration
Samba Servers
Network File System
[divya@localhost divya]$ grep 'Start' Syllabus\ ECAP\ 448
Getting Started with Linux
[divya@localhost divya]$
```



```
[divya@localhost divya]$ cat Syllabus\ ECAP\ 448
Divya
Lovely Professional University
ECAP 448
Getting Started with Linux
Installation Guide
Connecting to Internet
Installating Softwares
Utilities
File Systems
The shell and popular editors
The Bourne Shell and TC shell
Linux System Administration
Web Server Configuration
File Server Configuration
Samba Servers
Network File System
[divya@localhost divya]$ grep 'Start' Syllabus\ ECAP\ 448
Getting Started with Linux
[divya@localhost divya]$ grep 'Server' Syllabus\ ECAP\ 448
Web Server Configuration
File Server Configuration
Samba Servers
[divya@localhost divya]$
```

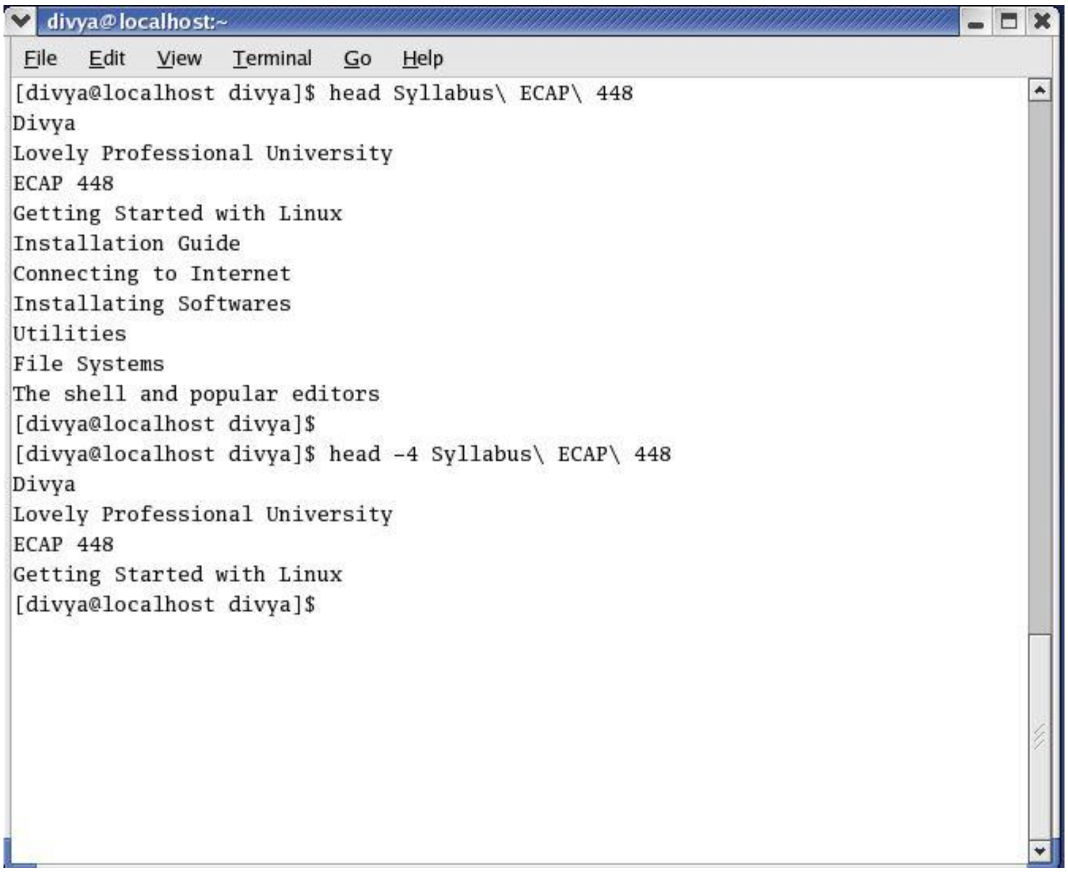
head: Displays the Beginning of a File

By default, the head utility displays the first ten lines of a file. For example, if you have a file named months that lists the 12 months of the year in calendar order, one to a line, then head displays Jan through Oct.



```
[divya@localhost divya]$ head Syllabus\ ECAP\ 448
Divya
Lovely Professional University
ECAP 448
Getting Started with Linux
Installation Guide
Connecting to Internet
Installating Softwares
Utilities
File Systems
The shell and popular editors
[divya@localhost divya]$
[divya@localhost divya]$
```

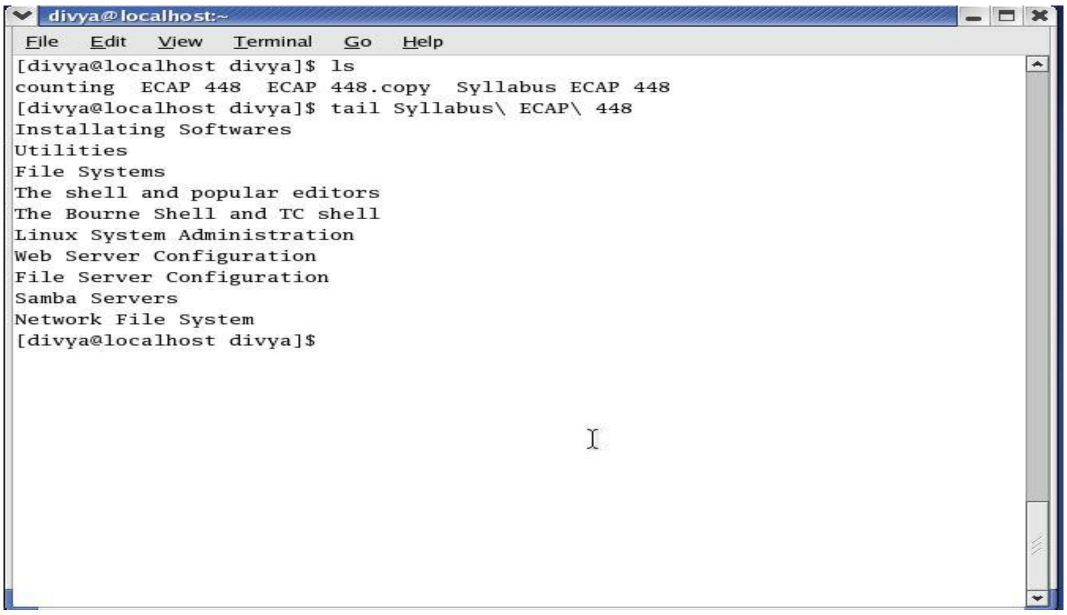
This utility can display any number of lines, so you can use it to look at only the first line of a file, at a full screen, or even more. To specify the number of lines displayed, include a hyphen followed by the number of lines you want head to display.



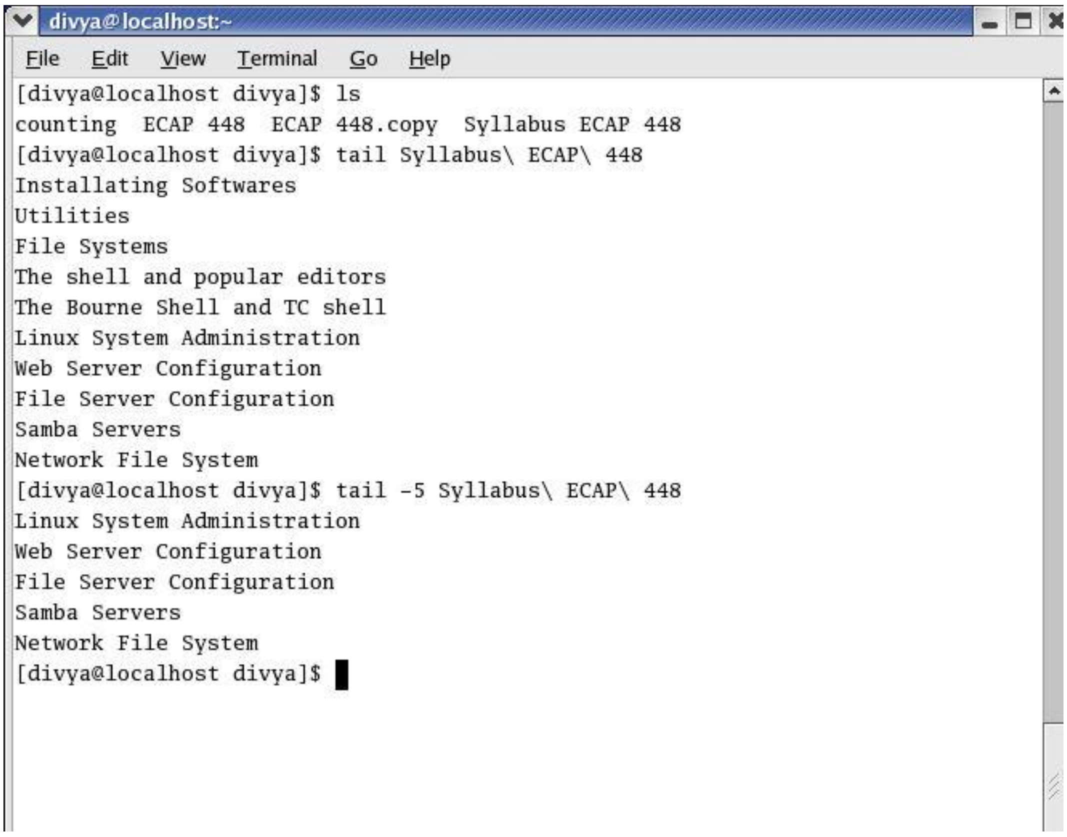
```
[divya@localhost divya]$ head Syllabus\ ECAP\ 448
Divya
Lovely Professional University
ECAP 448
Getting Started with Linux
Installation Guide
Connecting to Internet
Installating Softwares
Utilities
File Systems
The shell and popular editors
[divya@localhost divya]$
[divya@localhost divya]$ head -4 Syllabus\ ECAP\ 448
Divya
Lovely Professional University
ECAP 448
Getting Started with Linux
[divya@localhost divya]$
```

tail: Displays the End of a File

The tail utility is like head but by default displays the last ten lines of a file. Depending on how you invoke it, this utility can display fewer or more than ten lines.



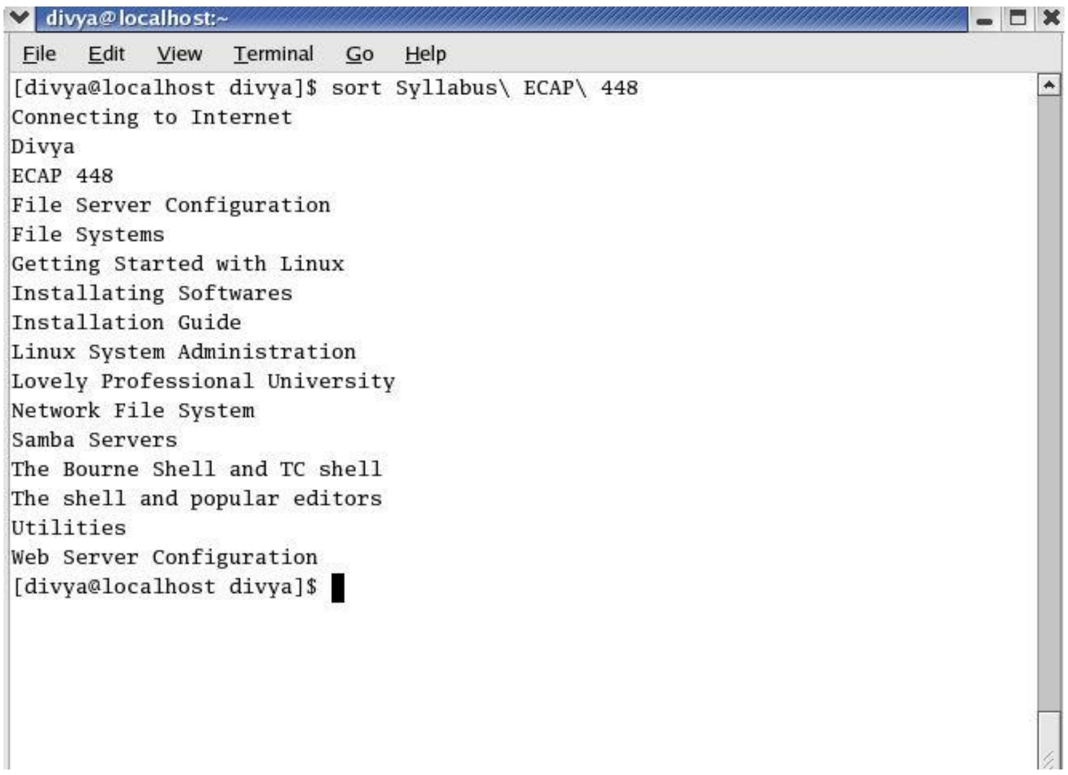
```
[divya@localhost divya]$ ls
counting  ECAP 448  ECAP 448.copy  Syllabus ECAP 448
[divya@localhost divya]$ tail Syllabus\ ECAP\ 448
Installating Softwares
Utilities
File Systems
The shell and popular editors
The Bourne Shell and TC shell
Linux System Administration
Web Server Configuration
File Server Configuration
Samba Servers
Network File System
[divya@localhost divya]$
```



```
[divya@localhost divya]$ ls
counting  ECAP 448  ECAP 448.copy  Syllabus ECAP 448
[divya@localhost divya]$ tail Syllabus\ ECAP\ 448
Installating Softwares
Utilities
File Systems
The shell and popular editors
The Bourne Shell and TC shell
Linux System Administration
Web Server Configuration
File Server Configuration
Samba Servers
Network File System
[divya@localhost divya]$ tail -5 Syllabus\ ECAP\ 448
Linux System Administration
Web Server Configuration
File Server Configuration
Samba Servers
Network File System
[divya@localhost divya]$
```

sort: Displays a File in Order

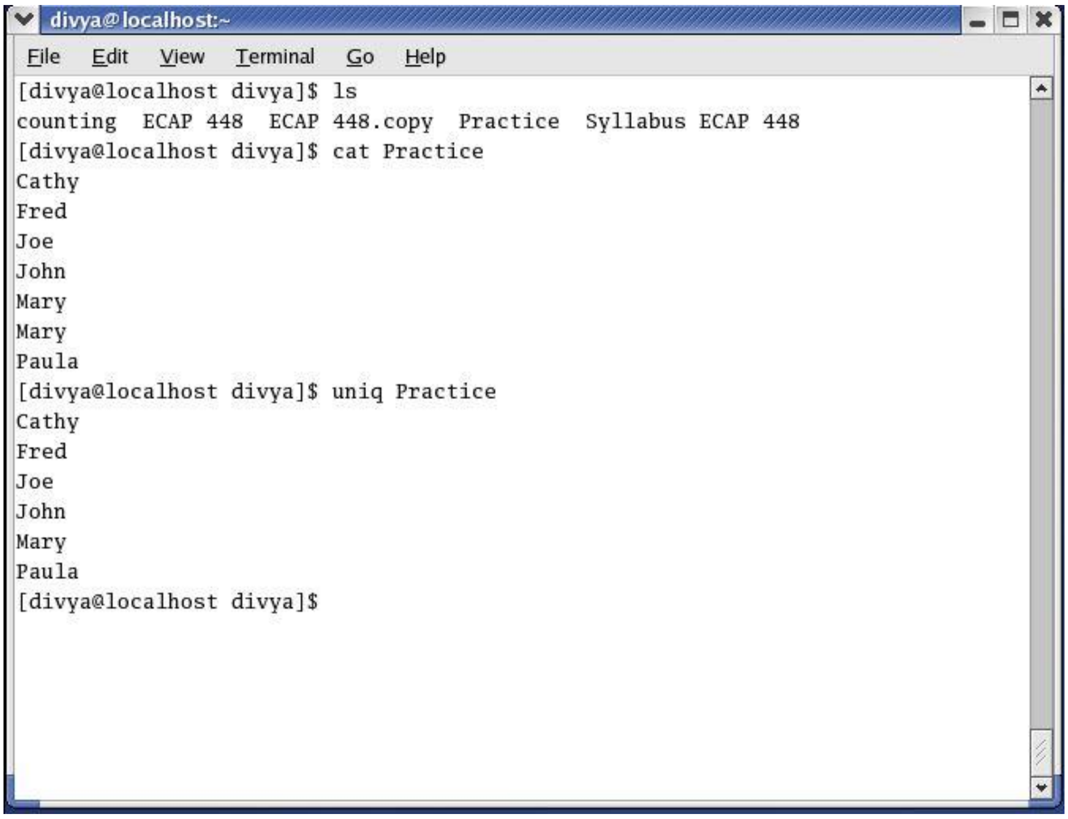
The sort utility displays the contents of a file in order by lines; it does not change the original file.



```
[divya@localhost divya]$ sort Syllabus\ ECAP\ 448
Connecting to Internet
Divya
ECAP 448
File Server Configuration
File Systems
Getting Started with Linux
Installating Softwares
Installation Guide
Linux System Administration
Lovely Professional University
Network File System
Samba Servers
The Bourne Shell and TC shell
The shell and popular editors
Utilities
Web Server Configuration
[divya@localhost divya]$
```

uniq: Removes Duplicate Lines from a File

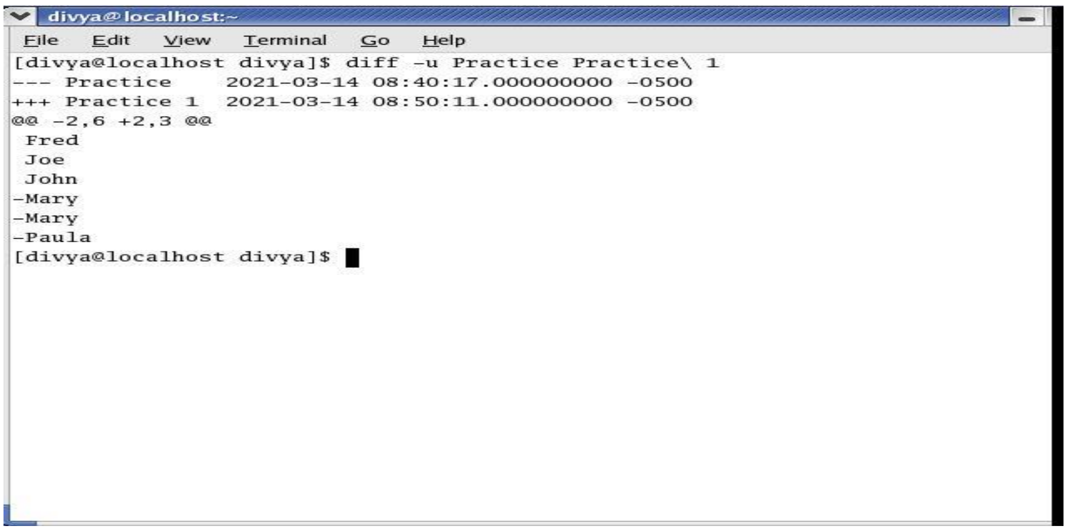
The uniq (unique) utility displays a file, skipping adjacent duplicate lines, but does not change the original file. If a file contains a list of names and has two successive entries for the same person, uniq skips the duplicate line. If a file is sorted before it is processed by uniq, this utility ensures that no two lines in the file are the same.



```
[divya@localhost divya]$ ls
counting  ECAP 448  ECAP 448.copy  Practice  Syllabus ECAP 448
[divya@localhost divya]$ cat Practice
Cathy
Fred
Joe
John
Mary
Mary
Paula
[divya@localhost divya]$ uniq Practice
Cathy
Fred
Joe
John
Mary
Paula
[divya@localhost divya]$
```

diff: Compares Two Files

The diff (difference) utility compares two files and displays a list of the differences between them. This utility does not change either file; it is useful when you want to compare two versions of a letter or a report or two versions of the source code for a program. The diff utility with the –u (unified output format) option first displays two lines indicating which of the files you are comparing will be denoted by a plus sign (+) and which by a minus sign (–).



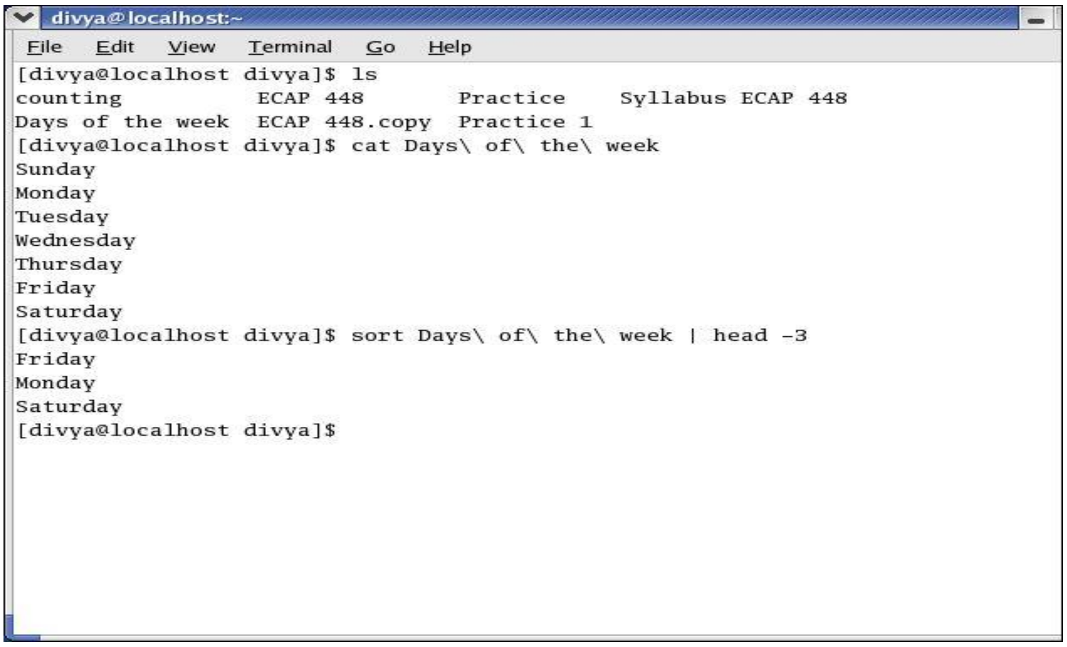
```
[divya@localhost divya]$ diff -u Practice Practice\ 1
--- Practice    2021-03-14 08:40:17.000000000 -0500
+++ Practice 1  2021-03-14 08:50:11.000000000 -0500
@@ -2,6 +2,3 @@
 Fred
 Joe
 John
-Mary
-Mary
-Paula
[divya@localhost divya]$
```

file: Identifies the Contents of a File

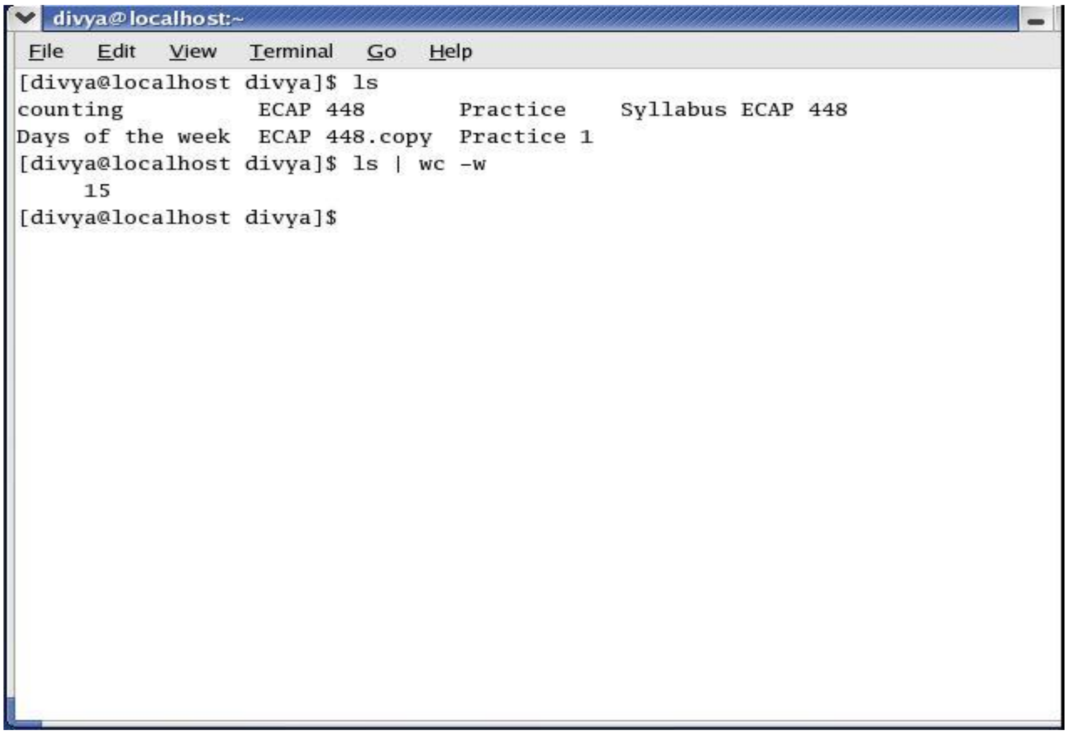
You can use the file utility to learn about the contents of a file without having to open and examine the file yourself.

| (Pipe): Communicates Between Processes

A process is the execution of a command by Linux. Communication between processes is one of the hallmarks of both UNIX and Linux. A pipe (written as a vertical bar [|] on the command line and appearing as a solid or broken vertical line on a keyboard) provides the simplest form of this kind of communication. Simply put, a pipe takes the output of one utility and sends that output as input to another utility.



```
[divya@localhost divya]$ ls
counting          ECAP 448       Practice     Syllabus ECAP 448
Days of the week  ECAP 448.copy  Practice 1
[divya@localhost divya]$ cat Days\ of\ the\ week
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
[divya@localhost divya]$ sort Days\ of\ the\ week | head -3
Friday
Monday
Saturday
[divya@localhost divya]$
```



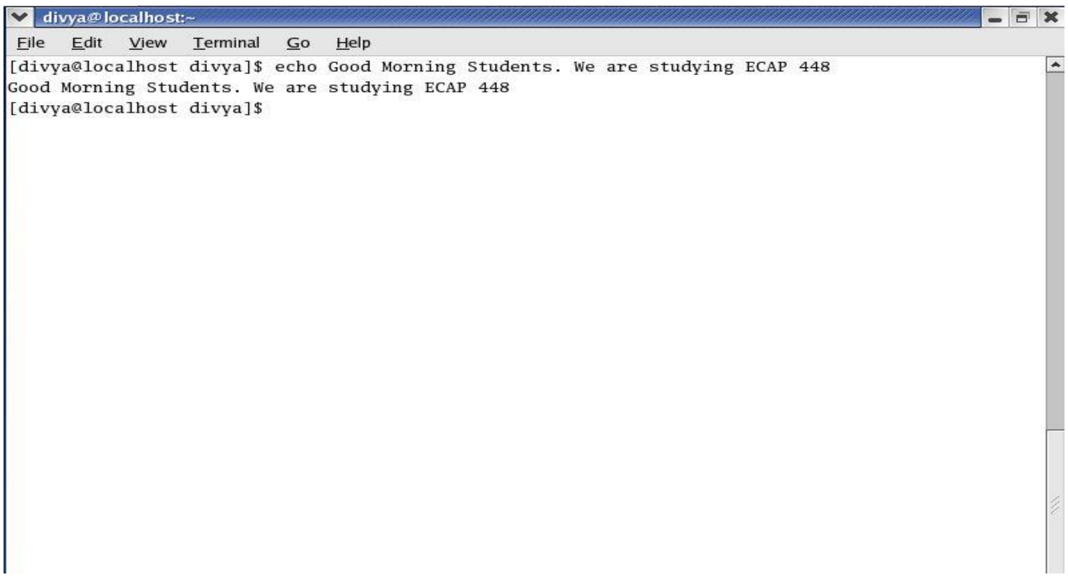
```
[divya@localhost divya]$ ls
counting          ECAP 448       Practice    Syllabus ECAP 448
Days of the week  ECAP 448.copy  Practice 1
[divya@localhost divya]$ ls | wc -w
     15
[divya@localhost divya]$
```

5.3 Four More Utilities

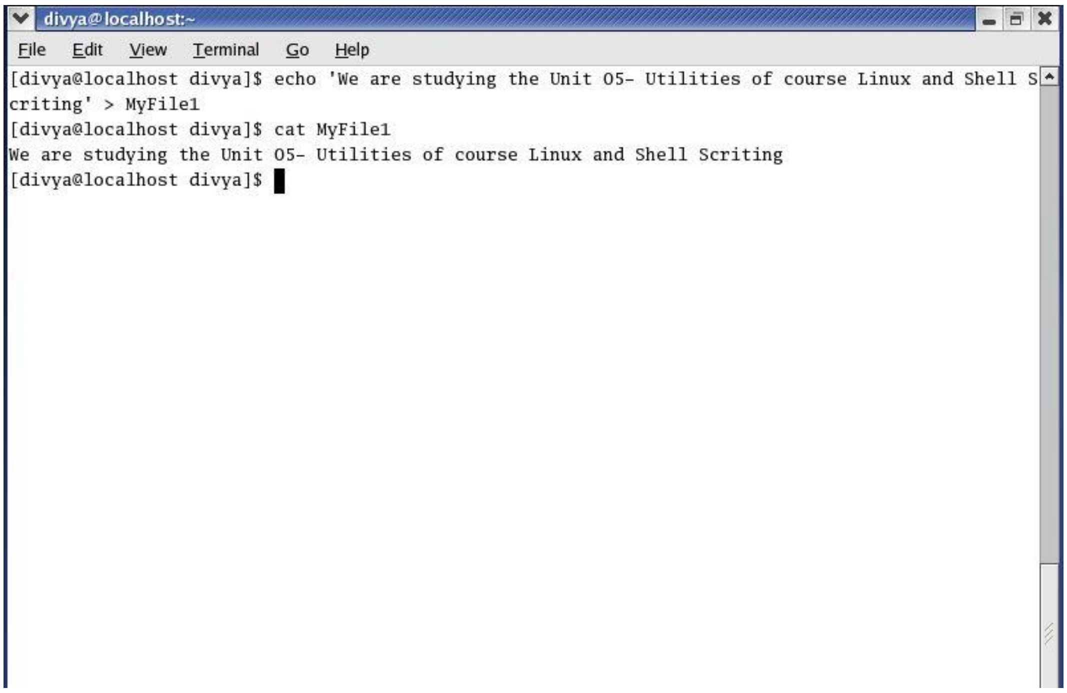
- echo
- date
- script
- unix2dos

echo: Displays Text

The echo utility copies the characters you type on the command line after echo to the screen. You can also send messages from shell scripts to the screen.



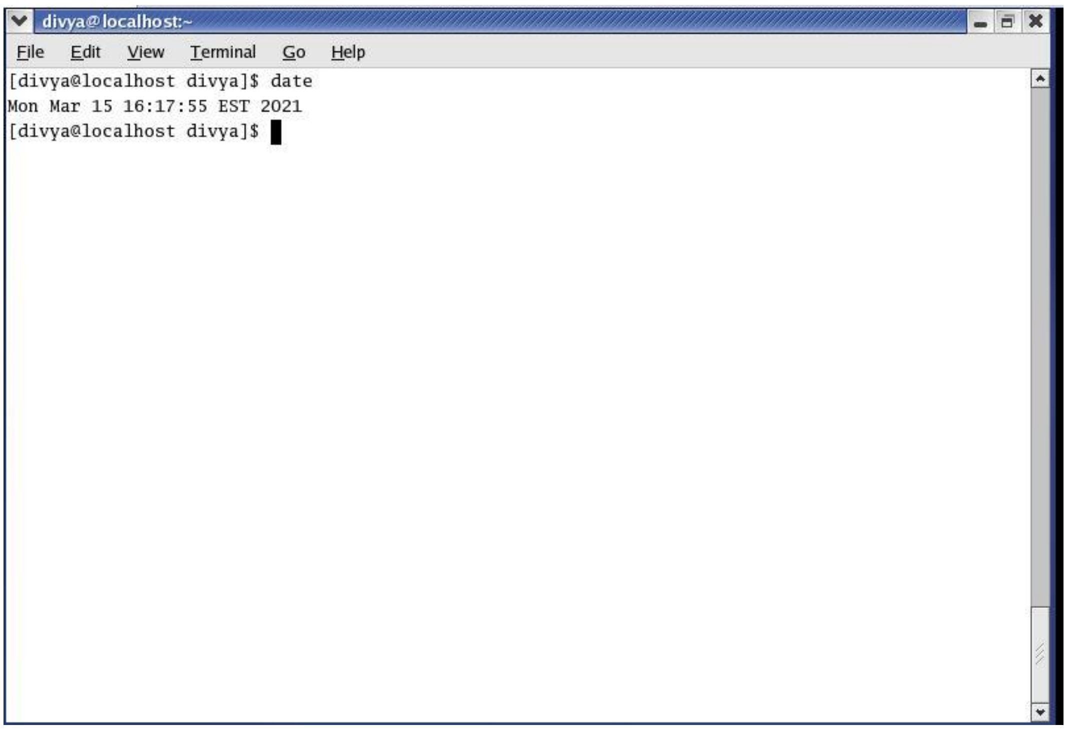
```
[divya@localhost divya]$ echo Good Morning Students. We are studying ECAP 448
Good Morning Students. We are studying ECAP 448
[divya@localhost divya]$
```



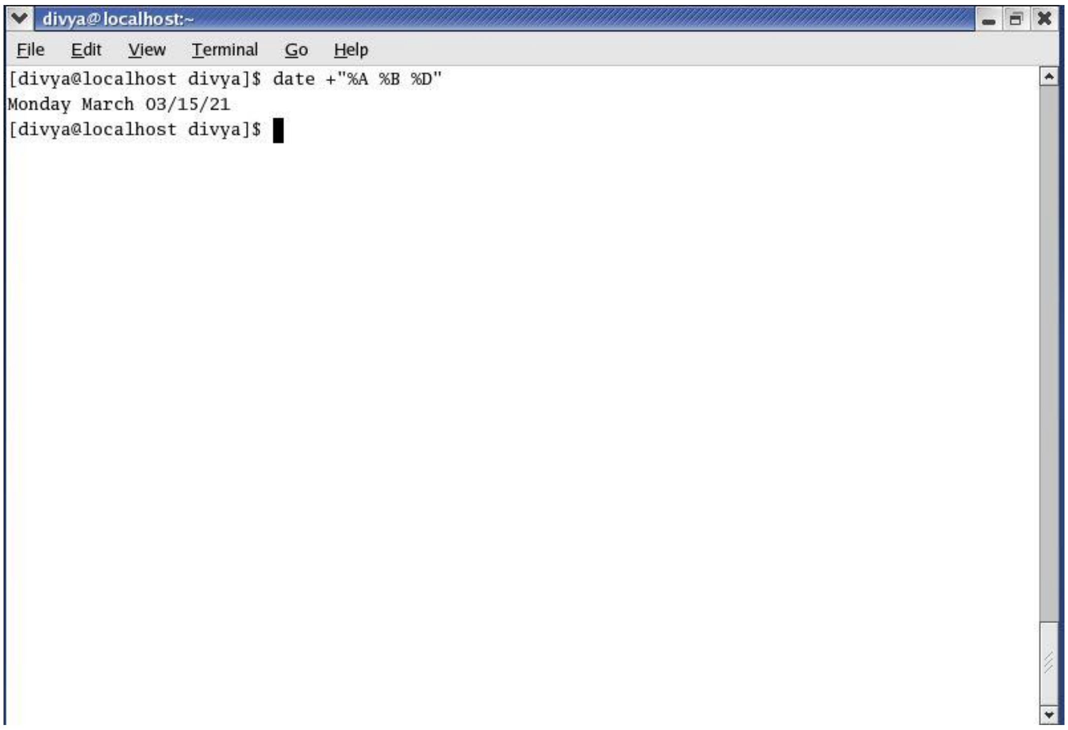
```
[divya@localhost divya]$ echo 'We are studying the Unit 05- Utilities of course Linux and Shell S
criting' > MyFile1
[divya@localhost divya]$ cat MyFile1
We are studying the Unit 05- Utilities of course Linux and Shell Scriting
[divya@localhost divya]$
```

date: Displays the Time and Date

The date utility displays the current date and time. You can choose the format and select the contents of the output of date.



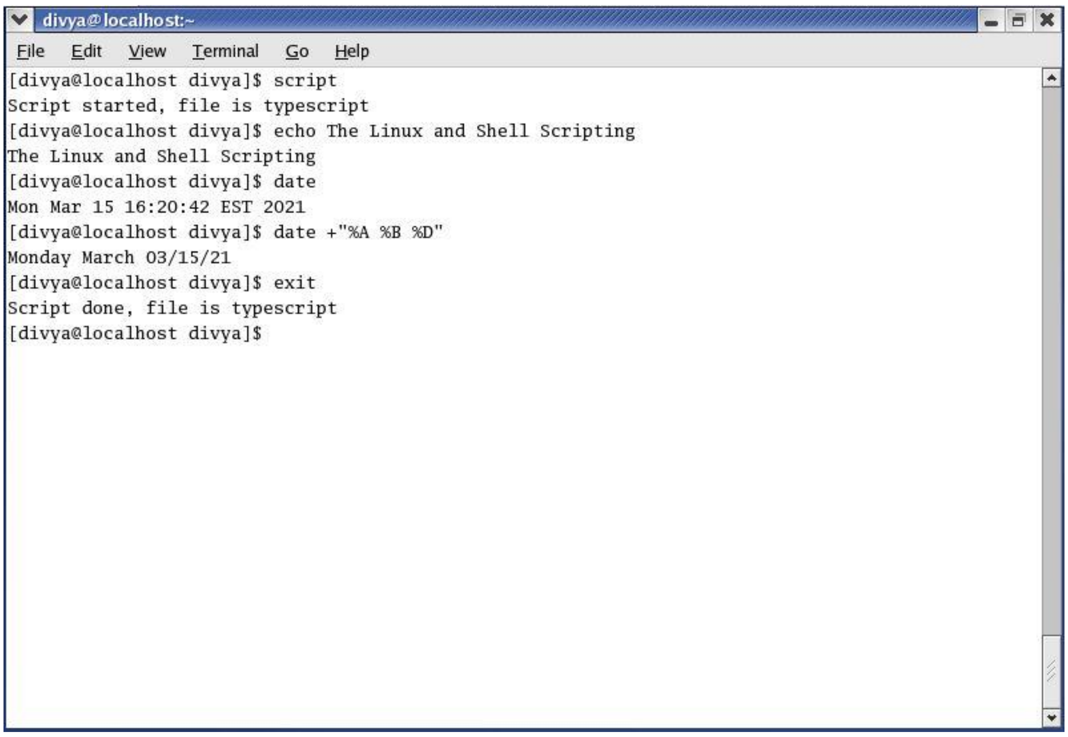
```
[divya@localhost divya]$ date
Mon Mar 15 16:17:55 EST 2021
[divya@localhost divya]$
```



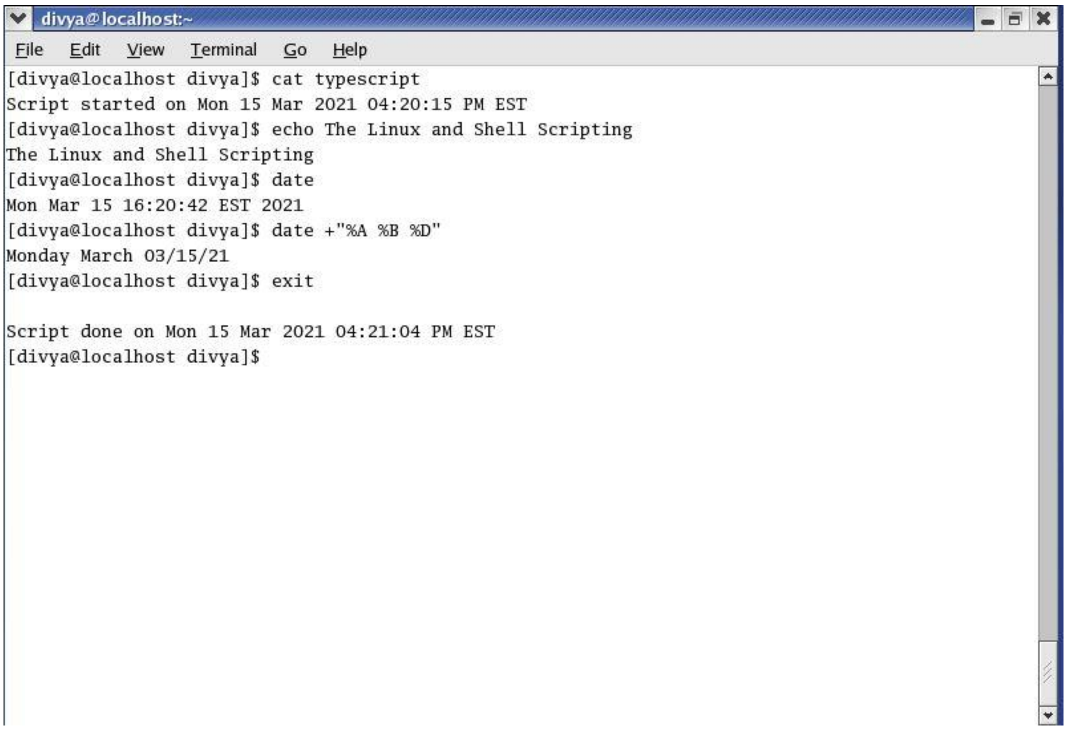
```
[divya@localhost divya]$ date +"%A %B %D"
Monday March 03/15/21
[divya@localhost divya]$ 
```

script: Records a Shell Session

The script utility records all or part of a login session, including your input and the system's responses. This utility is useful only from character-based devices. By default, script captures the session in a file named typescript. To specify a different filename, follow the script command with a SPACE and the filename



```
[divya@localhost divya]$ script
Script started, file is typescript
[divya@localhost divya]$ echo The Linux and Shell Scripting
The Linux and Shell Scripting
[divya@localhost divya]$ date
Mon Mar 15 16:20:42 EST 2021
[divya@localhost divya]$ date +"%A %B %D"
Monday March 03/15/21
[divya@localhost divya]$ exit
Script done, file is typescript
[divya@localhost divya]$
```



```
[divya@localhost divya]$ cat typescript
Script started on Mon 15 Mar 2021 04:20:15 PM EST
[divya@localhost divya]$ echo The Linux and Shell Scripting
The Linux and Shell Scripting
[divya@localhost divya]$ date
Mon Mar 15 16:20:42 EST 2021
[divya@localhost divya]$ date +"%A %B %D"
Monday March 03/15/21
[divya@localhost divya]$ exit

Script done on Mon 15 Mar 2021 04:21:04 PM EST
[divya@localhost divya]$
```

todos/unix2dos: Converts Linux Files to Windows Format

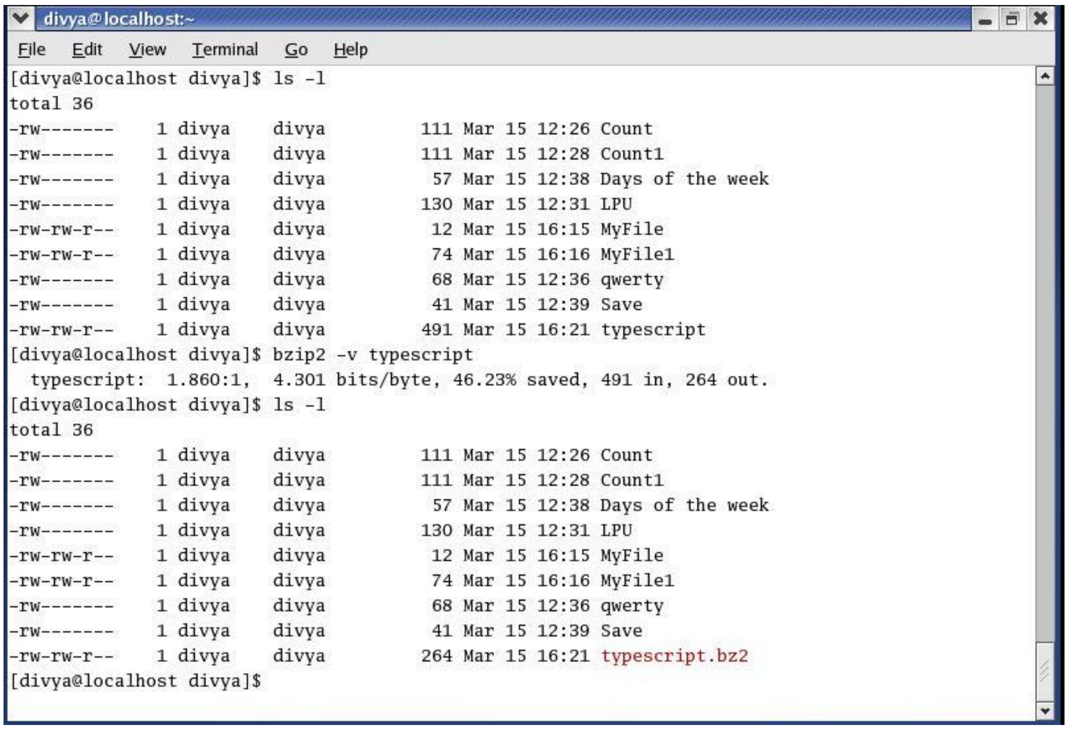
If you want to share a text file you created on a Linux system with someone on a system running Windows, you need to convert the file before the person on the other system can read it easily. The todos (to DOS; part of the tofrodos package) or unix2dos (UNIX to DOS; part of the unix2dos package) utility converts a Linux text file so it can be read on a Windows system. You can use the fromdos (from DOS; part of the tofrodos package) or dos2unix (DOS to UNIX; part of the dos2unix package) utility to convert Windows files so they can be read on a Linux system.

5.4 Compressing and Archiving Files

Large files use a lot of disk space and take longer than smaller files to transfer from one system to another over a network. If you do not need to look at the contents of a large file often, you may want to save it on a CD, DVD, or another medium and remove it from the hard disk. If you have a continuing need for the file, retrieving a copy from another medium may be inconvenient. To reduce the amount of disk space a file occupies without removing the file, you can compress the file without losing any of the information it holds. Similarly, a single archive of several files packed into a larger file is easier to manipulate, upload, download, and email than multiple files. You may download compressed, archived files from the Internet.

bzip2: Compresses a File

The bzip2 utility compresses a file by analyzing it and recoding it more efficiently. The new version of the file looks completely different. In fact, because the new file contains many nonprinting characters, you cannot view it directly. The –v (verbose) option causes bzip2 to report how much it was able to reduce the size of the file.

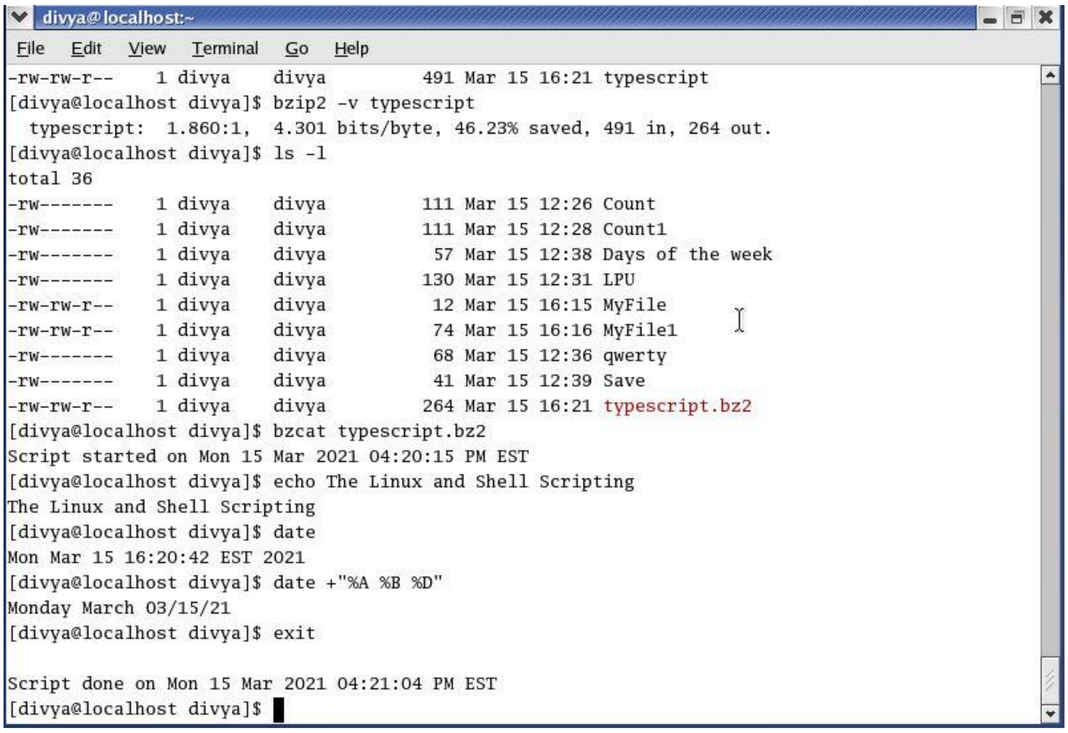


```
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya         111 Mar 15 12:26 Count
-rw-------    1 divya    divya         111 Mar 15 12:28 Count1
-rw-------    1 divya    divya          57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya         130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya          12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya          74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya          68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya          41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya         491 Mar 15 16:21 typescript
[divya@localhost divya]$ bzip2 -v typescript
  typescript:  1.860:1,  4.301 bits/byte, 46.23% saved, 491 in, 264 out.
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya         111 Mar 15 12:26 Count
-rw-------    1 divya    divya         111 Mar 15 12:28 Count1
-rw-------    1 divya    divya          57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya         130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya          12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya          74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya          68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya          41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya         264 Mar 15 16:21 typescript.bz2
[divya@localhost divya]$
```

The bzip2 utility also renamed the file, appending .bz2 to its name. This naming convention reminds you that the file is compressed; you would not want to display or print it, for example, without first decompressing it.

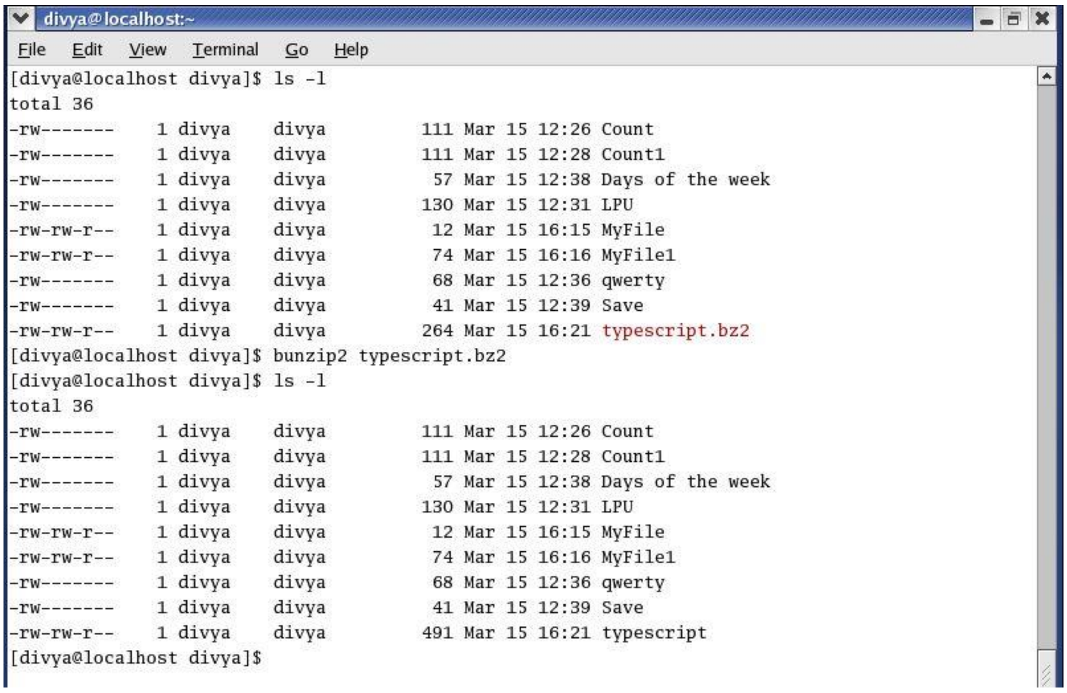
bunzip2 and bzcat: Decompress a File

The bzcat utility displays a file that has been compressed with bzip2. The equivalent of cat for .bz2 files, bzcat decompresses the compressed data and displays the decompressed data. Like cat, bzcat does not change the source file.



```
-rw-rw-r--    1 divya    divya         491 Mar 15 16:21 typescript
[divya@localhost divya]$ bzip2 -v typescript
  typescript:  1.860:1,  4.301 bits/byte, 46.23% saved, 491 in, 264 out.
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya         111 Mar 15 12:26 Count
-rw-------    1 divya    divya         111 Mar 15 12:28 Count1
-rw-------    1 divya    divya          57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya         130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya          12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya          74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya          68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya          41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya         264 Mar 15 16:21 typescript.bz2
[divya@localhost divya]$ bzcat typescript.bz2
Script started on Mon 15 Mar 2021 04:20:15 PM EST
[divya@localhost divya]$ echo The Linux and Shell Scripting
The Linux and Shell Scripting
[divya@localhost divya]$ date
Mon Mar 15 16:20:42 EST 2021
[divya@localhost divya]$ date +"%A %B %D"
Monday March 03/15/21
[divya@localhost divya]$ exit

Script done on Mon 15 Mar 2021 04:21:04 PM EST
[divya@localhost divya]$
```

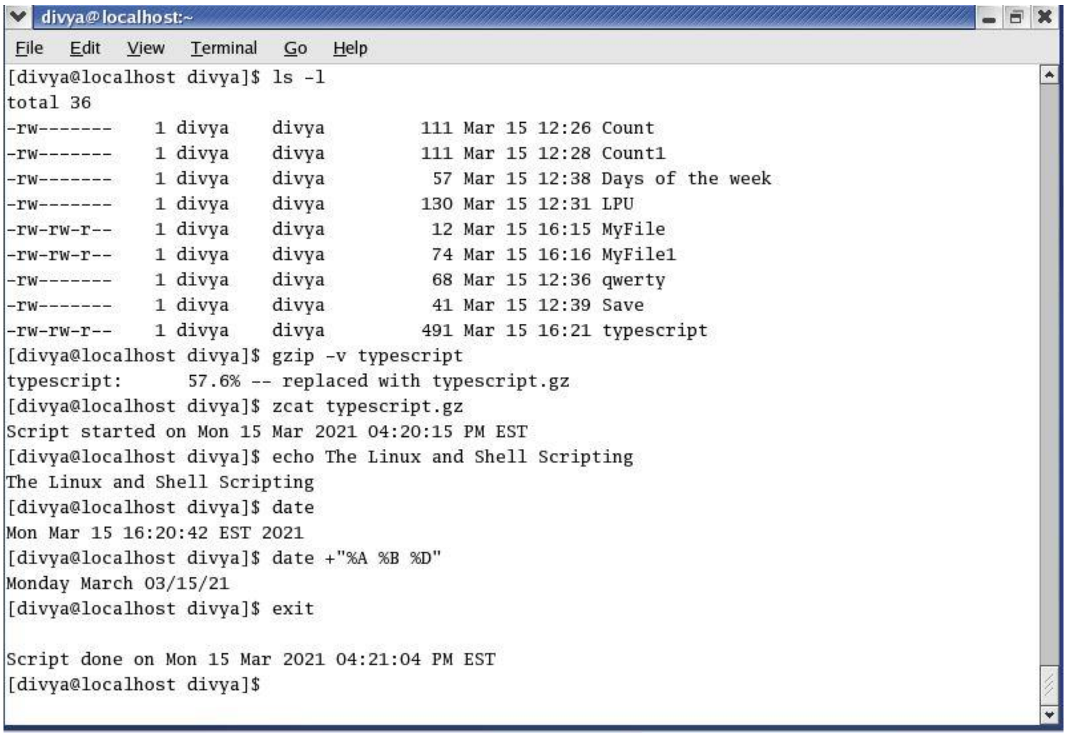
You can use the bunzip2 utility to restore a file that has been compressed with bzip2.



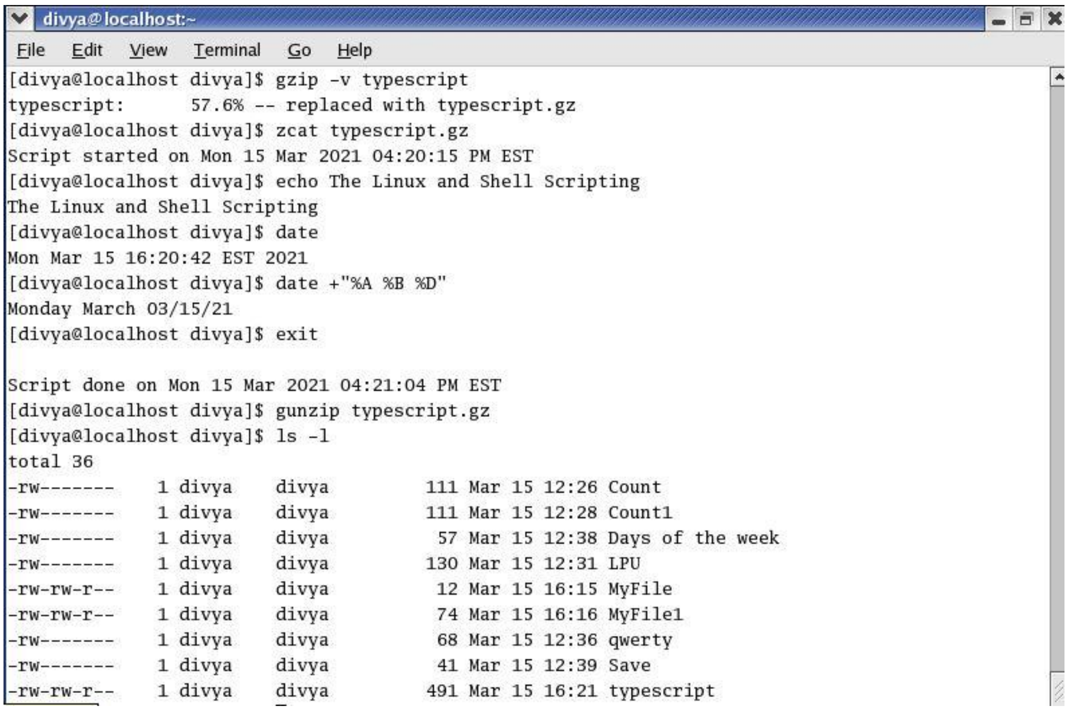
```
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya         111 Mar 15 12:26 Count
-rw-------    1 divya    divya         111 Mar 15 12:28 Count1
-rw-------    1 divya    divya          57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya         130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya          12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya          74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya          68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya          41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya         264 Mar 15 16:21 typescript.bz2
[divya@localhost divya]$ bunzip2 typescript.bz2
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya         111 Mar 15 12:26 Count
-rw-------    1 divya    divya         111 Mar 15 12:28 Count1
-rw-------    1 divya    divya          57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya         130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya          12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya          74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya          68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya          41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya         491 Mar 15 16:21 typescript
[divya@localhost divya]$
```

gzip: Compresses a File

The gzip (GNU zip) utility is older and less efficient than bzip2. Its flags and operation are very similar to those of bzip2. A file compressed by gzip is marked by a .gz filename extension. Linux stores manual pages in gzip format to save disk space; likewise, files you download from the Internet are frequently in gzip format. The gunzip utility to restore a file that has been compressed with gzip. zcat decompresses the compressed data and displays the decompressed data.



```
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya         111 Mar 15 12:26 Count
-rw-------    1 divya    divya         111 Mar 15 12:28 Count1
-rw-------    1 divya    divya          57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya         130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya          12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya          74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya          68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya          41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya         491 Mar 15 16:21 typescript
[divya@localhost divya]$ gzip -v typescript
typescript:      57.6% -- replaced with typescript.gz
[divya@localhost divya]$ zcat typescript.gz
Script started on Mon 15 Mar 2021 04:20:15 PM EST
[divya@localhost divya]$ echo The Linux and Shell Scripting
The Linux and Shell Scripting
[divya@localhost divya]$ date
Mon Mar 15 16:20:42 EST 2021
[divya@localhost divya]$ date +"%A %B %D"
Monday March 03/15/21
[divya@localhost divya]$ exit

Script done on Mon 15 Mar 2021 04:21:04 PM EST
[divya@localhost divya]$
```

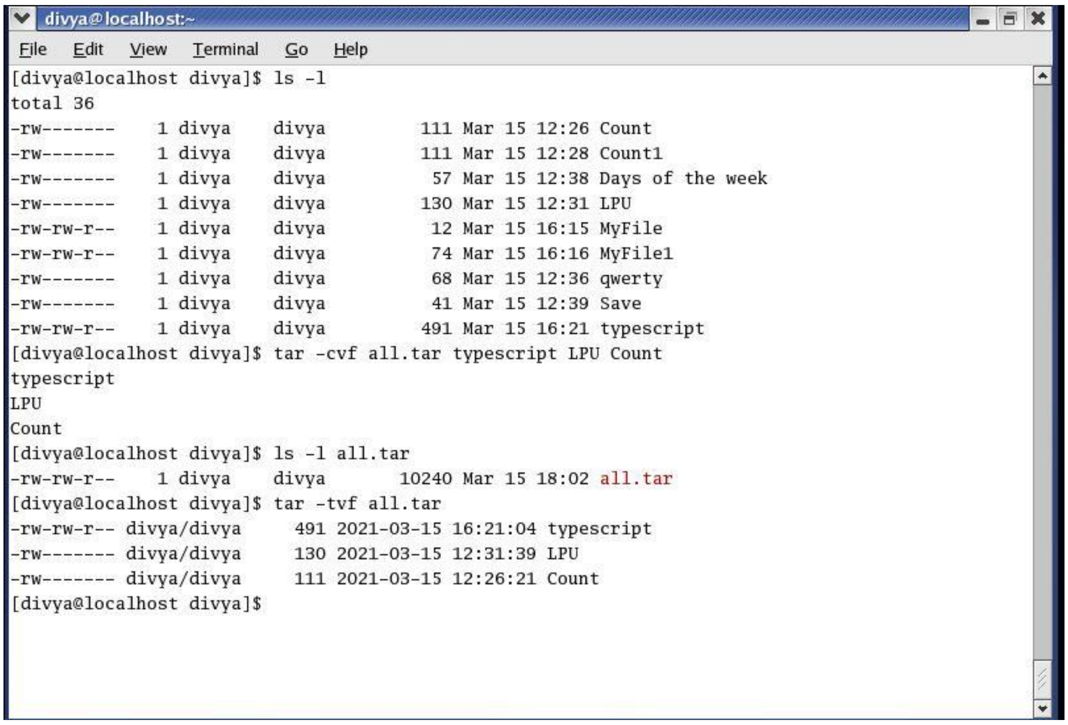


```
[divya@localhost divya]$ gzip -v typescript
typescript:      57.6% -- replaced with typescript.gz
[divya@localhost divya]$ zcat typescript.gz
Script started on Mon 15 Mar 2021 04:20:15 PM EST
[divya@localhost divya]$ echo The Linux and Shell Scripting
The Linux and Shell Scripting
[divya@localhost divya]$ date
Mon Mar 15 16:20:42 EST 2021
[divya@localhost divya]$ date +"%A %B %D"
Monday March 03/15/21
[divya@localhost divya]$ exit

Script done on Mon 15 Mar 2021 04:21:04 PM EST
[divya@localhost divya]$ gunzip typescript.gz
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya        111 Mar 15 12:26 Count
-rw-------    1 divya    divya        111 Mar 15 12:28 Count1
-rw-------    1 divya    divya         57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya        130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya         12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya         74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya         68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya         41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya        491 Mar 15 16:21 typescript
```

Do not confuse gzip and gunzip with the zip and unzip utilities. These last two are used to pack and unpack zip archives containing several files compressed into a single file that has been imported from or is being exported to a system running Windows.

tar: Packs and Unpacks Archives

The tar utility performs many functions. Its name is short for tape archive, as its original function was to create and read archive and backup tapes. Today it is used to create a single file (called a tar file, archive, or tarball) from multiple files or directory hierarchies and to extract files from a tar file.



```
[divya@localhost divya]$ ls -l
total 36
-rw-------    1 divya    divya        111 Mar 15 12:26 Count
-rw-------    1 divya    divya        111 Mar 15 12:28 Count1
-rw-------    1 divya    divya         57 Mar 15 12:38 Days of the week
-rw-------    1 divya    divya        130 Mar 15 12:31 LPU
-rw-rw-r--    1 divya    divya         12 Mar 15 16:15 MyFile
-rw-rw-r--    1 divya    divya         74 Mar 15 16:16 MyFile1
-rw-------    1 divya    divya         68 Mar 15 12:36 qwerty
-rw-------    1 divya    divya         41 Mar 15 12:39 Save
-rw-rw-r--    1 divya    divya        491 Mar 15 16:21 typescript
[divya@localhost divya]$ tar -cvf all.tar typescript LPU Count
typescript
LPU
Count
[divya@localhost divya]$ ls -l all.tar
-rw-rw-r--    1 divya    divya      10240 Mar 15 18:02 all.tar
[divya@localhost divya]$ tar -tvf all.tar
-rw-rw-r-- divya/divya     491 2021-03-15 16:21:04 typescript
-rw------- divya/divya     130 2021-03-15 12:31:39 LPU
-rw------- divya/divya     111 2021-03-15 12:26:21 Count
[divya@localhost divya]$
```

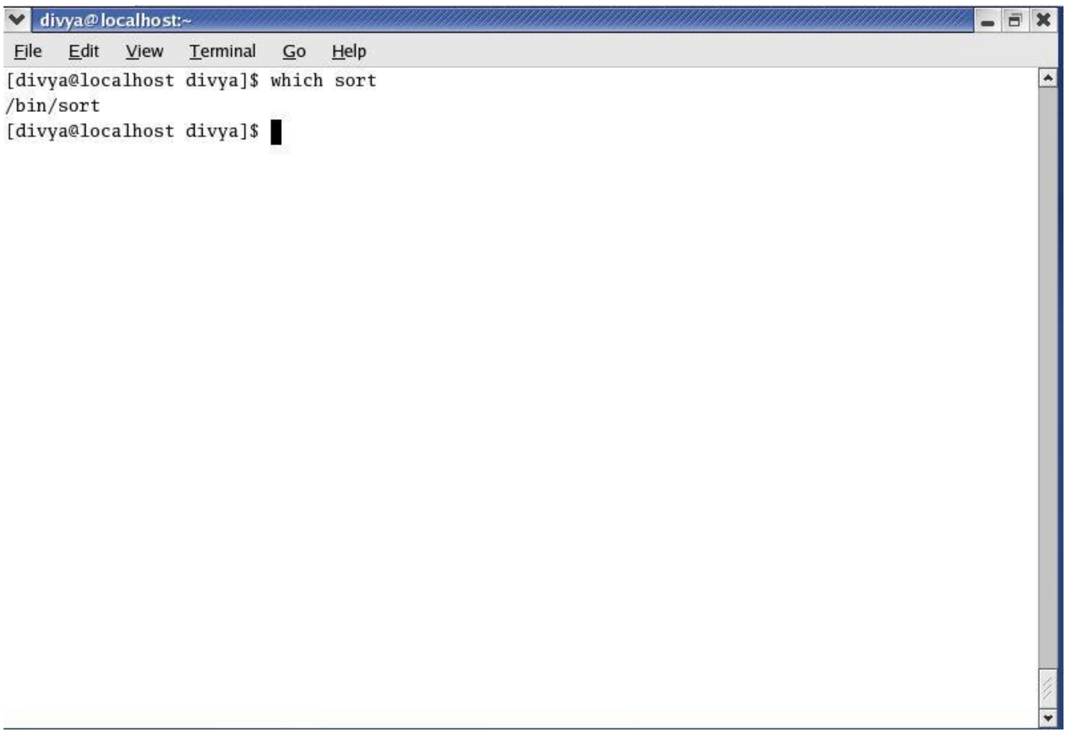
Tar uses the –c (create), –v (verbose), and –f (write to or read from a file) options to create an archive named all.tar from these files. Each line of output displays the name of the file tar is appending to the archive it is creating. The tar utility adds overhead when it creates an archive. The final command uses the –t option to display a table of contents for the archive. You can use bzip2, compress, or gzip to compress tar files, making them easier to store and handle. Many files you download from the Internet will already be in one of these formats. Files that have been processed by tar and compressed by bzip2 frequently have a filename extension of .tar.bz2 or .tbz. Those processed by tar and gzip have an extension of .tar.gz, .tgz, or .gz, whereas files processed by tar and compress use .tar.Z as the extension.

5.5 Locating Commands

The where is and locate utilities can help you find a command whose name you have forgotten or whose location you do not know. When multiple copies of a utility or program are present, which tells you which copy you will run. The slocate utility searches for files on the local system.

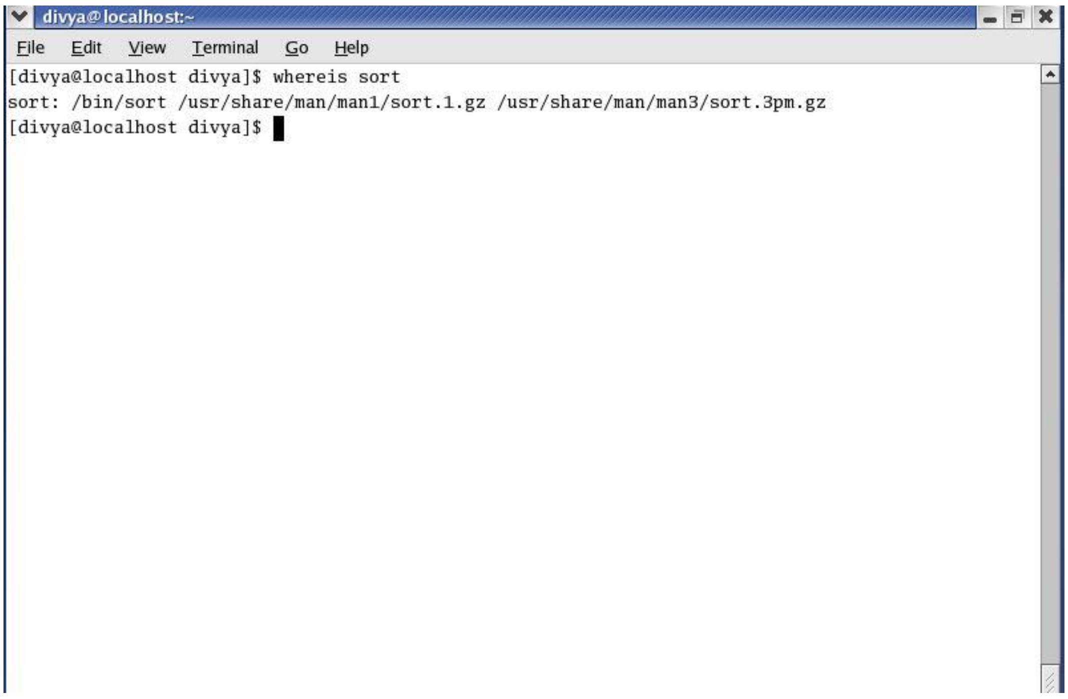
Which and where is: Locate a Utility

When you give Linux a command, the shell searches a list of directories for a program with that name and runs the first one it finds. This list of directories is called a search path. If you do not change the search path, the shell searches only a standard set of directories and then stops searching. However, other directories on the system may also contain useful utilities. The which utility locates utilities by displaying the full pathname of the file for the utility. The local system may include several utilities that have the same name. When you type the name of a utility, the shell searches for the utility in your search path and runs the first one it finds. You can find out which copy of the utility the shell will run by using which.



```
[divya@localhost divya]$ which sort
/bin/sort
[divya@localhost divya]$ 
```

The where is utility searches for files related to a utility by looking in standard locations instead of using your search path.

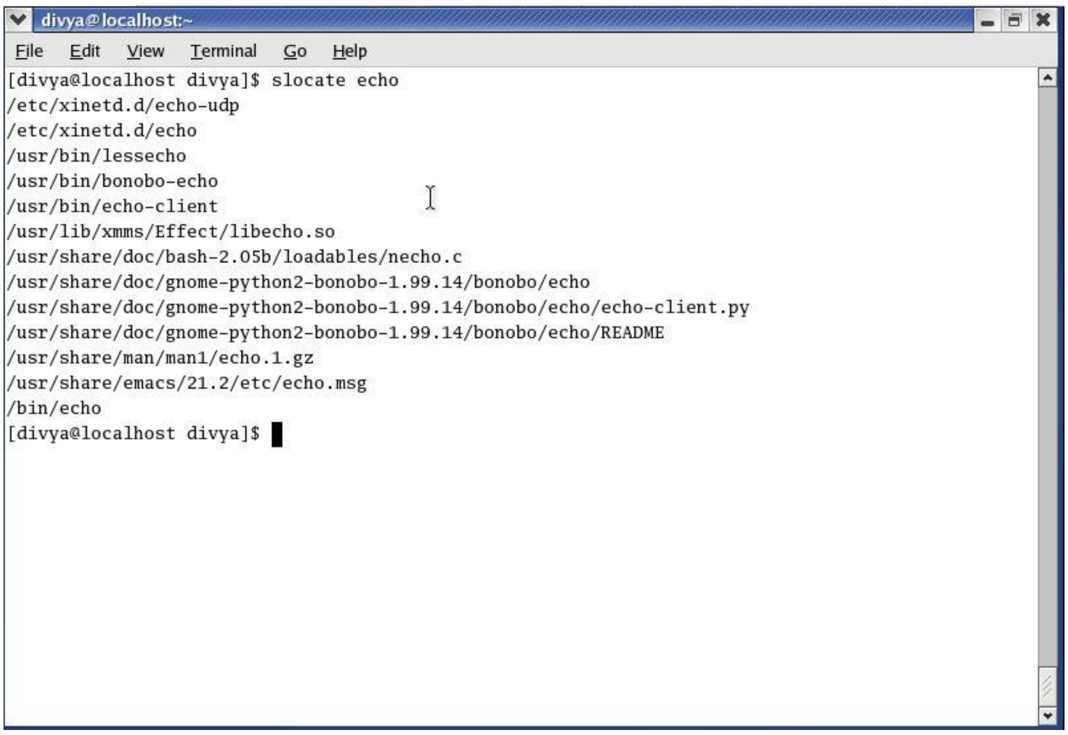


```
[divya@localhost divya]$ whereis sort
sort: /bin/sort /usr/share/man/man1/sort.1.gz /usr/share/man/man3/sort.3pm.gz
[divya@localhost divya]$
```

Where is finds three references to sort: the sort utility file, a sort header file, and the sort man page.

slocate/locate: Searches for a File

The slocate (secure locate) or locate utility searches for files on the local system.



```
[divya@localhost divya]$ slocate echo
/etc/xinetd.d/echo-udp
/etc/xinetd.d/echo
/usr/bin/lessecho
/usr/bin/bonobo-echo
/usr/bin/echo-client
/usr/lib/xmms/Effect/libecho.so
/usr/share/doc/bash-2.05b/loadables/necho.c
/usr/share/doc/gnome-python2-bonobo-1.99.14/bonobo/echo
/usr/share/doc/gnome-python2-bonobo-1.99.14/bonobo/echo/echo-client.py
/usr/share/doc/gnome-python2-bonobo-1.99.14/bonobo/echo/README
/usr/share/man/man1/echo.1.gz
/usr/share/emacs/21.2/etc/echo.msg
/bin/echo
[divya@localhost divya]$
```

Summary:

- When you log in the system, you work in the home directory.

- When you want to view a file that is longer than one screen, you can use either the less utility or the more utility.
- If the destination-file exists before you give a cp command, cp overwrites it. The cp –i (interactive) option prompts you before it overwrites a file.
- If a file contains a list of names and has two successive entries for the same person, uniq skips the duplicate line.
- The diff utility with the –u (unified output format) option first displays two lines indicating which of the files you are comparing will be denoted by a plus sign (+) and which by a minus sign (–).
- You can use the file utility to learn about the contents of a file without having to open and examine the file yourself.
- By default script captures the session in a file named typescript. To specify a different filename, follow the script command with a SPACE and the filename
- The todos (to DOS; part of the tofrodos package) or unix2dos (UNIX to DOS; part of the unix2dos package) utility converts a Linux text file so it can be read on a Windows system.
- Linux stores manual pages in gzip format to save disk space; likewise, files you download from the Internet are frequently in gzip format.
- The gunzip utility to restore a file that has been compressed with gzip.
- The whereis and slocate utilities can help you find a command whose name you have forgotten or whose location you do not know.

Keywords

- **Directory:** A directory is a resource that can hold files. On other operating systems, like Windows, a directory is referred to as a folder.
- **ls:** The ls utility lists the names of files which are available.
- **cat:** The cat utility displays the contents of a text file.
- **rm:** The rm (remove) utility deletes a file.
- **cp:**: The cp (copy) utility makes a copy of a file. This utility can copy any file, including text and executable program (binary) files.
- **mv**: The mv (move) utility can rename a file without making a copy of it.
- **grep:** The grep utility searches through one or more files to see whether any contain a specified string of characters.
- **sort**: The sort utility displays the contents of a file in order by lines; it does not change the original file.
- **diff:** The diff (difference) utility compares two files and displays a list of the differences between them.
- **uniq**: The uniq (unique) utility displays a file, skipping adjacent duplicate lines, but does not change the original file.
- **Pipe:** A pipe (written as a vertical bar [|] on the command line and appearing as a solid or broken vertical line on a keyboard) provides the simplest form of this kind of communication.
- **echo:** The echo utility copies the characters you type on the command line after echo to the screen.
- **date**: The date utility displays the current date and time.

- **script:** The script utility records all or part of a login session, including your input and the system's responses.
- **bzip2:** The bzip2 utility compresses a file by analyzing it and recoding it more efficiently.
- **bzcat:** The bzcat utility displays a file that has been compressed with bzip2.
- **bunzip2**: You can use the bunzip2 utility to restore a file that has been compressed with bzip2.
- **whereis**: The whereis utility searches for files related to a utility by looking in standard locations instead of using your search path.
- **slocate:** The slocate (secure locate) or locate utility searches for files on the local system.

Self Assessment

1. Which of these utilities is used to convert a Linux text file so that it can be read on a Windows system?

A. todos
B. dos2unix
C. echo
D. script

2. Which of these utilities records a shell session?

A. echo
B. date
C. script
D. cat

3. Which of these is not used in Linux System?

A. zip
B. bzip2
C. bunzip2
D. zcat

4. Which of these utilities is used to compress a file?

A. bzip2
B. bunzip2
C. gunzip
D. zcat

5. Which of these utilities records a shell session?

A. echo
B. date
C. script
D. cat

6. In less/more, which of these keys should be pressed to display the next screen?

A. SPACE
B. ENTER
C. CTRL
D. ALT

7. Which of these utilities removes duplicate lines from a file?
A. grep
B. uniq
C. sort
D. None of the above

8. Which option with rm provides the interactive deletion?
A. -i
B. -a
C. -b
D. -r

9. Which of these utilities are used when you want to view a file that is longer than one page?
A. more
B. less
C. Both more and less
D. None of the above

10. Which of these utility displays the contents of a text file?
A. ls
B. cat
C. rm
D. All of the above mentioned

11. Which of these utilities displays the names of files that are available?
A. ls
B. cat
C. rm
D. All of the above mentioned

12. Which of these utilities compares two files and display the difference between them?
A. grep
B. uniq
C. diff
D. differ

13. Which symbol is used for representation of a pipe?

A. #
B. $
C. |
D. &

14. Which of these represents the basic utilities in Linux?
A. ls
B. cat
C. rm
D. All of the above mentioned

15. When you log in a Linux system, you work in _________ directory
A. root
B. home
C. var
D. None of the above

Answers for Self Assessment

1.	A	2.	C	3.	A	4.	A	5.	C
6.	A	7.	C	8.	A	9.	C	10.	B
11.	A	12.	C	13.	C	14.	D	15.	B

Review Questions:

1. What is command line utilities? Give some examples and explain the usage of few basic utilities.
2. Which basic utility is used to delete a file? How can we make it interactive?
3. What are pager utilities? How these are useful?
4. Which utilities are used to work with files? Explain any five utilities in detail.
5. Explain the use and syntax of echo, date and script utilities.
6. What is compressing and archiving of files? Explain the utilities used for it.
7. Explain what are locating commands?

Further Readings

Mark G. Sobell, A Practical Guide to Fedora and Red Hat Enterprise Linux, Fifth Edition, Pearson Education

Web Links

https://www.javatpoint.com/linux-commands

Unit 06: File Systems

CONTENTS

Objectives

After studying this unit, you will be able to:

- Obtain user and system information
- Communicate with other users
- Understand the filesystem
- Understand the pathnames
- Work with directories
- Understand the access permission
- Understand the Access Control Lists and Links

Introduction

There are some utilities that provide information about who is using the system, what those users are doing, and how the system is running. To find out who is using the local system, you can employ one of several utilities that vary in the details they provide and the options they support.

6.1 Obtaining User and System Information

- who
- finger
- w

The oldest utility, who, produces a list of users who are logged in on the local system, the device each person is using, and the time each person logged in. The w and finger utilities show more detail, such as each user's full name and the command line each user is running. You can use the finger utility to retrieve information about users on remote systems if the local system is attached to a network.

who: Lists Users on the System

The who utility displays a list of users who are logged in on the local system.

```
$ who
sam       tty4         2009-07-25 17:18
max       tty2         2009-07-25 16:42
zach      tty1         2009-07-25 16:39
max       pts/4        2009-07-25 17:27 (coffee)
```

The information that who displays is useful when you want to communicate with a user on the local system.If you need to find out which terminal you are using or what time you logged in, you can use the command who am i:

```
$ who am i
max      tty2         2009-07-25 16:42
```

finger: Lists Users on the System

You can use finger to display a list of users who are logged in on the local system. In addition to usernames, finger supplies each user's full name along with information about which device the user's terminal is connected to, how recently the user typed something on the keyboard, when the user logged in, and available contact information. If the user has logged in over the network, the name of the remote system is shown as the user's location.

```
$ finger
Login     Name           Tty      Idle  Login Time   Office ...
max       Max Wild      *tty2           Jul 25 16:42
max       Max Wild       pts/4     3    Jul 25 17:27 (coffee)
sam       Sam the Great *tty4     29    Jul 25 17:18
zach      Zach Brill    *tty1   1:07    Jul 25 16:39
```

On systems where security is a concern, the system administrator may disable finger. You can also use finger to learn more about an individual by specifying a username on the command line.

```
$ finger max
Login: max                              Name: Max Wild
Directory: /home/max                    Shell: /bin/tcsh
On since Fri Jul 25 16:42 (PDT) on tty2 (messages off)
On since Fri Jul 25 17:27 (PDT) on pts/4 from coffee
   3 minutes 7 seconds idle
New mail received Sat Jul 25 17:16 2009 (PDT)
     Unread since Sat Jul 25 16:44 2009 (PDT)
Plan:
I will be at a conference in Hawaii all next week.
If you need to see me, contact Zach Brill, x1693.
```

You can also use finger to display a user's username. The finger utility, which is not case sensitive, can search for information on Helen using her first or last name.

```
$ finger HELEN
Login: hls                              Name: Helen Simpson.
...
$ finger simpson
Login: hls                              Name: Helen Simpson.
...
```

w: Lists Users on the System

The w utility displays a list of the users who are logged in on the local system. The information that w displays is useful when you want to communicate with someone at your installation.

```
$ w
 17:47:35 up 1 day,  8:10,  4 users,  load average: 0.34, 0.23, 0.26
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
sam      tty4     -               17:18   29:14m  0.20s  0.00s vi memo
max      tty2     -               16:42    0.00s  0.20s  0.07s w
zach     tty1     -               16:39    1:07   0.05s  0.00s run_bdgt
max      pts/4    coffee          17:27    3:10m  0.24s  0.24s -bash
```

The first line the w utility displays includes the time of day, the period of time the computer has been running (in days, hours, and minutes), the number of users logged in, and the load average (how busy the system is).

6.2 Communicating with Other Users

write: Sends a Message

These are used to exchange messages and files with other users either interactively or through email. It is enabled as a two-way communication.

The syntax of a write command line is: write username [terminal]. The username is the username of the user you want to communicate with. The terminal is an optional device name that is useful if the user is logged in more than once. You can display the usernames and device names of all users who are logged in on the local system by using who, w, or finger. To establish two-way communication with another user, you and the other user must each execute write, specifying the other's username as the username. The write utility then copies text, line by line, from one keyboard/display to the other. Sometimes it helps to establish a convention, such as typing o (for "over") when you are ready for the other person to type and typing oo (for "over and out") when you are ready to end the conversation. When you want to stop communicating with the other user, press CONTROL-D at the beginning of a line. Pressing CONTROL-D tells write to quit, displays EOF (end of file) on the other user's terminal, and returns you to the shell. The other user must do the same.

If the Message from banner appears on your screen and obscures something you are working on, press CONTROL- L or CONTROL- R to refresh the screen and remove the banner. Then you can clean up, exit from your work, and respond to the person who is writing to you. You have to remember who is writing to you, however, because the banner will no longer appear on the screen.

mesg: Denies or Accepts Messages

By default, messages to your screen are blocked. Give the following mesg command to allow other users to send you messages: $ mesg y

If Max had not given this command before Zach tried to send him a message, Zach might have seen the following message: $ **write max**

```
write: max has messages disabled
```

You can block messages by entering mesg n. Give the command mesg by itself to display is y (for "yes, messages are allowed") or is n (for "no, messages are not allowed").If you have messages blocked and you write to another user, write displays the following message because, even if you are allowed to write to another user, the user will not be able to respond to you:

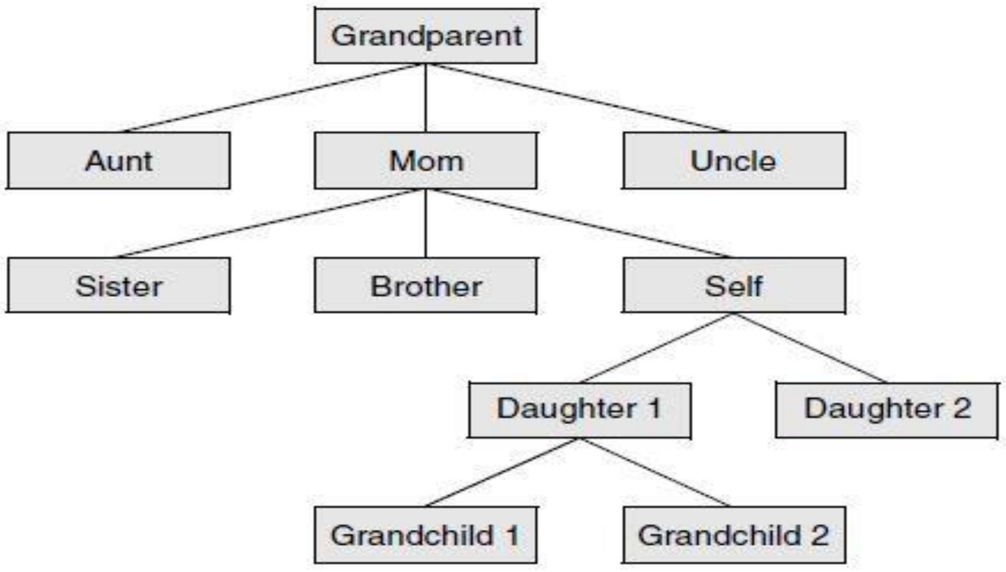
```
$ write max

write: write: you have write permission turned off.
```

Email

Email enables you to communicate with users on the local system and, if the installation is part of a network, with other users on the network. If you are connected to the Internet, you can communicate electronically with users around the world. Email utilities differ from write in that email utilities can send a message when the recipient is not logged in. In this case the email is stored until the recipient reads it. These utilities can also send the same message to more than one user at a time. Many email programs are available for Linux, including the original character-based mailx program, Mozilla/Thunderbird, pine, mail through emacs, KMail, and evolution. Another popular graphical email program is sylpheed. Two programs are available that can make any email program easier to use and more secure. The procmail program creates and maintains email servers and mailing lists; preprocesses email by sorting it into appropriate files and directories. The GNU Privacy Guard encrypts and decrypts email and makes it almost impossible for an unauthorized person to read.

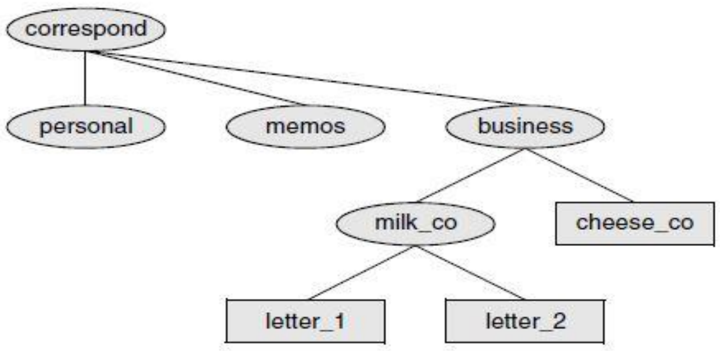
6.3 The Filesystem

A filesystem is a set of data structures that usually resides on part of a disk and that holds directories of files. Filesystems store user and system data that are the basis of users' work on the system and the system's existence. A hierarchical structure frequently takes the shape of a pyramid. One example: a family's lineage. A couple has a child, who may in turn have several children, each of whom may have more children. This hierarchical structure is called a family tree.



Like the family tree it resembles, the Linux filesystem is called a tree. It consists of a set of connected files. This structure allows you to organize files so you can easily find any one. On a standard Linux system, each user starts with one directory, to which the user can add subdirectories to any desired level. By creating multiple levels of subdirectories, a user can expand the structure as needed. Typically each subdirectory is dedicated to a single subject, such as a person, project, or event. The subject dictates whether a subdirectory should be subdivided further. For example, a secretary's subdirectory named correspond.

Example: A secretary's directories



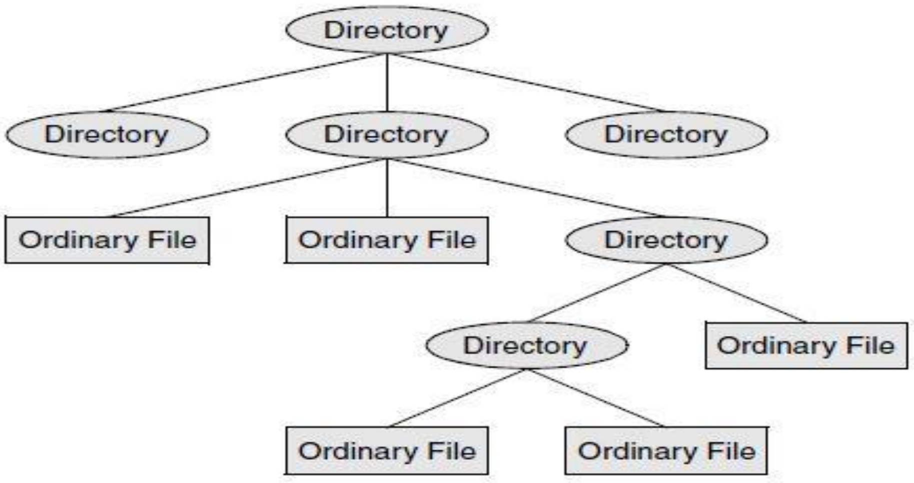
This directory contains three subdirectories: business, memos, and personal. The business directory contains files that store each letter the secretary types.

Strength of Linux File System

One major strength of the Linux filesystem is its ability to adapt to users' needs. You can take advantage of this strength by strategically organizing your files so they are most convenient and useful for you.

Directory Files and Ordinary Files

Like a family tree, the tree representing the filesystem is usually pictured upside down, with its root at the top. The tree "grows" downward from the root, with paths connecting the root to each of the other files. At the end of each path is either an ordinary file or a directory file. Ordinary files, or simply files, appear at the ends of paths that cannot support other paths. Directory files, also referred to as directories or folders, are the points that other paths can branch off from. When you refer to the tree, up is toward the root and down is away from the root. Directories directly connected by a path are called parents (closer to the root) and children (farther from the root).A pathname is a series of names that trace a path along branches from one file to another.



Filenames

Every file has a filename. The maximum length of a filename varies with the type of filesystem; Linux supports several types of filesystems. Although most of today's filesystems allow files with names up to 255 characters long, some filesystems restrict filenames to fewer characters. Choose characters from the following list:

- Uppercase letters (A–Z)
- Lowercase letters (a–z)
- Numbers (0–9)
- Underscore (_)
- Period (.)
- Comma (,)

Like the children of one parent, no two files in the same directory can have the same name. Files in different directories, like the children of different parents, can have the same name. The filenames you choose should mean something. Too often a directory is filled with important files with such unhelpful names as hold1, wombat, and junk, not to mention foo and foobar. Such names are poor choices because they do not help you recall what you stored in a file. The following filenames conform to the suggested syntax and convey information about the contents of the file:

- Correspond
- January
- Davis
- Reports
- 2001
- acct_payable

If you keep the filenames short, they are easy to type. The disadvantage of short filenames is that they are typically less descriptive than long filenames. Long filenames enable you to assign descriptive names to files. To help you select among files without typing entire filenames, shells support filename completion. You can use uppercase and/or lowercase letters within filenames, but be careful: Many filesystems are case sensitive. For example, the popular ext family of filesystems and the UFS filesystem are case sensitive, so files named JANUARY, January, and January refer to three distinct files. The FAT family of filesystems (used mostly for removable media) is not case sensitive, so those three filenames represent the same file. The HFS+ filesystem, which is the default OS X filesystem, is case preserving but not case sensitive. Although you can use SPACEs within filenames, it is a poor idea. Because a SPACE is a special character, you must quote it on a command line. Use periods or underscores instead of SPACEs: joe.05.04.26, new_stuff.

```
$ lpr my\ file
$ lpr "my file"
```

Filename Extensions

A filename extension is the part of the filename following an embedded period. The filename extensions help describe the contents of the file.

Filename with Extension	Meaning of Extension
compute.c	A C programming language source file
compute.o	The object code file of compute.c
compute	The executable file of compute.c
memo-0410.txt	A text file
memo.pdf	A PDF file, view with xpdf or kpdf
memo.ps	A PostScript file, view with gs or kpdf
memo.z	A file compressed wih compress; use uncompress or gunzip to decompress
Memo.tgz or memo.tar.gz	A tar archive of files compressed with gzip
Memo.gz	A file compressed with gzip, view with zcat or decompress with gunzip
Memo.bz2	A file compressed with bzip2; view with bzcat or decompress with bunzip2
Memo.html	A file meant to be viewed using a Web browser, such as firefox

.gif, .jpg, .jpeg, .bmp, .tif or .tiff	A file containing graphical information, such as picture

Hidden Filenames

A filename that begins with a period is called a hidden filename because ls does not normally display it. The command ls –a displays all filenames, even hidden ones. Names of startup files usually begin with a period so that they are hidden and do not clutter a directory listing.The .plan file is also hidden. Two special hidden entries—a single and double period (. and ..)—appear in every directory.

The Working Directory

While you are logged in on a Linux system, you are always associated with a directory. The directory you are associated with is called the working directory or current directory. Sometimes this association is referred to in a physical sense: "You are in (or working in) the zach directory." The pwd (print working directory) shell builtin displays the pathname of the working directory.

Your Home Directory

When you first log in, the working directory is your home directory. To display the pathname of your home directory, use pwd just after you log in. Linux home directories are typically located in /home. When used without any arguments, the ls utility displays a list of the files in the working directory. All the files you have created up to this point were created in your home directory.

Startup Files

Startup files, which appear in your home directory, give the shell and other programs information about you and your preferences. Because the startup files have hidden filenames, you must use the ls –a command to see whether one is in your home directory.

6.4 Pathnames

Every file has a pathname, which is a trail from a directory through part of the directory hierarchy to an ordinary file or a directory. Within a pathname, a slash (/) to the right of a filename indicates that the file is a directory file. The file following the slash can be an ordinary file or a directory file.

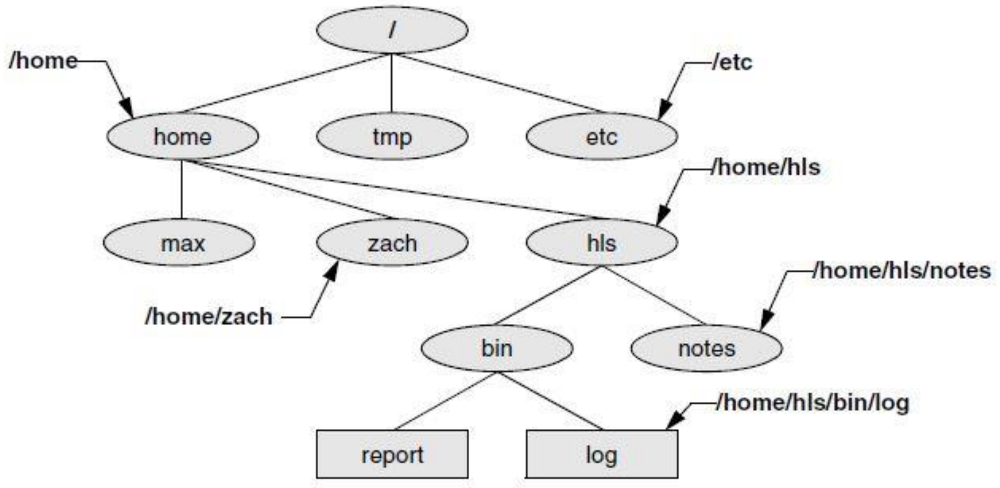
- Absolute Pathnames
- Relative Pathnames

Absolute Pathnames

The root directory of the filesystem hierarchy does not have a name. It is referred to as the root directory. It is represented by a / (slash) standing alone or at the left end of a pathname. An absolute pathname starts with a slash (/), which represents the root directory. The slash is followed by the name of a file located in the root directory. An absolute pathname continues, tracing a path through all intermediate directories, to the file identified by the pathname. String all the filenames in the path together, following each directory with a slash (/). This string of filenames is called an absolute pathname because it locates a file absolutely by tracing a path from the root directory to the file. Using an absolute pathname, you can list or otherwise work with any file on the local system, regardless of the working directory at the time you give the command. For example, Sam can give the following command while working in his home directory to list the files in the /usr/bin directory:

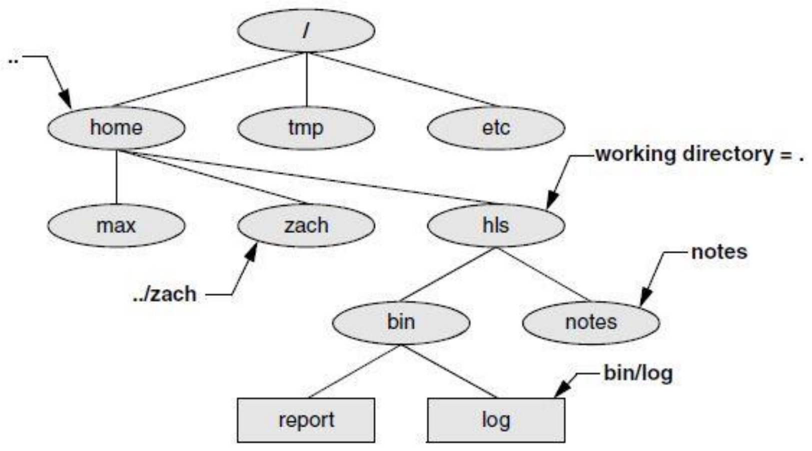
```
$ pwd
/home/sam
```

```
$ ls /usr/bin
7z kwin
7za kwin_killer_helper
822-date kwin_rules_dialog
```



Relative Pathnames

A relative pathname traces a path from the working directory to a file. The pathname is relative to the working directory. Any pathname that does not begin with the root directory (represented by /) or a tilde (~) is a relative pathname. Like absolute pathnames, relative pathnames can trace a path through many directories. The simplest relative pathname is a simple filename, which identifies a file in the working directory.



Significance of the Working Directory

To access any file in the working directory, you need only a simple filename. To access a file in another directory, you must use a pathname. Typing a long pathname is tedious and increases the chance of making a mistake. This possibility is less likely under a GUI, where you click filenames or icons. You can choose a working directory for any task to reduce the need for long pathnames.

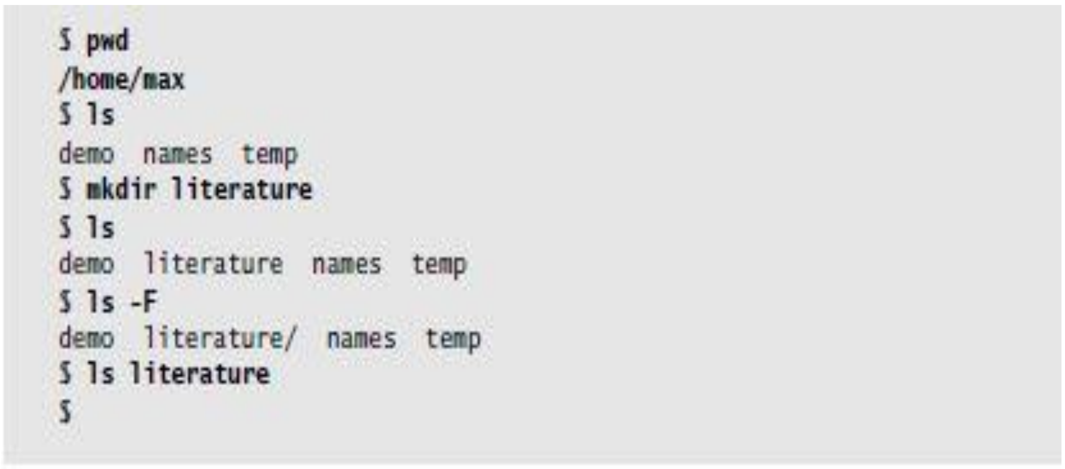
6.5 Working with Directories

- how to create directories (mkdir),

- switch between directories (cd),
- remove directories (rmdir)
- mv, cp: Move or Copy Files
- mv: Moves a Directory

mkdir: Creates a Directory

The mkdir utility creates a directory. The argument to mkdir becomes the pathname of the new directory.



```
$ pwd
/home/max
$ ls
demo  names  temp
$ mkdir literature
$ ls
demo  literature  names  temp
$ ls -F
demo  literature/  names  temp
$ ls literature
$
```

He uses a relative pathname (a simple filename) because he wants the literature directory to be a child of the working directory. Max could have used an absolute pathname to create the same directory: mkdir /home/max/literature or mkdir ~max/literature.There are two ways to create the promo directory as a child of the newly created literature directory.

The first way checks that /home/max is the working directory and uses a relative pathname:

```
$ pwd
/home/max
$ mkdir literature/promo
```

The second way uses an absolute pathname:

```
$ mkdir /home/max/literature/promo
```

cd: Changes to Another Working Directory

The cd (change directory) utility makes another directory the working directory but does not change the contents of the working directory.

```
$ cd /home/max/literature
$ pwd
/home/max/literature
$ cd
$ pwd
/home/max
$ cd literature
$ pwd
/home/max/literature
```

The working directory versus your home directory

The working directory is not the same as your home directory. Your home directory remains the same for the duration of your session and usually from session to session. Immediately after you log in, you are always working in the same directory: your home directory. Unlike your home directory, the working directory can change as often as you like. You have no set working directory, which explains why some people refer to it as the current directory. When you log in and until you change directories using cd, your home directory is the working directory.

The . and .. Directory Entries

The mkdir utility automatically puts two entries in each directory it creates: a single period (.) and a double period (..). The . is synonymous with the pathname of the working directory and can be used in its place; the .. Is synonymous with the pathname of the parent of the working directory. These entries are hidden because their filenames begin with a period.

rmdir: Deletes a Directory

The rmdir (remove directory) utility deletes a directory. You cannot delete the working directory or a directory that contains files other than the . and .. entries. If you need to delete a directory that has files in it, first use rm to delete the files and then delete the directory. You do not have to (nor can you) delete the . and .. entries; rmdir removes them automatically. The following command deletes the promo directory: $ rmdir /home/max/literature/promo

The rm utility has a –r option (rm –r filename) that recursively deletes files, including directories, within a directory and also deletes the directory itself.

Although rm –r is a handy command, you must use it carefully. Do not use it with an ambiguous file reference such as *. It is shockingly easy to wipe out your entire home directory with a single short command.

mv, cp: Move or Copy Files

You can use this utility to move files from one directory to another (change the pathname of a file) as well as to change a simple filename. When used to move one or more files to a new directory, the mv command has this syntax: mv existing-file-list directory.

If the working directory is /home/max, Max can use the following command to move the files names and temp from the working directory to the literature directory:$ mv names temp literature.

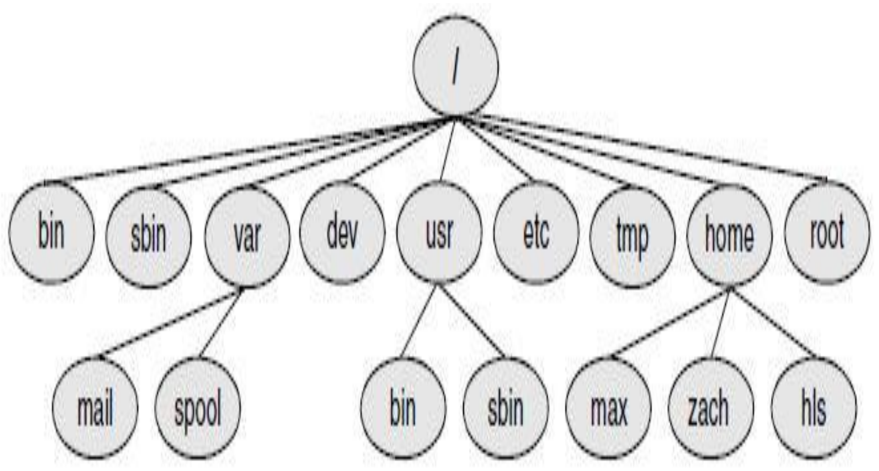
This command changes the absolute pathnames of the names and temp files from /home/max/names and /home/max/temp to /home/max/literature/names and /home/max/literature/temp, respectively. Like most Linux commands, mv accepts either absolute or relative pathnames. The cp utility works in the same way as mv does, except that it makes copies of the existing-file-list in the specified directory.

mv: Moves a Directory

Just as it moves ordinary files from one directory to another, so mv can move directories. The syntax is similar except that you specify one or more directories, not ordinary files, to move:mv existing-directory-list new-directory

If new-directory does not exist, the existing-directory-list must contain just one directory name, which mv changes to new-directory (mv renames the directory). Although you can rename directories using mv, you cannot copy their contents with cp unless you use the –r (recursive) option.

A typical FHS-based Linux filesystem structure



Important Standard Directories and Files

/	Root: The root directory, present in all Linux filesystem structures, is the ancestor of all files in the filesystem.
/bin	Essential command binaries Holds the files needed to bring the system up and run it when it first comes up in single-user or recovery mode.
/boot	Static files of the boot loader Contains all files needed to boot the system.
/dev	Device files Contains all files that represent peripheral devices, such as disk drives, terminals, and printers
/etc	Machine–local system configuration files Holds administrative, configuration, and other system files. One of the most important is /etc/passwd, which contains a list of all users who have permission to use the system.
/etc/opt	Configuration files for add-on software packages kept in /opt
/etc/X11	Machine–local configuration files for the X Window System

/home	User home directories
/lib	Shared libraries
/lib/modules	Loadable kernel modules
/mnt	Mount point for temporarily mounting filesystems
/opt	Add-on (optional) software packages
/proc	Kernel and process information virtual filesystem
/root	Home directory for the root account
/sbin	Essential system binaries Utilities used for system administration are stored in /sbin and /usr/sbin. The /sbin directory includes utilities needed during the booting process, and /usr/sbin holds utilities used after the system is up and running.
/sys	Device pseudofilesystem
/tmp	Temporary Files
/usr/games	Games and educational programs
/usr/include	Header files used by C programs
/usr/lib	Libraries
/usr/local	Local hierarchy Holds locally important files and directories that are added to the system. Subdirectories can include bin, games, include, lib, sbin, share, and src.
/usr/sbin	Nonvital system administration binaries See /sbin.
/usr/share	Architecture-independent data Subdirectories can include dict, doc, games, info, locale, man, misc, terminfo, and zoneinfo.
/usr/share/doc	Documentation
/usr/share/info	GNU info system's primary directory

/usr/share/man	**Online manuals**
/usr/src	**Source code**
/var	**Variable data Files with contents that vary as the system runs are kept in subdirectories** under **/var. The most common examples are temporary files, system log** files, spooled files, and user mailbox files.
/var/log	**Log files Contains lastlog (a record of the last login by each user), messages (system** messages from **syslogd), and wtmp (a record of all logins/logouts), among other log** files.
/var/spool	**Spooled application data Contains anacron, at, cron, lpd, mail, mqueue, samba,** and other directories.

6.6 Linux Access Permissions

Linux supports two methods of controlling who can access a file and how they can access it:

1) Traditional Linux access permissions
2) Access Control Lists (ACLs).

Three types of users can access a file:

1) The owner of the file (owner),
2) A member of a group that the file is associated with (group),
3) Everyone else (other).

Ways to Access an Ordinary File:

- A user can attempt to access an ordinary file in three ways:

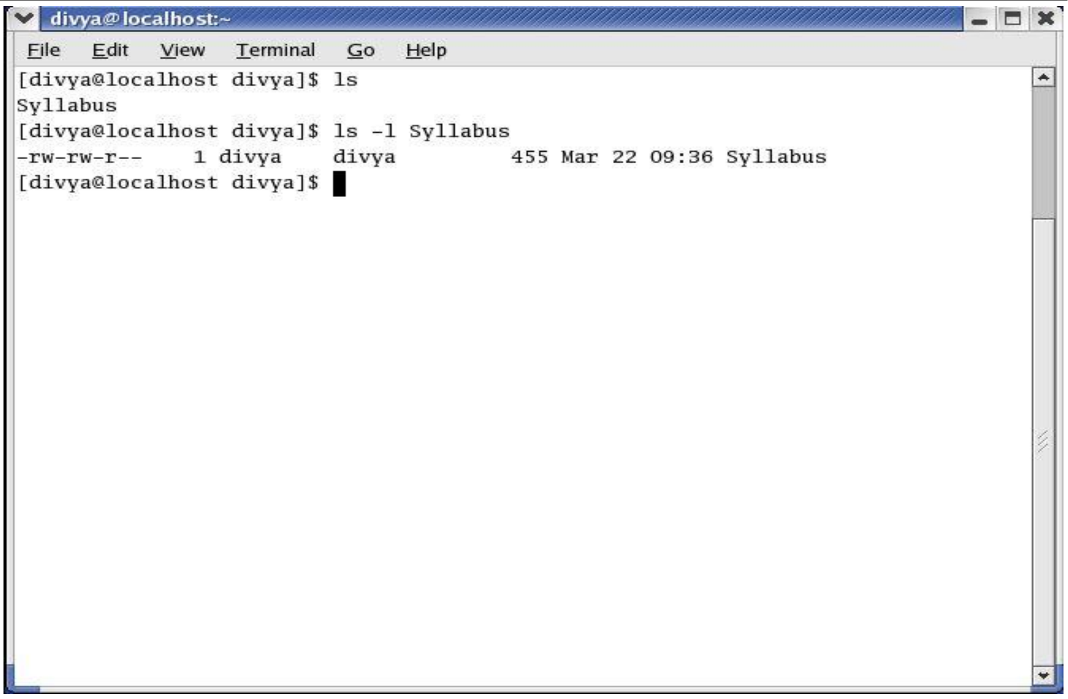
1) Read from
2) Write to
3) Execute it.

Access Permissions

- ls –l: Displays Permissions
- chmod: Changes Access Permissions
- Setuid and Setgid Permissions
- Directory Access Permissions

ls –l: Displays Permissions:

When you call ls with the –l option and the name of one or more ordinary files, ls displays a line of information about the file.



```
[divya@localhost divya]$ ls
Syllabus
[divya@localhost divya]$ ls -l Syllabus
-rw-rw-r--    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$
```

From left to right, the lines that an ls –l command displays contain: The type of file (first character), the file's access permissions (the next nine characters), the ACL flag (present if the file has an ACL), the number of links to the file, the name of the owner of the file (usually the person who created the file), the name of the group the file is associated with, the size of the file in characters (bytes), the date and time the file was created or last modified, the name of the file and the type of file is a hyphen (–) because it is an ordinary file.

Character	Meaning
-	Ordinary
b	Block Device
c	Character Device
d	Directory
p	FIFO (names Pipe)
l	Symbolic link

The next three characters specify the access permissions for the owner of the file.

- R indicates read permission,
- w indicates write permission,

- x indicates execute permission.

A – in one of the positions indicates that the owner does not have the permission associated with that position. In a similar manner the next three characters represent permissions for the group. The final three characters represent permissions for other (everyone else).Although execute permission can be allowed for any file, it does not make sense to assign execute permission to a file that contains a document, such as a letter.

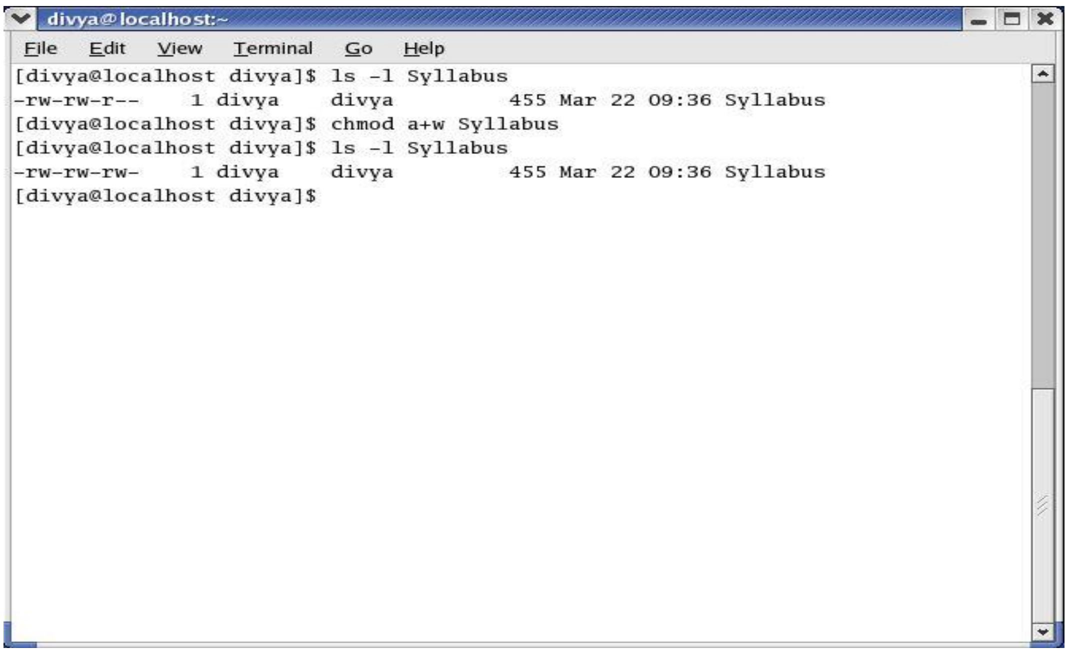
chmod: Changes Access Permissions

The Linux file access permission scheme lets you give other users access to the files you want to share yet keep your private files confidential.

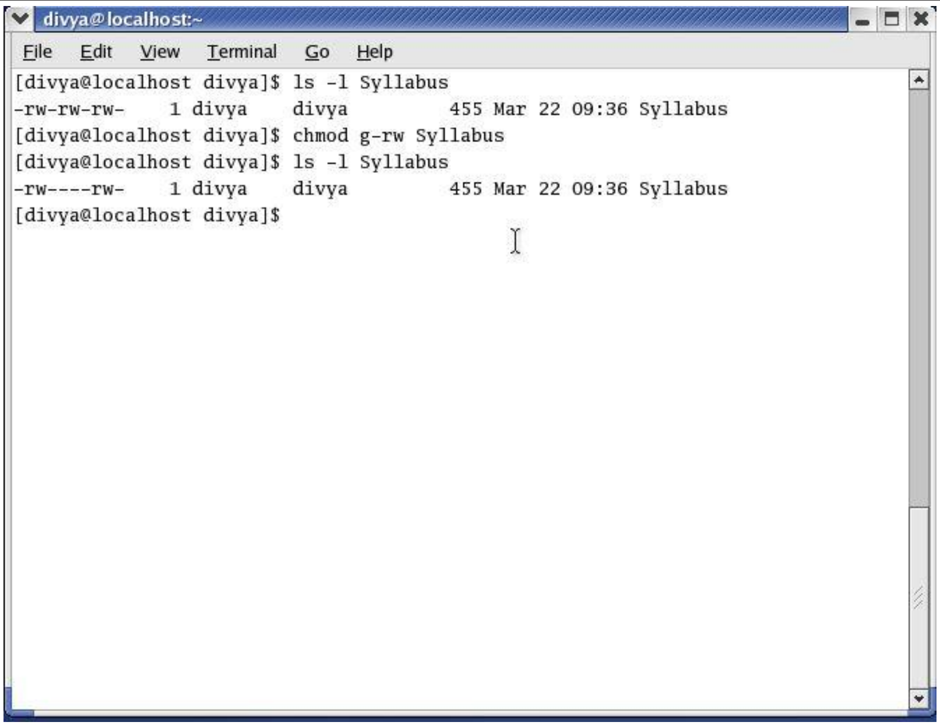
- You can allow other users to read from and write to a file.
- You can also allow others only to read from a file.
- Or you can allow others only to write to a file.

A user with root privileges can access any file on the system. Anyone who can gain root privileges has full access to all files, regardless of the file's owner or access permissions. The owner of a file controls which users have permission to access the file and how those users can access it. When you own a file, you can use the chmod (change mode) utility to change access permissions for that file. You can specify symbolic (relative) or numeric (absolute) arguments to chmod.

Symbolic Arguments to chmod



```
[divya@localhost divya]$ ls -l Syllabus
-rw-rw-r--    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$ chmod a+w Syllabus
[divya@localhost divya]$ ls -l Syllabus
-rw-rw-rw-    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$
```

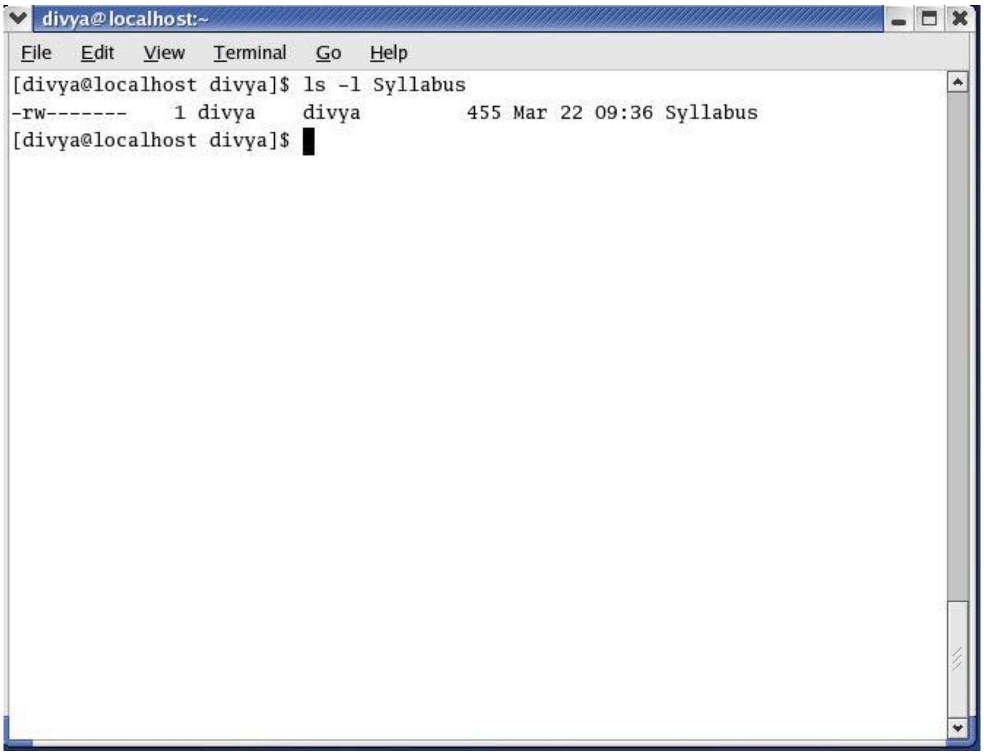


```
[divya@localhost divya]$ ls -l Syllabus
-rw-rw-rw-    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$ chmod g-rw Syllabus
[divya@localhost divya]$ ls -l Syllabus
-rw----rw-    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$
```

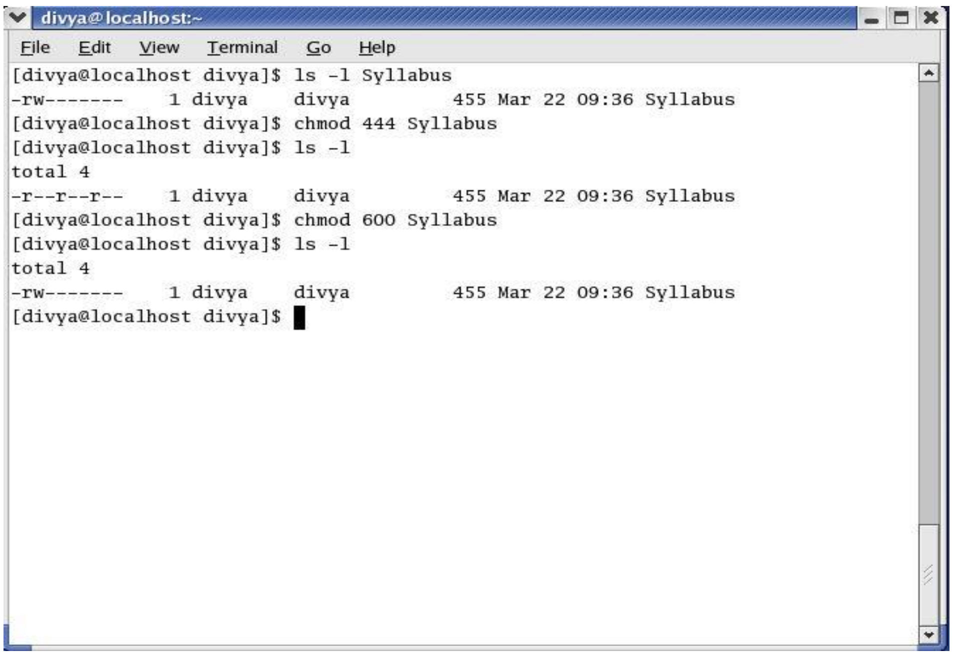
When using chmod, many people assume that the o stands for owner; it does not. The o stands for other, whereas u stands for owner (user).UGO (user-group-other)

Numeric Arguments to chmod

You can also use numeric arguments to specify permissions with chmod. In place of the letters and symbols specifying permissions used, here numeric arguments comprise three octal digits. The first digit specifies permissions for the owner, the second for the group, and the third for other users. A 1 gives the specified user(s) execute permission, 2 gives write permission, and 4 gives read permission. Construct the digit representing the permissions for the owner, group, or others by ORing (adding) the appropriate values.



```
[divya@localhost divya]$ ls -l Syllabus
-rw-------    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$
```



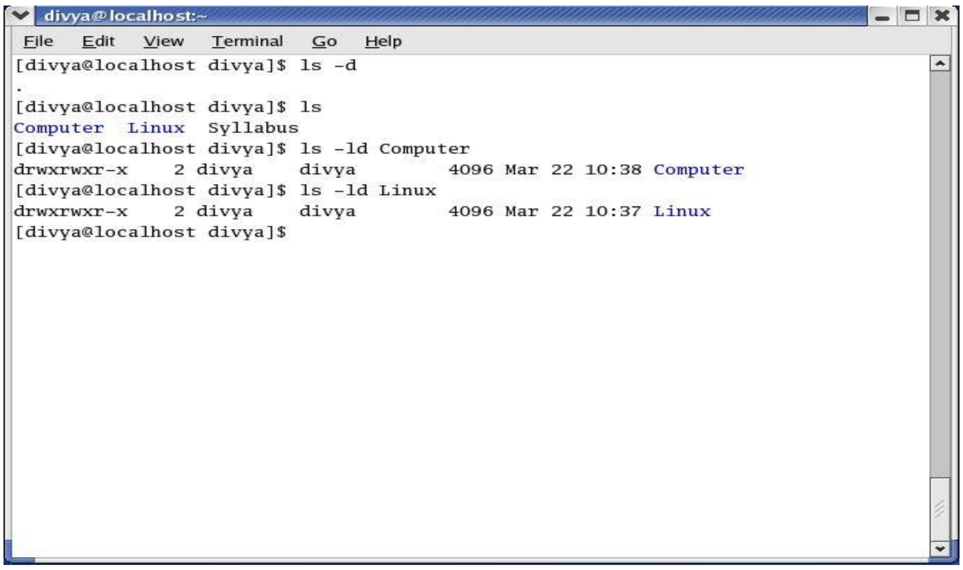
```
[divya@localhost divya]$ ls -l Syllabus
-rw-------    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$ chmod 444 Syllabus
[divya@localhost divya]$ ls -l
total 4
-r--r--r--    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$ chmod 600 Syllabus
[divya@localhost divya]$ ls -l
total 4
-rw-------    1 divya    divya         455 Mar 22 09:36 Syllabus
[divya@localhost divya]$
```

Setuid and Setgid Permissions

When you execute a file that has setuid (set user ID) permission, the process executing the file takes on the privileges of the file's owner. For example, if you run a setuid program that removes all files in a directory, you can remove files in any of the file owner's directories, even if you do not normally have permission to do so. In a similar manner, setgid (set group ID) permission gives the process executing the file the privileges of the group the file is associated with.

Directory Access Permissions

Access permissions have slightly different meanings when they are used with directories. Although the three types of users can read from or write to a directory, the directory cannot be executed. Execute permission is redefined for a directory: It means that you can cd into the directory and/or examine files that you have permission to read from in the directory. It has nothing to do with executing a file. When you have only execute permission for a directory, you can use ls to list a file in the directory if you know its name. You cannot use ls without an argument to list the entire contents of the directory.



```
[divya@localhost divya]$ ls -d
.
[divya@localhost divya]$ ls
Computer  Linux   Syllabus
[divya@localhost divya]$ ls -ld Computer
drwxrwxr-x    2 divya    divya        4096 Mar 22 10:38 Computer
[divya@localhost divya]$ ls -ld Linux
drwxrwxr-x    2 divya    divya        4096 Mar 22 10:37 Linux
[divya@localhost divya]$
```

The d at the left end of the line that ls displays indicates that Linux is a directory. The person has read, write, and execute permissions; members of the group also have read, write and execute permissions; and other users have only read and execute permissions.

6.7 Access Control Lists

The ACLs provide finer-grained control over which users can access specific directories and files than do traditional Linux permissions. Using ACLs you can specify the ways in which each of several users can access a directory or file.

Reduced Performance

Because ACLs can reduce performance, do not enable them on filesystems that hold system files, where the traditional Linux permissions are sufficient. Also be careful when moving, copying, or archiving files: Not all utilities preserve ACLs. In addition, you cannot copy ACLs to filesystems that do not support ACLs.

Rules

An ACL comprises a set of rules. **Rule:** It specifies how a specific user or group can access the file that the ACL is associated with. There are two kinds of rules: **access rules** and **default rules.**

- **Access rule:** It specifies access information for a single file or directory.
- **Default ACL:** It pertains to a directory only; it specifies default access information (an ACL) for any file in the directory that is not given an explicit ACL.

Most utilities do not preserve ACLs

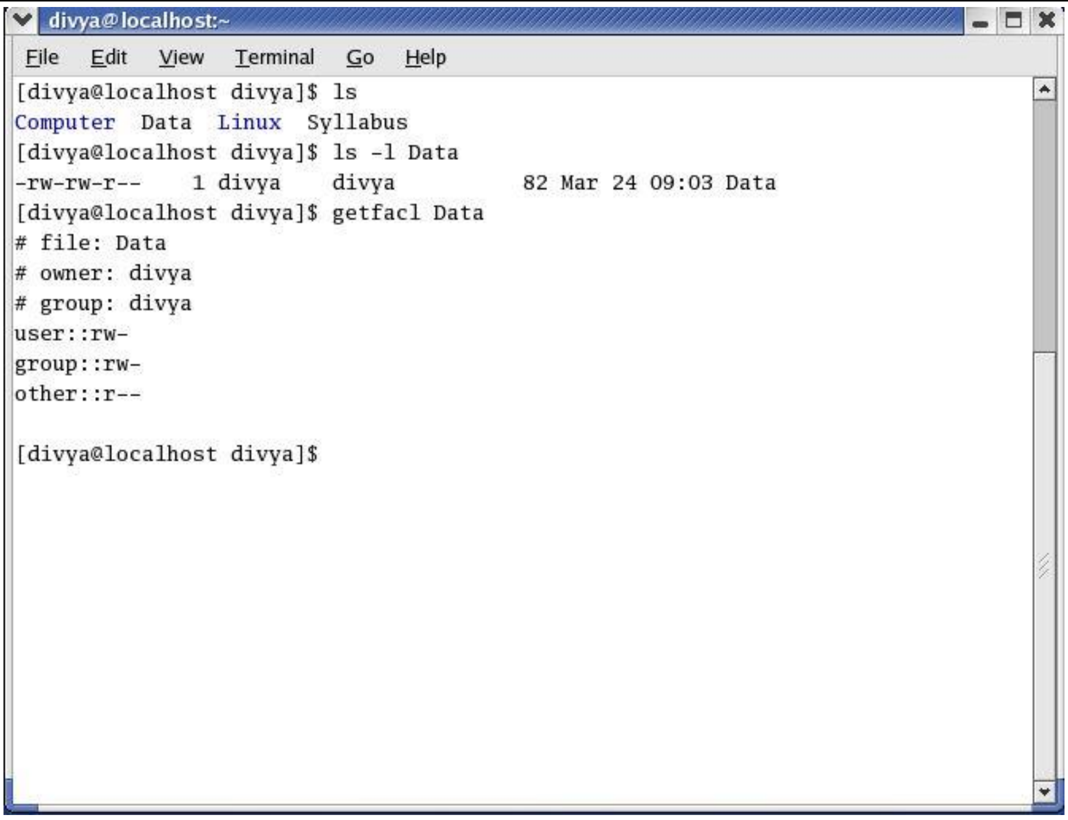
- **cp utility:** When used with the –p (preserve) or –a (archive) option, cp preserves ACLs when it copies files.
- **mv utility:** It also preserves ACLs.
- Other utilities, such as tar, cpio, and dump, do not support ACLs.

Enabling ACLs

Before you can use ACLs you must install the acl software package. Linux officially supports ACLs on ext2, ext3, and ext4 filesystems only, although informal support for ACLs is available on other filesystems. To use ACLs on an ext filesystem, you must mount the device with the acl option. For example, if you want to mount the device represented by /home so you can use ACLs on files in /home, you can add acl to its options list in /etc/fstab:

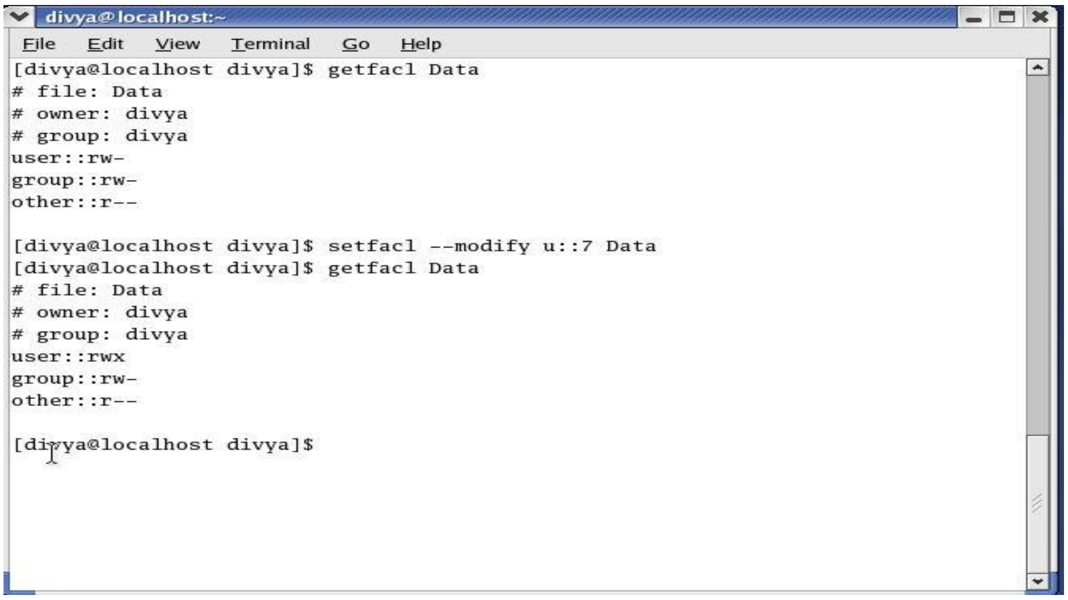
```
$ grep home /etc/fstab
LABEL=/home /home ext3 defaults,acl 1 2
```

After changing fstab, you need to remount /home before you can use ACLs.

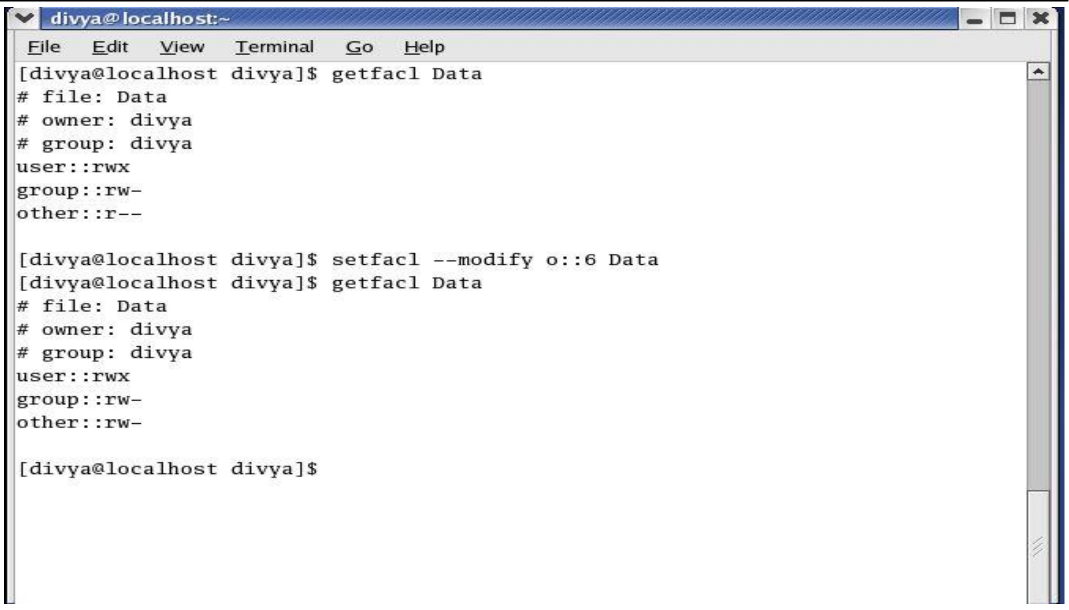


```
[divya@localhost divya]$ ls
Computer  Data  Linux  Syllabus
[divya@localhost divya]$ ls -l Data
-rw-rw-r--    1 divya    divya         82 Mar 24 09:03 Data
[divya@localhost divya]$ getfacl Data
# file: Data
# owner: divya
# group: divya
user::rw-
group::rw-
other::r--

[divya@localhost divya]$
```

The first three lines of the getfacl output comprise the header; they specify the name of the file, the owner of the file, and the group the file is associated with. In the line that starts with user, the two colons (::) with no name between them indicate that the line specifies the permissions for the owner of the file. Similarly, the two colons in the group line indicate that the line specifies permissions for the group the file is associated with. The two colons following other are there for consistency: No name can be associated with other.



```
[divya@localhost divya]$ getfacl Data
# file: Data
# owner: divya
# group: divya
user::rw-
group::rw-
other::r--

[divya@localhost divya]$ setfacl --modify u::7 Data
[divya@localhost divya]$ getfacl Data
# file: Data
# owner: divya
# group: divya
user::rwx
group::rw-
other::r--

[divya@localhost divya]$
```



```
[divya@localhost divya]$ getfacl Data
# file: Data
# owner: divya
# group: divya
user::rwx
group::rw-
other::r--

[divya@localhost divya]$ setfacl --modify o::6 Data
[divya@localhost divya]$ getfacl Data
# file: Data
# owner: divya
# group: divya
user::rwx
group::rw-
other::rw-

[divya@localhost divya]$
```

Setting up ACL :

1) To add permission for user

setfacl -m "u:user:permissions" /path/to/file

2) To add permissions for a group

setfacl -m "g:group:permissions" /path/to/file

3) To allow all files or directories to inherit ACL entries from the directory it is within

setfacl -dm "entry" /path/to/dir

4) To remove a specific entry

setfacl -x "entry" /path/to/file

5) To remove all entries

setfacl -b path/to/file

Modifying ACL using setfacl:

To add permissions for a user (user is either the user name or ID): **# setfacl -m "u:user:permissions".** To add permissions for a group (group is either the group name or ID): **# setfacl -m "g:group:permissions".** To allow all files or directories to inherit ACL entries from the directory it is within: **# setfacl -dm "entry".**

View ACL :

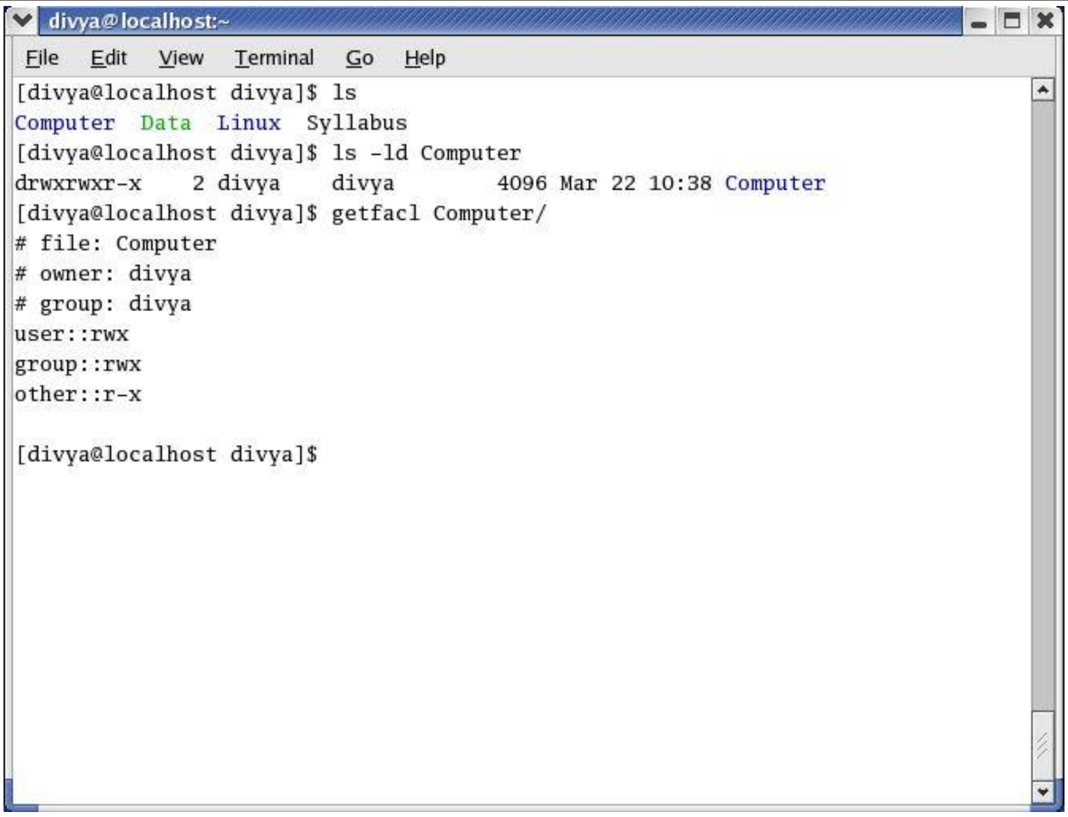
To show permissions : **# getfacl filename**

Remove ACL :

If you want to remove the set ACL permissions, use setfacl command with **-b** option.

Setting Default Rules for a Directory

The setfacl command uses the **–d** (default) option to add two default rules to the ACL for dir. These rules apply to all files in the dir directory that do not have explicit ACLs.

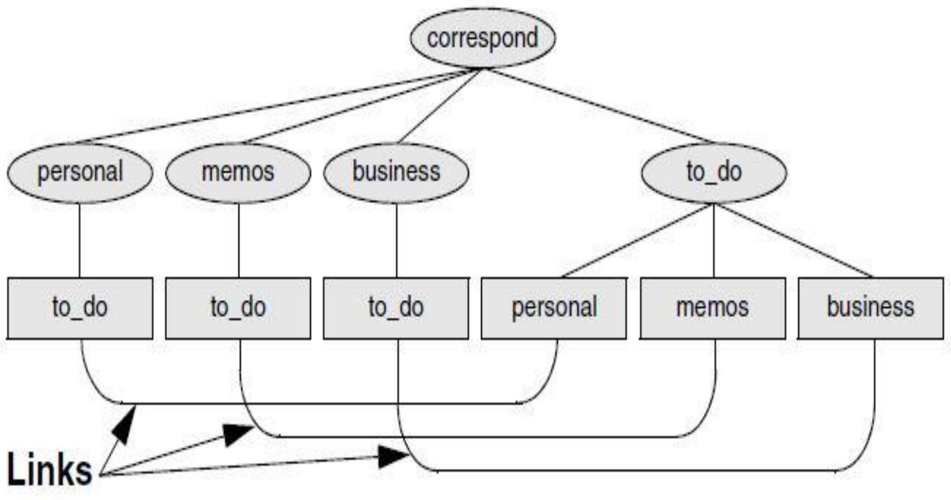


```
[divya@localhost divya]$ ls
Computer  Data  Linux  Syllabus
[divya@localhost divya]$ ls -ld Computer
drwxrwxr-x    2 divya    divya        4096 Mar 22 10:38 Computer
[divya@localhost divya]$ getfacl Computer/
# file: Computer
# owner: divya
# group: divya
user::rwx
group::rwx
other::r-x

[divya@localhost divya]$
```

6.8 Links

A link is a pointer to a file. Every time you create a file by using vim, touch, cp, or any other means, you are putting a pointer in a directory. This pointer associates a filename with a place on the disk. When you specify a filename in a command, you are indirectly pointing to the place on the disk that holds the information you want.

Using links to cross-classify files



Using links

Sharing files can be useful when two or more people are working on the same project and need to share some information. You can make it easy for other users to access one of your files by creating additional links to the file. To share a file with another user, first give the user permission to read from and write to the file. You may also have to change the access permissions of the parent directory of the file to give the user read, write, or execute permission. Once the permissions are appropriately set, the user can create a link to the file so that each of you can access the file from your separate directory hierarchies. A link can also be useful to a single user with a large directory hierarchy. You can create links to cross-classify files in your directory hierarchy, using different classifications for different tasks.

Hard Links

A hard link to a file appears as another file. If the file appears in the same directory as the linked-to file, the links must have different filenames because two files in the same directory cannot have the same name. You can create a hard link to a file only from within the filesystem that holds the file. A hard link is a direct link to the data on disk. This means data can be accessed directly via an original filename or a hard link. Both the original file and the hard link are direct links to the data on disk. The use of a hard link allows multiple filenames to be associated with the same data on disk.

Soft Links

These are also known as symbolic links or symlink. It refers to a symbolic path indicating the abstract location of another file. A symbolic link (also sometimes known as a soft link) does not link directly to the data on disk but to another link to the data on disk. On most operating systems folders may only be linked using a symlink. Symbolic links can link across file systems to link a folder on an external hard drive.

Hard link vs. Soft link in Linux

Hard links cannot link directories and cross file system boundaries. Soft or symbolic links are just like hard links. It allows to associate multiple filenames with a single file. However, symbolic links allows: to create links between directories and can cross file system boundaries. These links behave differently when the source of the link is moved or removed. Symbolic links are not updated. Hard links always refer to the source, even if moved or removed.

Summary

- The oldest utility, who, produces a list of users who are logged in on the local system, the device each person is using, and the time each person logged in. The w and finger utilities show more detail, such as each user's full name and the command line each user is running.
- On systems where security is a concern, the system administrator may disable finger.
- Email enables you to communicate with users on the local system and, if the installation is part of a network, with other users on the network. If you are connected to the Internet, you can communicate electronically with users around the world.
- The popular graphical email program is sylpheed in Linux.
- On a standard Linux system, each user starts with one directory, to which the user can add subdirectories to any desired level.
- One major strength of the Linux filesystem is its ability to adapt to users' needs.
- When you refer to the tree, up is toward the root and down is away from the root.
- If you keep the filenames short, they are easy to type. The disadvantage of short filenames is that they are typically less descriptive than long filenames.

Keywords

- Finger: This utility is used to retrieve information about users on remote systems if the local system is attached to a network.
- Filesystem: A filesystem is a set of data structures that usually resides on part of a disk and that holds directories of files.
- Ordinary files: These are simply files, appear at the ends of paths that cannot support other paths.
- Directory files: These are also referred to as directories or folders, are the points that other paths can branch off from.
- Pathname: A pathname is a series of names that trace a path along branches from one file to another.
- Startup files: These appear in your home directory, give the shell and other programs information about you and your preferences.
- Cd utility: The cd (change directory) utility makes another directory the working directory but does not change the contents of the working directory.

Self Assessment

1. Which of these utilities is used to change to another working directory?

A. mkdir
B. cd
C. mv
D. None of the above

2. Any pathname that does not begin with _ is a relative pathname.

A. /
B. ~
C. Either / or ~
D. None of the above

3. When you refer to the tree, _____ is towards the root and ____ is away from the root.

A. up, down
B. down, up
C. left, right
D. right, left

4. Which of these files appear at the ends of paths that cannot support other paths?

A. Directory files
B. Ordinary files
C. Base files
D. None of the above

5. Which of these builtin is used to display the pathname of the working directory?

A. pwd
B. work
C. path

D. None of the above

6. With a slash (/), we represent
A. Home directory
B. Root directory
C. Path directory
D. None of the above

7. Which of these utilities is used for communication when the recipient is not logged in?
A. echo
B. email
C. cat
D. None of the above

8. The utility *who* produces the
A. List of users who are logged in on the local system
B. Device each person is using
C. The time each person is logged in
D. All of the above mentioned

9. On those systems, where security is concern, the system administrator can disable ___________
A. echo
B. finger
C. ls
D. None of the above

10. Which of these is used for establishing and ending the conversation?
A. o
B. oo
C. Both of the above
D. None of the above

11. Which of these utilities preserve ACLs?
A. cp
B. mv
C. Both cp and mv
D. None of the above

12. Which of these links exists in Linux?
A. Hard links
B. Soft links
C. Water links

D. Any of these

13. Which of these files are the points that other paths can branch off from?

A. Directory files
B. Ordinary files
C. Base files
D. None of the above

14. Which of these builtin is used to display the pathname of the working directory?

A. pwd
B. work
C. path
D. None of the above

15. Which of these files appear at the ends of paths that cannot support other paths?

A. Directory files
B. Ordinary files
C. Base files
D. None of the above

Answers for Self Assessment

1.	B	2.	C	3.	A	4.	B	5.	A
6.	B	7.	B	8.	D	9.	B	10.	C
11.	C	12.	C	13.	A	14.	A	15.	B

Review Questions

1. Which utilities can be used to obtain system and user information? Explain.
2. How can we communicate with other users in Linux?
3. What is a filesystem? Write the strengths of linux filesystem.
4. What are directory files and ordinary files? Explain their differences with examples.
5. What is a pathname? Write its types.
6. What is a working directory? Explain its significance. Write the utilities which are used to work with directories.
7. What are access permissions? Explain in detail.
8. What is an access control lists? Write its features. How can we set up ACLs?
9. What is a link? Explain the difference between hard links and soft links.

Further Readings

Mark G. Sobell, A Practical Guide to Fedora and Red Hat Enterprise Linux, Fifth Edition, Pearson Education, Inc.

Web Links

https://www.redhat.com/sysadmin/linux-access-control-lists

https://www.geeksforgeeks.org/soft-hard-links-unixlinux/

Unit 07: The Shell and popular Editors

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand the Command Line
- Understand the Standard Input and Standard Output
- Understand about filename generation and pathname extension
- Understand about builtins
- Know how to use and features of vim
- Understand the command mode and input mode
- Know the difference between emacs and vim
- Understand the functionalities and basic editing commands in emacs

Introduction

The important component of any computer system is an operating system. An operating system is the software that directly manages a system's hardware and resources, like CPU, memory, and storage. The OS sits between applications and hardware and makes the connections between all your software and the physical resources that do the work.Think about an OS like a car engine. An engine can run on its own, but it becomes a functional car when it's connected with a transmission, axles, and wheels. Without the engine running properly, the rest of the car won't work. An Operating system is made of many components, but its two prime components are – kernel and shell.

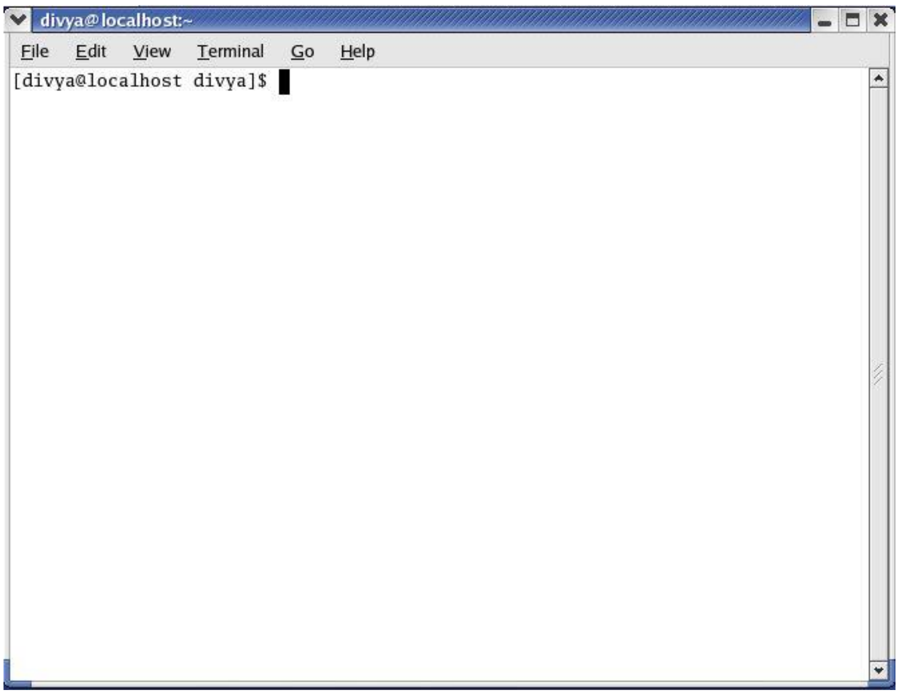
A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one. It is the base component of the OS. Without it, the OS doesn't work. The kernel manages the system's resources and communicates with the hardware. It's responsible for memory, process, and file management.A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it. When you run the terminal, the Shell issues a command prompt (usually $), where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal. The Shell wraps around the delicate interior of an OS protecting it from accidental damage. Hence the name Shell.

7.1 The Shell

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finish executing, it displays that program's output.Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions. You can cause the shell to execute various types of programs—such as shell scripts, application programs, and programs you have written—in the same way. The line that contains the command, including any arguments, is called the command line.The syntax for a basic command line is

```
command [arg1] [arg2] ... [argn] RETURN
```

The prompt, $, which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command.Shell reads your input after you press Enter



```
[divya@localhost divya]$
```

Rules for commands

There are some rules for writing the commands. These are:

- One or more SPACEs must separate elements on the command line.
- The command is the name of the command, arg1 through argn are arguments, and RETURN is the keystroke that terminates all command lines.
- The brackets in the command-line syntax indicate that the arguments they enclose are optional.
- Not all commands require arguments.

Command Name

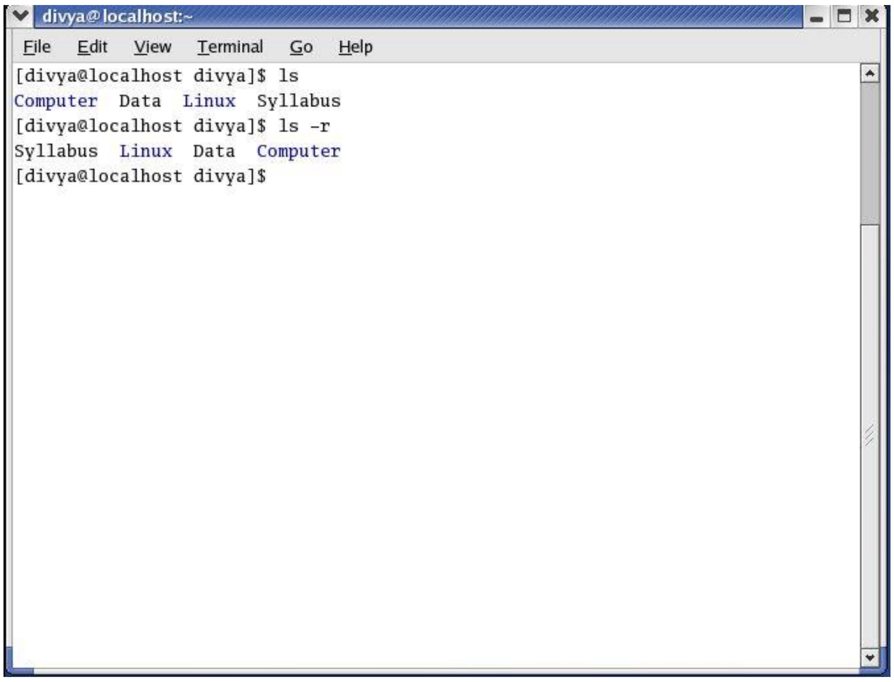
Some useful Linux command lines consist of only the name of the command without any arguments. Commands that require arguments typically give a short error message, called a usage message.

Arguments

On the command line each sequence of nonblank characters is called a token or word. An argument is a token, such as a filename, string of text, number, or other object that a command acts on. For example, the argument to a vim or emacs command is the name of the file you want to edit.The following command line shows cp copying the file named temp to temp copy: $ cp temp temp copy. Arguments are numbered starting with the command itself, which is argument zero. cp arument-0, temp argument-1 and temp copy argument-2.The cp utility requires at least two arguments on the command line.

Options

An option is an argument that modifies the effects of a command.You can frequently specify more than one option, modifying the command in several different ways.Most utilities require you to prefix options with a single hyphen. However, this requirement is specific to the utility and not the shell. GNU program options are frequently preceded by two hyphens in a row. For example, --help generates a (sometimes extensive) usage message.



```
[divya@localhost divya]$ ls
Computer  Data  Linux  Syllabus
[divya@localhost divya]$ ls -r
Syllabus  Linux  Data  Computer
[divya@localhost divya]$
```

Combining options

When you need to use several options, you can usually group multiple single-letter options into one argument that starts with a single hyphen; do not put SPACEs between the options. You cannot combine options that are preceded by two hyphens in this way. Specific rules for combining options depend on the program you are running.

Option arguments

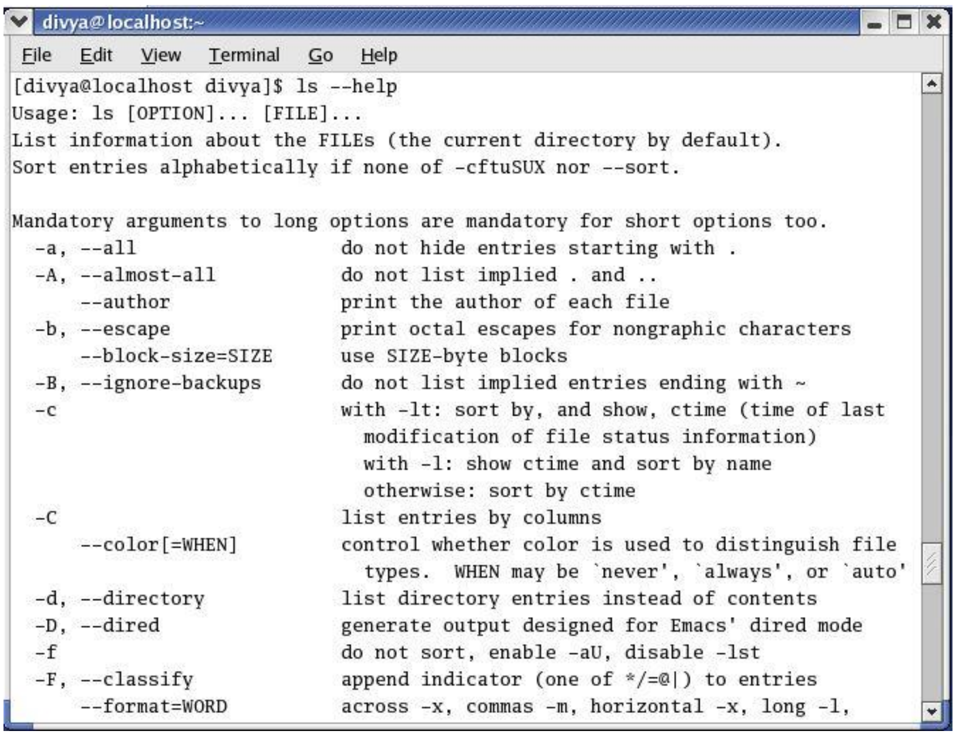
Some utilities have options that themselves require arguments. For example, the gcc utility has a –o option that must be followed by the name you want to give the executable file that gcc generates. Typically, an argument to an option is separated from its option letter by a SPACE.

Arguments that start with a hyphen

Another convention allows utilities to work with arguments, such as filenames, that start with a hyphen. If a file's name is –l, the following command is ambiguous:

$ ls -l

This command could mean you want ls to display a long listing of all files in the working directory or a listing of the file named –l..



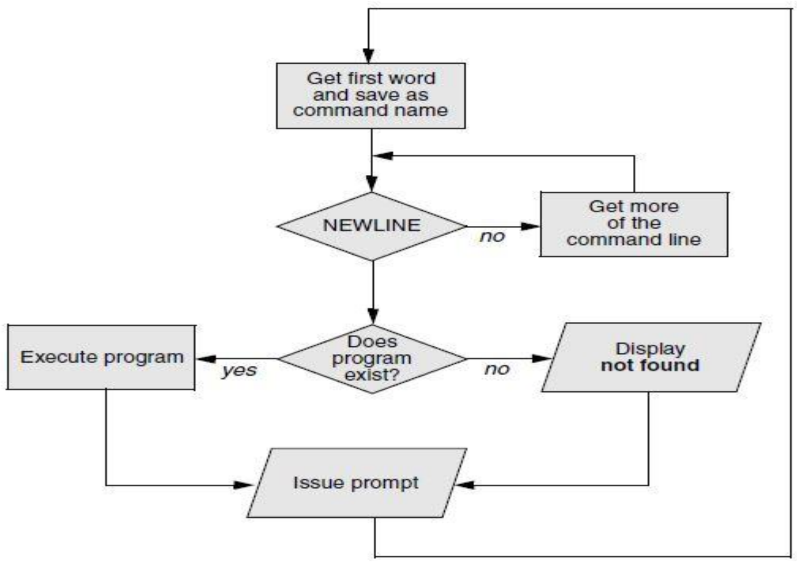
```
[divya@localhost divya]$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuSUX nor --sort.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not hide entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               print the author of each file
  -b, --escape               print octal escapes for nongraphic characters
      --block-size=SIZE      use SIZE-byte blocks
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information)
                               with -l: show ctime and sort by name
                               otherwise: sort by ctime
  -C                         list entries by columns
      --color[=WHEN]         control whether color is used to distinguish file
                               types.  WHEN may be `never', `always', or `auto'
  -d, --directory            list directory entries instead of contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -lst
  -F, --classify             append indicator (one of */=@|) to entries
      --format=WORD          across -x, commas -m, horizontal -x, long -l,
```

Processing the Command Line

As you enter a command line, the Linux tty device driver (part of the Linux kernel) examines each character to see whether it must take immediate action.

Key	Result
CONTROL-H	To erase a character
CONTROL-U	To kill a line
CONTROL-W	To erase a word

When the character you entered does not require immediate action, the device driver stores the character in a buffer and waits for additional characters. When you press RETURN, the device driver passes the command line to the shell for processing.



Parsing the command line

When the shell processes a command line, it looks at the line as a whole and parses (breaks) it into its component parts. Next, the shell looks for the name of the command. Usually, the name of the command is the first item on the command line after the prompt (argument zero).The shell typically takes the first characters on the command line up to the first blank (TAB or SPACE) and then looks for a command with that name. The command name (the first token) can be specified on the command line either as a simple filename or as a pathname. For example, you can call the ls command in either of the following ways:

```
$ ls
$ /bin/ls
```

Executing the Command Line

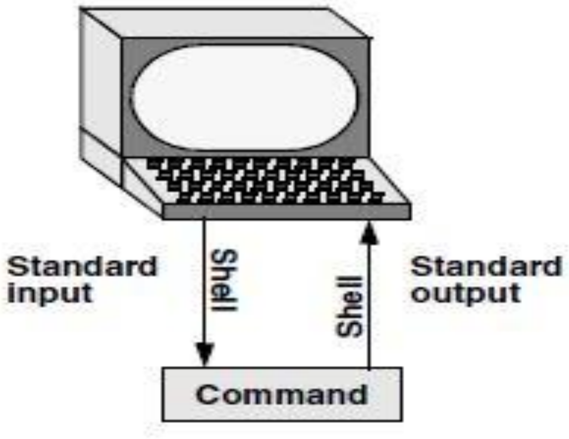
If it finds an executable file with the same name as the command, the shell starts a new process. A process is the execution of a command by Linux. The shell makes each command-line argument, including options and the name of the command, available to the called program. While the command is executing, the shell waits for the process to finish. At this point the shell is in an inactive state called sleep. When the program finishes execution, it passes its exit status to the shell. The shell then returns to an active state (wakes up), issues a prompt, and waits for another command.

Editing the Command Line

You can repeat and edit previous commands and edit the current command line.

7.2 Standard Input and Standard Output

Standard output is a place a program can send information, such as text. The program never "knows" where the information it sends to standard output is going. The information can go to a printer, an ordinary file, or the screen. By default, the shell directs standard output from a command to the screen.



Standard input is a place that a program gets information from. As with standard output the program never "knows" where the information comes from. In addition to standard input and standard output, a running program normally has a place to send error messages: standard error.

The Screen as a File

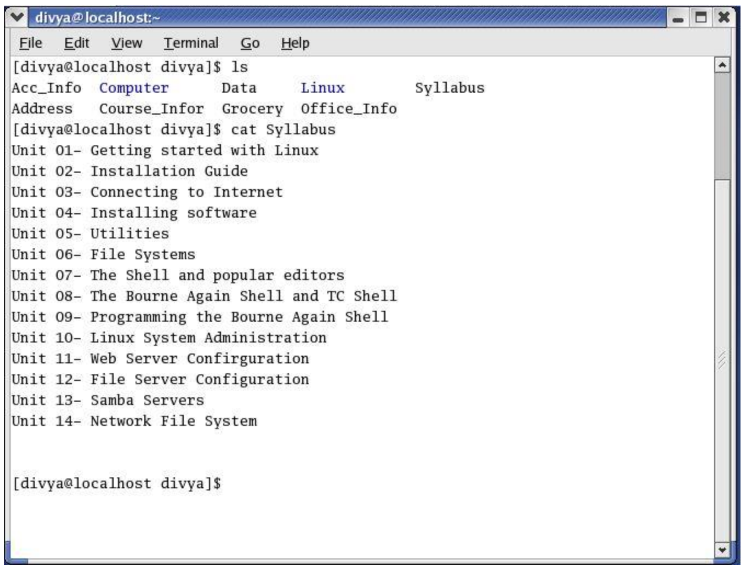
Linux have an additional type of file: a device file. A device file resides in the Linux file structure, usually in the /dev directory, and represents a peripheral device, such as a screen and keyboard, printer, or disk drive. The device name that the who utility displays after your username is the filename of the screen and keyboard.

The Keyboard and Screen as Standard Input and Standard Output

When you first log in, the shell directs standard output of your commands to the device file that represents the screen. Directing output in this manner causes it to appear on the screen.

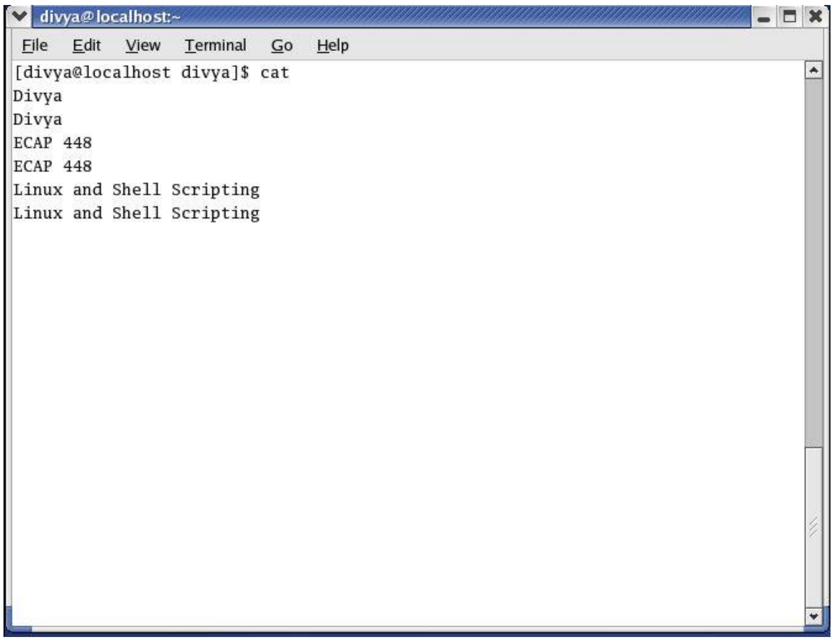
cat utility

The cat utility provides a good example of the way the keyboard and screen function as standard input and standard output, respectively. When you use cat, it copies a file to standard output. Because the shell directs standard output to the screen, cat displays the file on the screen. Up to this point cat has taken its input from the filename (argument) you specify on the command line.



```
[divya@localhost divya]$ ls
Acc_Info  Computer     Data     Linux       Syllabus
Address   Course_Infor Grocery  Office_Info
[divya@localhost divya]$ cat Syllabus
Unit 01- Getting started with Linux
Unit 02- Installation Guide
Unit 03- Connecting to Internet
Unit 04- Installing software
Unit 05- Utilities
Unit 06- File Systems
Unit 07- The Shell and popular editors
Unit 08- The Bourne Again Shell and TC Shell
Unit 09- Programming the Bourne Again Shell
Unit 10- Linux System Administration
Unit 11- Web Server Confirguration
Unit 12- File Server Configuration
Unit 13- Samba Servers
Unit 14- Network File System


[divya@localhost divya]$
```

When you do not give cat any argument (that is, when you give the command cat followed immediately by RETURN), cat takes its input from standard input. Thus, when called without an argument, cat copies standard input to standard output, one line at a time.



```
[divya@localhost divya]$ cat
Divya
Divya
ECAP 448
ECAP 448
Linux and Shell Scripting
Linux and Shell Scripting
```

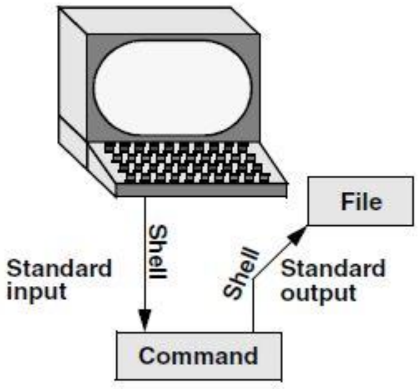
Because the shell associate cat's standard input with the keyboard and cat's standard output with the screen, when you type a line of text cat copies the text from standard input (the keyboard) to standard output (the screen).The cat utility keeps copying text until you enter CONTROL-D on a line by itself. Pressing CONTROL-D sends an EOF (end of file) signal to cat to indicate that it has reached the end of standard input and there is no more text for it to copy. The cat utility then finishes execution and returns control to the shell, which displays a prompt.

7.3 Redirection

The term redirection encompasses the various ways you can cause the shell to alter where standard input of a command comes from and where standard output goes to.By default, the shell associates' standard input and standard output of a command with the keyboard and the screen.You can cause the shell to redirect standard input or standard output of any command by associating the input or output with a command or file other than the device file representing the keyboard and the screen.

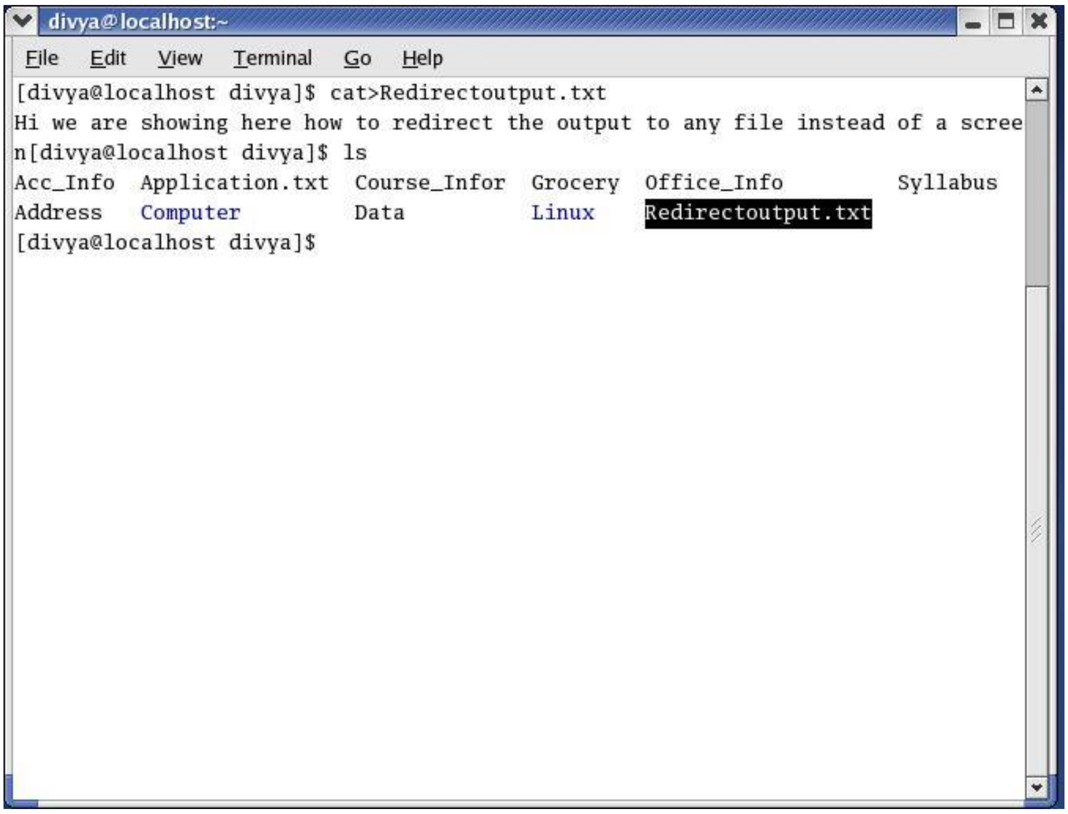
Redirecting Standard Output

The redirect output symbol (>) instructs the shell to redirect the output of a command to the specified file instead of to the screen.

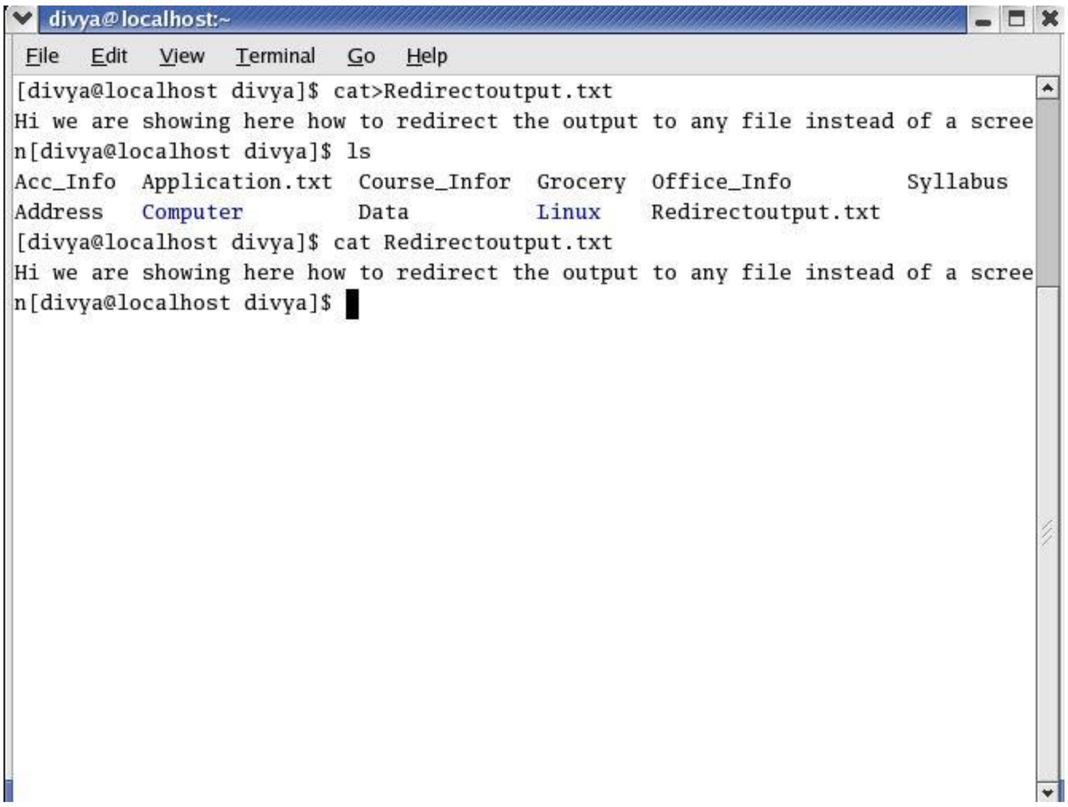


The format of a command line that redirects output is command [arguments] > filename

where command is any executable program (such as an application program or a utility), arguments are optional arguments, and filename is the name of the ordinary file the shell redirects the output to.

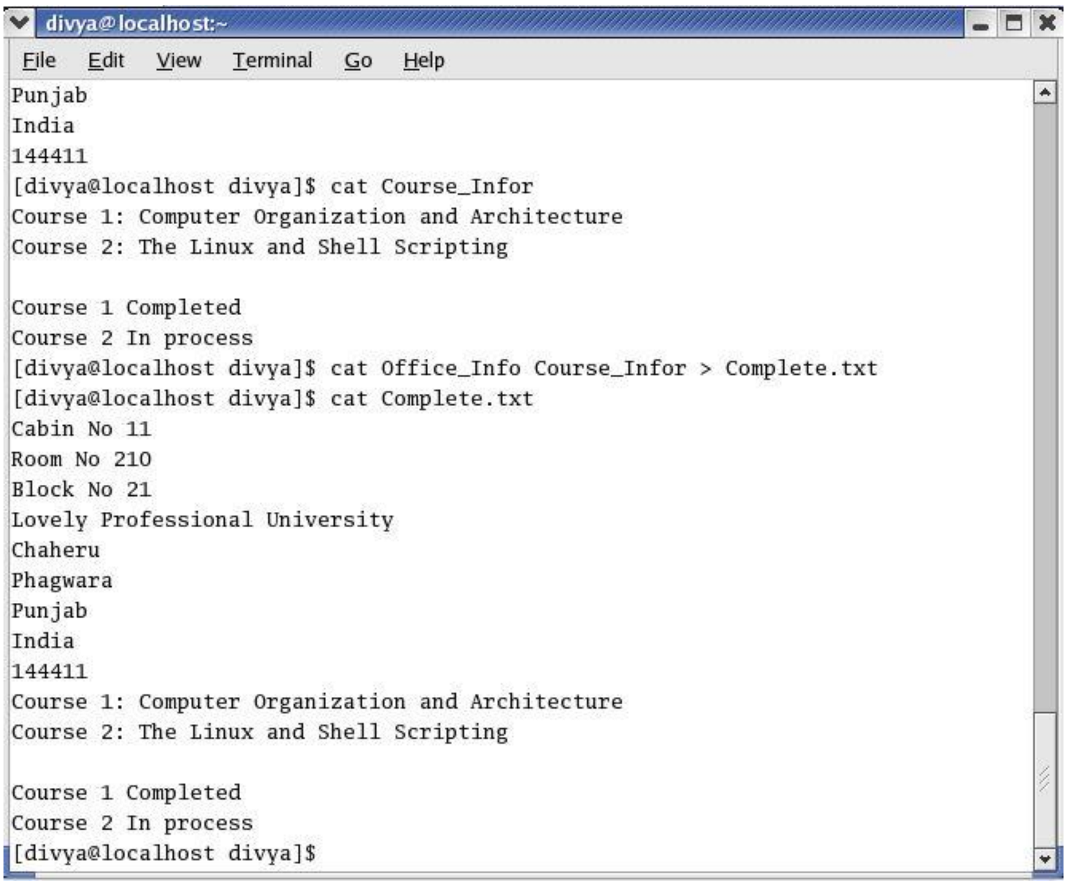


```
[divya@localhost divya]$ cat>Redirectoutput.txt
Hi we are showing here how to redirect the output to any file instead of a scree
n[divya@localhost divya]$ ls
Acc_Info   Application.txt  Course_Infor  Grocery  Office_Info        Syllabus
Address    Computer         Data          Linux    Redirectoutput.txt
[divya@localhost divya]$
```



```
[divya@localhost divya]$ cat>Redirectoutput.txt
Hi we are showing here how to redirect the output to any file instead of a scree
n[divya@localhost divya]$ ls
Acc_Info   Application.txt  Course_Infor  Grocery  Office_Info        Syllabus
Address    Computer         Data          Linux    Redirectoutput.txt
[divya@localhost divya]$ cat Redirectoutput.txt
Hi we are showing here how to redirect the output to any file instead of a scree
n[divya@localhost divya]$
```

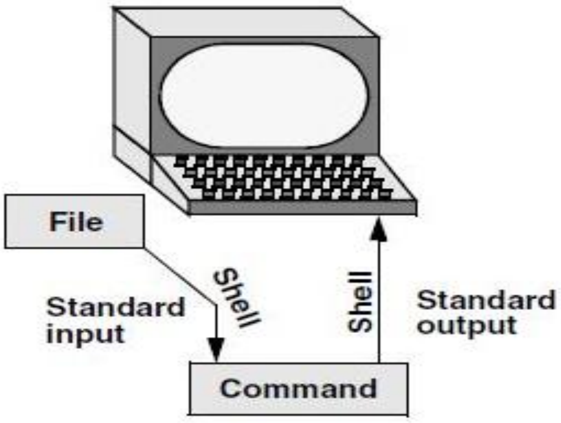
The redirect output symbol on the command line causes the shell to associate cat's standard output with the redirectoutput.txt file specified on the command line.Redirecting standard output from cat is a handy way to create a file without using an editor.The drawback is that once you enter a line and press RETURN, you cannot edit the text. While you are entering a line, the erase and kill keys work to delete text. This procedure is useful for creating short, simple files.The cat is used and the redirect output symbol to catenate (join one after the other—the derivation of the name of the cat utility) several files into one larger file.



```
Punjab
India
144411
[divya@localhost divya]$ cat Course_Infor
Course 1: Computer Organization and Architecture
Course 2: The Linux and Shell Scripting

Course 1 Completed
Course 2 In process
[divya@localhost divya]$ cat Office_Info Course_Infor > Complete.txt
[divya@localhost divya]$ cat Complete.txt
Cabin No 11
Room No 210
Block No 21
Lovely Professional University
Chaheru
Phagwara
Punjab
India
144411
Course 1: Computer Organization and Architecture
Course 2: The Linux and Shell Scripting

Course 1 Completed
Course 2 In process
[divya@localhost divya]$
```

Redirecting Standard Input

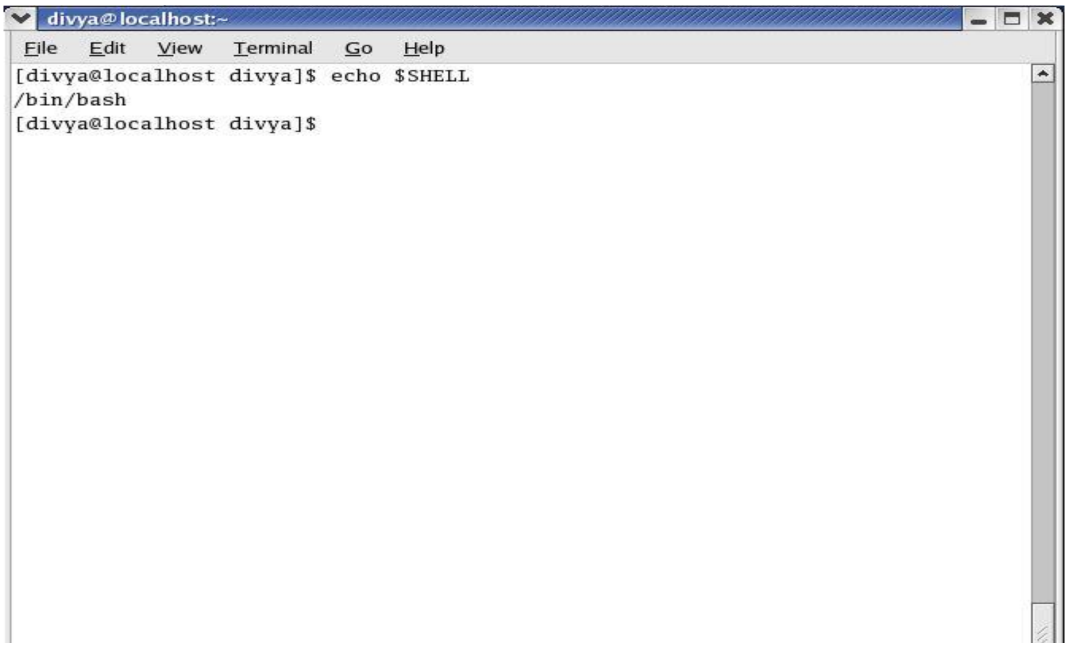
The redirect input symbol (<) instructs the shell to redirect a command's input to come from the specified file instead of from the keyboard.



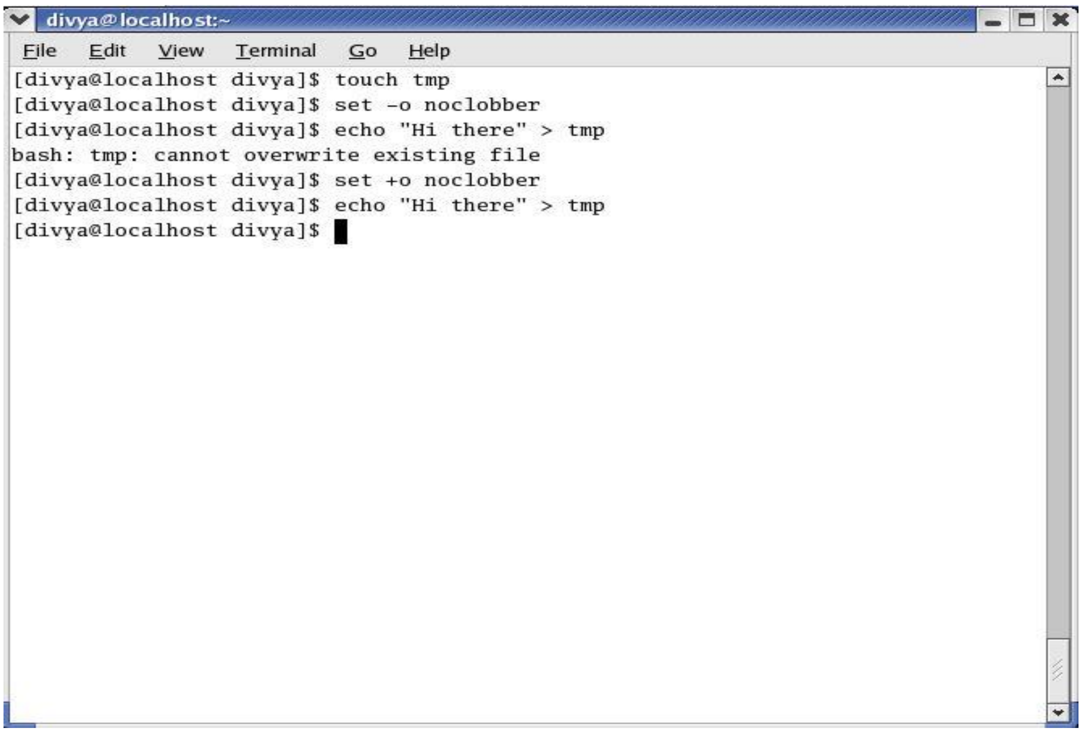
The format of a command line that redirects input is: command [arguments] < filename where command is any executable program (such as an application program or a utility), arguments are optional arguments, and filename is the name of the ordinary file the shell redirects the input from.

noclobber: Avoids Overwriting Files

The shell provides the noclobber feature that prevents overwriting a file using redirection.Under bash you can enable this feature by setting noclobber using the command set –onoclobber. The same command with +o unsets noclobber. Under tcsh use set noclobber and unset noclobber.With noclobber set, if you redirect output to an existing file, the shell displays an error message and does not execute the command.



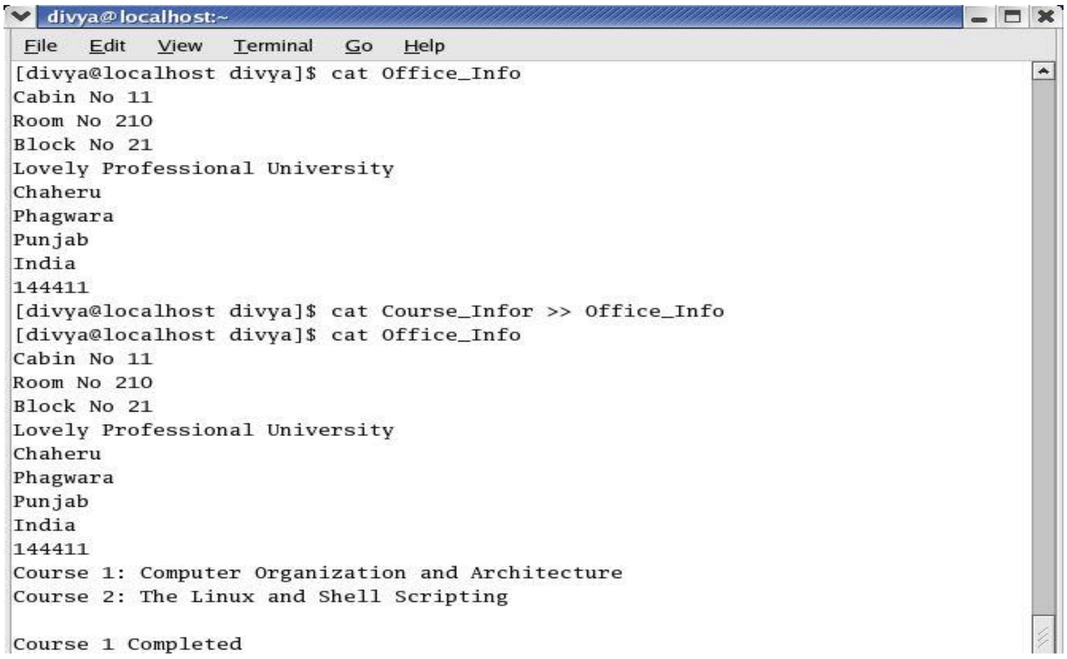
```
[divya@localhost divya]$ echo $SHELL
/bin/bash
[divya@localhost divya]$
```



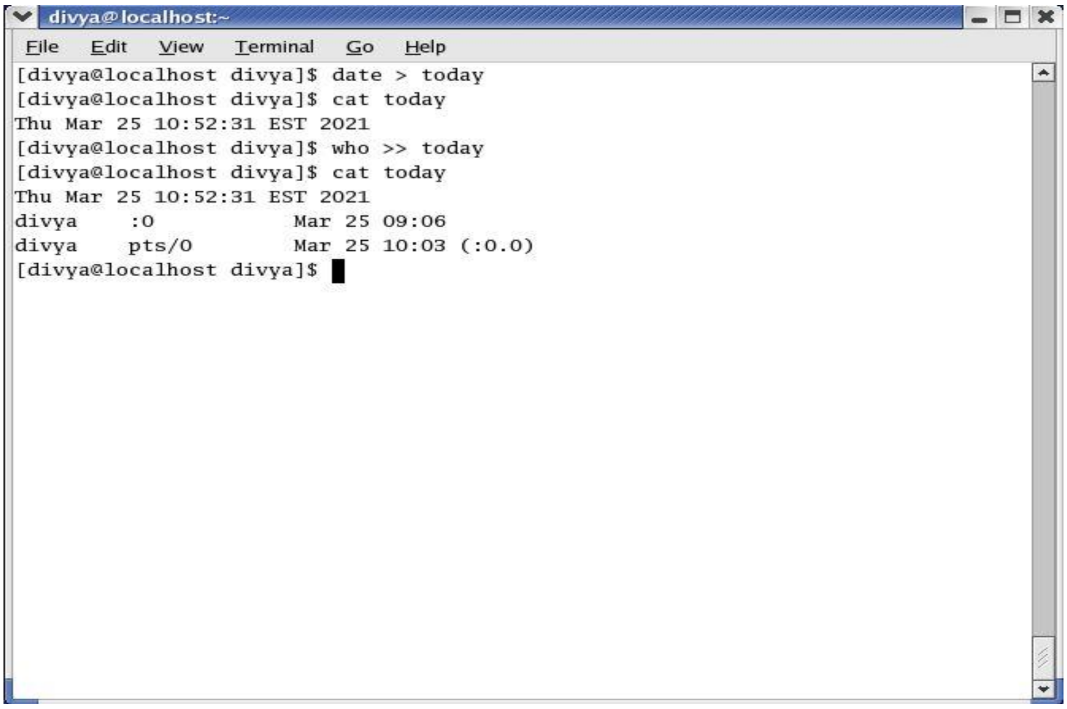
```
[divya@localhost divya]$ touch tmp
[divya@localhost divya]$ set -o noclobber
[divya@localhost divya]$ echo "Hi there" > tmp
bash: tmp: cannot overwrite existing file
[divya@localhost divya]$ set +o noclobber
[divya@localhost divya]$ echo "Hi there" > tmp
[divya@localhost divya]$ 
```

Appending Standard Output to a File

The append output symbol (>>) causes the shell to add new information to the end of a file, leaving existing information intact. This symbol provides a convenient way of catenating two files into one.



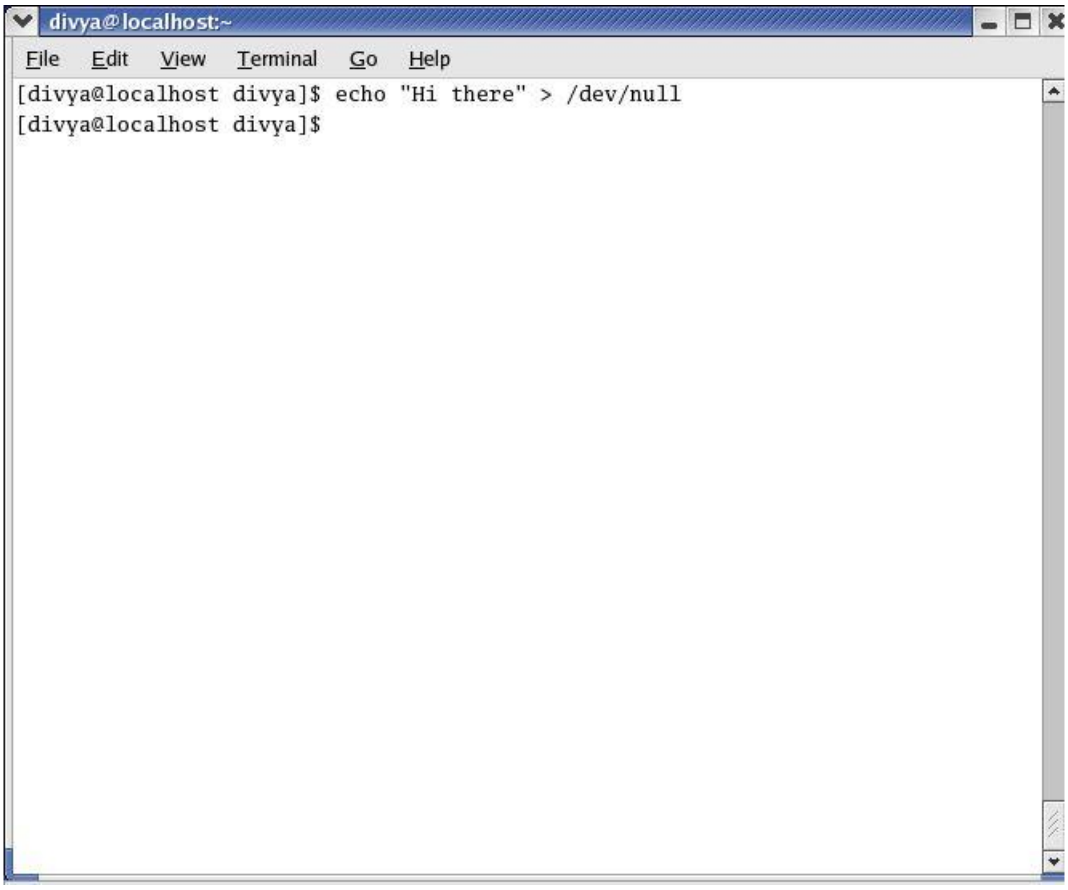
```
[divya@localhost divya]$ cat Office_Info
Cabin No 11
Room No 210
Block No 21
Lovely Professional University
Chaheru
Phagwara
Punjab
India
144411
[divya@localhost divya]$ cat Course_Infor >> Office_Info
[divya@localhost divya]$ cat Office_Info
Cabin No 11
Room No 210
Block No 21
Lovely Professional University
Chaheru
Phagwara
Punjab
India
144411
Course 1: Computer Organization and Architecture
Course 2: The Linux and Shell Scripting

Course 1 Completed
```



```
[divya@localhost divya]$ date > today
[divya@localhost divya]$ cat today
Thu Mar 25 10:52:31 EST 2021
[divya@localhost divya]$ who >> today
[divya@localhost divya]$ cat today
Thu Mar 25 10:52:31 EST 2021
divya    :0          Mar 25 09:06
divya    pts/0       Mar 25 10:03 (:0.0)
[divya@localhost divya]$
```

/dev/null: Making Data Disappear

The /dev/null device is a data sink, commonly referred to as a bit bucket. You can redirect output that you do not want to keep or see to /dev/null and the output will disappear without a trace.



```
[divya@localhost divya]$ echo "Hi there" > /dev/null
[divya@localhost divya]$
```

7.4 Pipes

The shell uses a pipe to connect standard output of one command to standard input of another command. A pipe (sometimes referred to as a pipeline) has the same effect as redirecting standard output of one command to a file and then using that file as standard input to another command. The symbol for a pipe is a vertical bar (|).The syntax of a command line using a pipe is

```
command_a [arguments] | command_b [arguments]
```

The preceding command line uses a pipe on a single command line to generate the same result as the following three command lines:

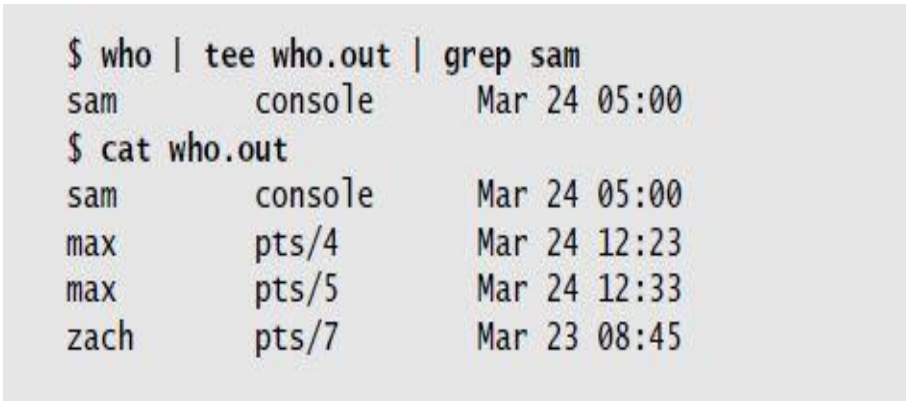
```
command_a [arguments] > temp
command_b [arguments] < temp
rm temp
```

7.5 Filters

A filter is a command that processes an input stream of data to produce an output stream of data. A command line that includes a filter uses a pipe to connect standard output of one command to the filter's standard input. Another pipe connects the filter's standard output to standard input of another command. Not all utilities can be used as filters. In this example, sort is a filter, taking standard input from standard output of who and using a pipe to redirect standard output to standard input of lpr. This command line sends the sorted output of who to the printer:$ who | sort | lpr

tee: Sends Output in Two Directions

The tee utility copies its standard input both to a file and to standard output. This utility is aptly named: It takes a single stream of input and sends the output in two directions.



```
$ who | tee who.out | grep sam
sam        console     Mar 24 05:00
$ cat who.out
sam        console     Mar 24 05:00
max        pts/4       Mar 24 12:23
max        pts/5       Mar 24 12:33
zach       pts/7       Mar 23 08:45
```

The output of who is sent via a pipe to standard input of tee. The tee utility saves a copy of standard input in a file named who outand sends a copy to standard output. Standard output of tee goes via a pipe to standard input of grep, which displays only those lines containing the string sam.

7.6 Running a Command in the Background

All commands up to this point have been run in the foreground. When you run a command in the foreground, the shell waits for it to finish before displaying another prompt and allowing you to continue. When you run a command in the background, you do not have to wait for the command to finish before running another command. A job is a series of one or more commands that can be connected by pipes. You can have only one foreground job in a window or on a screen, but you can have many background jobs. Multitasking is the running of more than one job at a time.

To run a job in the background, type an ampersand (&) just before the RETURN that ends the command line. The shell assigns a small number to the job and displays this job number between brackets. Following the job number, the shell displays the process identification (PID) number—a larger number assigned by the operating system. Each of these numbers identifies the job running in the background. The shell then displays another prompt, and you can enter another command. When the background job finishes, the shell displays a message giving both the job number and the command line used to run the command. This command runs in the background; it sends the output of ls through a pipe to lpr, which sends it to the printer.

```
$ ls -l | lpr&
[1] 22092
$
```

The [1] following the command line indicates that the shell has assigned job number 1 to this job. The 22092 is the PID number of the first command in the job. When this background job completes execution, the shell displays the message[1]+ Done ls -l | lp

Moving a Job from the Foreground to the Background

You can suspend a foreground job (stop it from running without aborting the job) by pressing the suspend key, usually CONTROL-Z. The shell then stops the process and disconnects standard input from the keyboard. You can put a suspended job in the background and restart it by using the bg command followed by the job number. You do not need to specify the job number when there is only one stopped job. Only the foreground job can take input from the keyboard.

To connect the keyboard to a program running in the background, you must bring it to the foreground. To do so, type fg without any arguments when only one job is in the background. When more than one job is in the background, type fg, or a percent sign (%), followed by the number of the job you want to bring into the foreground. The shell displays the command you used to start the job (promptme in the following example), and you can enter any input the program requires to continue:

```
bash $ fg 1
promptme
```

kill: Aborting a Background Job

The interrupt key (usually CONTROL-C) cannot abort a background process; you must use kill for this purpose. Follow kill on the command line with either the PID number of the process you want to abort or a percent sign (%) followed by the job number. If you forget a PID number, you can use the ps (process status) utility to display it. The following example runs a tail –f outfile command as a background job, uses ps to display the PID number of the process, and aborts the job with kill:

```
$ tail -f outfile&
[1] 18228
$ ps | grep tail
18228 pts/4 00:00:00 tail
$ kill 18228
[1]+ Terminated tail -f outfile
$
```

If you forget a job number, you can use the jobs command to display a list of job numbers. The next example is like the previous one except it uses the job number instead of the PID number to identify the job to be killed. Sometimes the message saying the job is terminated does not appear until you press RETURN after the RETURN that executes the kill command.

```
$ tail -f outfile&
[1] 18236
$ bigjob&
[2] 18237
$ jobs
[1]- Running tail -f outfile&
[2]+ Running bigjob&
$ kill %1
$ RETURN
[1]- Terminated tail -f outfile
$
```

7.7 Filename Generation/Pathname Expansion

When you give the shell abbreviated filenames that contain special characters, also called metacharacters, the shell can generate filenames that match the names of existing files. These special characters are also referred to as wildcards because they act much as the jokers do in a deck of cards. When one of these characters appears in an argument on the command line, the shell expands that argument in sorted order into a list of filenames and passes the list to the program called by the command line. Filenames that contain these special characters are called ambiguous file references because they do not refer to any one specific file. The process that the shell performs on these filenames is called pathname expansion or globing.

Ambiguous file references refer to a group of files with similar names quickly, saving the effort of typing the names individually. They can also help find a file whose name you do not remember in its entirety. If no filename matches the ambiguous file reference, the shell generally passes the unexpanded reference—special characters and all—to the program.

The ? Special Character

The question mark (?) is a special character that causes the shell to generate filenames. It matches any single character in the name of an existing file. The following command uses this special character in an argument to the lpr utility: $ lprmemo?.

The shell expands the memo? argument and generates a list of files in the working directory that have names composed of memo followed by any single character. The shell then passes this list to lpr. The lpr utility never "knows" the shell generated the filenames it was called with. If no filename matches the ambiguous file reference, the shell passes the string itself (memo?) to lpr or, if it is set up to do so, passes a null string. The following example uses ls first to display the names of all files in the working directory and then to display the filenames that memo? matches:

```
$ ls
mem    memo12    memo9   memomax    newmemo5
memo   memo5    memoa    memos

$ ls memo?
memo5    memo9    memoa     memos
```

The memo? ambiguous file reference does not match mem, memo, memo12, memomax, or newmemo5. You can also use a question mark in the middle of an ambiguous file reference:

```
$ ls
7may4report    may4report    mayqreportmay_report
may14report    may4report.79    mayreportmay.report

$ ls may?report
may.report    may4report    may_reportmayqreport
```

The * Special Character

The asterisk (*) performs a function like that of the question mark but matches any number of characters, including zero characters, in a filename.

```
$ ls
amemo    memo.0612    memosally memsam user.memo    mem       memoa       memosam.0620
sallymemo    memo  memorandum  memosam.keep     typescript

$ echo memo*
memo    memo.0612   memoa    memorandum memosally    memosam.0620    memosam.keep

$ echo *mo
amemo    memo   sallymemouser.memo

$ echo *sam*
memosam.0620    memosam.keepmemsam
```

The ambiguous file reference memo* does not match amemo, mem, sallymemo, or user.memo. Like the question mark, an asterisk does not match a leading period in a filename.The –a option causes ls to display hidden filenames. The command echo * does not display . (the working directory), .. (the parent of the working directory), .aaa, or .profile. In contrast, the command echo .* displays only those four names:

```
$ ls
aaamemo.sally    sally.0612    thurs
memo.0612   report   saturday
```

```
$ ls -a
.       .aaaaaamemo.sally    sally.0612      thurs
..      .profile      memo.0612        report       saturday

$ echo *
aaa     memo.0612     memo.sally   report sally.0612 saturdaythurs

$ echo .*
.    ..     .aaa       .profile
```

In the following example, .p* does not match memo.0612, private, reminder, or report. The ls .* command causes ls to list .private and .profile in addition to the contents of the . directory (the working directory) and the .. directory (the parent of the working directory). When called with the same argument, echo displays the names of files (including directories) in the working directory that begin with a dot (.), but not the contents of directories.

```
    $ ls -a
    .      .private    memo.0612     reminder
    ..     .profile     private      report

    $ echo .p*
    .private       .profile

    $ ls .*
    .private       .profile
    .:
    memo.0612     private       reminder       report
    ..:
    .
    .

    $ echo .*
    .    ..     .private       .profile
```

You can plan to take advantage of ambiguous file references when you establish conventions for naming files. For example, when you end all text filenames with .txt, you can reference that group of files with *.txt.The next command uses this convention to send all text files in the working directory to the printer.The ampersand (&) causes lpr to run in the background.

```
        $ lpr *.txt &
```

The [ ] Special Characters

A pair of brackets surrounding a list of characters causes the shell to match filenames containing the individual characters. Whereas memo? matches memo followed by any character, memo[17a] is more restrictive: It matches only memo1, memo7, and memoa. The brackets define a character class that includes all the characters within the brackets. The shell expands an argument that includes a character-class definition, by substituting each member of the character class, one at a time, in place of the brackets and their contents. The shell then passes the list of matching filenames to the program it is calling. Each character-class definition can replace only a single character within a filename. The brackets and their contents are like a question mark that substitutes only the members of the character class. The first of the following commands lists the names of all files in the working directory that begin with a, e, i, o, or u. The second command displays the contents of the files named page2.txt, page4.txt, page6.txt, and page8.txt.

```
$ echo [aeiou]*

...

$ less page[2468].txt

...
```

A hyphen within brackets defines a range of characters within a character-class definition. For example, [6–9] represents [6789], [a–z] represents all lowercase letters in English, and [a–zA–Z] represents all letters, both uppercase and lowercase, in English. The following command lines show three ways to print the files named part0, part1, part2, part3, and part5. Each of these command lines causes the shell to call lpr with five filenames:

```
$ lpr part0 part1 part2 part3 part5

$ lpr part[01235]

$ lpr part[0-35]
```

The following command line prints 39 files, part0 through part38:

```
$ lprpart[0-9] part[12][0-9] part3[0-8]
```

The first of the following commands lists the files in the working directory whose names start with a through m.

```
$ echo [a-m]*

...
```

The second lists files whose names end with x, y, or z.

```
$ echo *[x-z]

...
```

The ls utility cannot interpret ambiguous file references. First ls is called with an argument of ?old. The shell expands ?old into a matching filename, hold, and passes that name to ls.

```
$ ls ?old

hold
```

The second command is the same as the first, except the ? is quoted.

```
$ ls \?old

ls: ?old: No such file or directory
```

The shell does not recognize this question mark as a special character and passes it to ls. The ls utility generates an error message saying that it cannot find a file named ?old (because there is no file named ?old).

7.8 Builtins

A builtin is a utility (also called a command) that is built into a shell. Each of the shells has its own set of builtins. When it runs a builtin, the shell does not fork a new process. Consequently, builtins run more quickly and can affect the environment of the current shell. Because builtins are used in the same way as utilities, you will not typically be aware of whether a utility is built into the shell or is a standalone utility. The echo utility, for example, is a shell builtin. The shell always executes a shell builtin before trying to find a command or utility with the same name.

Listing bash builtins

To display a list of bash builtins, give the command info bash and select the Shell Builtin Commands menu. Then select the Bourne Shell Builtins and/or Bash Builtins menus. The bash info page is part of the bash-doc package—you can view only the man page (even using info) if this package is not installed. Because bash was written by GNU, the info page has better information than does the man page.

Listing tcshbuiltins

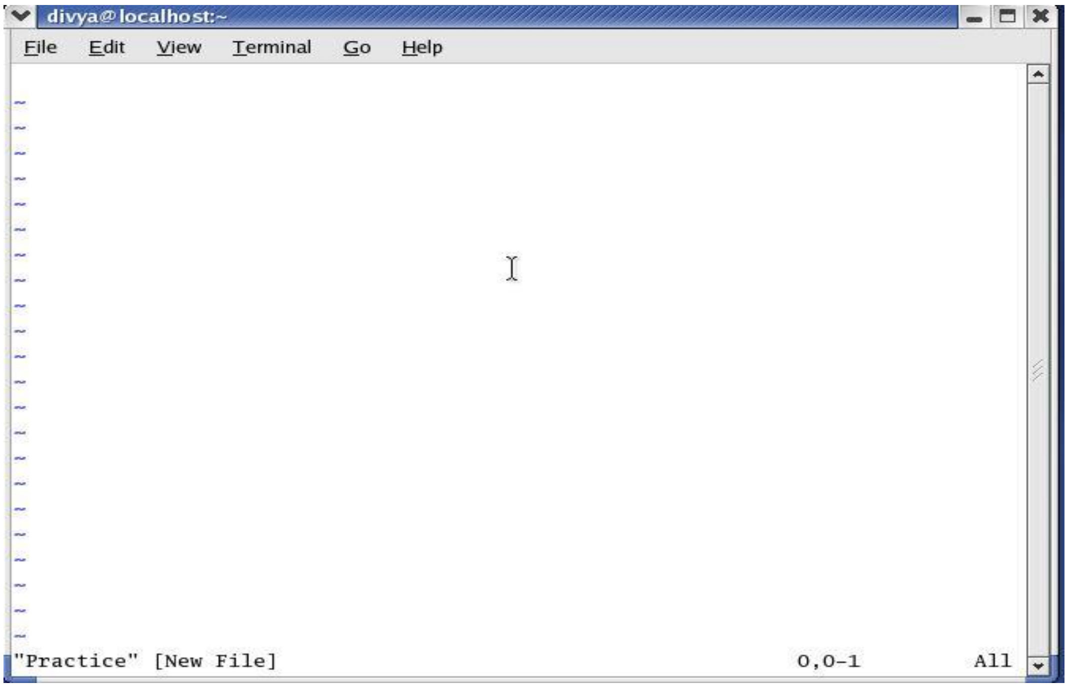
For tcsh, give the command man tcsh to display the tcsh man page and then search with the following command: /Builtin commands$ (search for the string at the end of a line).

7.9 vim

The vim editor is not a text formatting program. It does not justify margins or provide the output formatting features of a sophisticated word processing system such as OpenOffice.org Writer. Vim is a sophisticated text editor meant to be used to write code (C, HTML, Java, etc.), short notes, and input to a text formatting system, such as groff or troff.

Starting vim

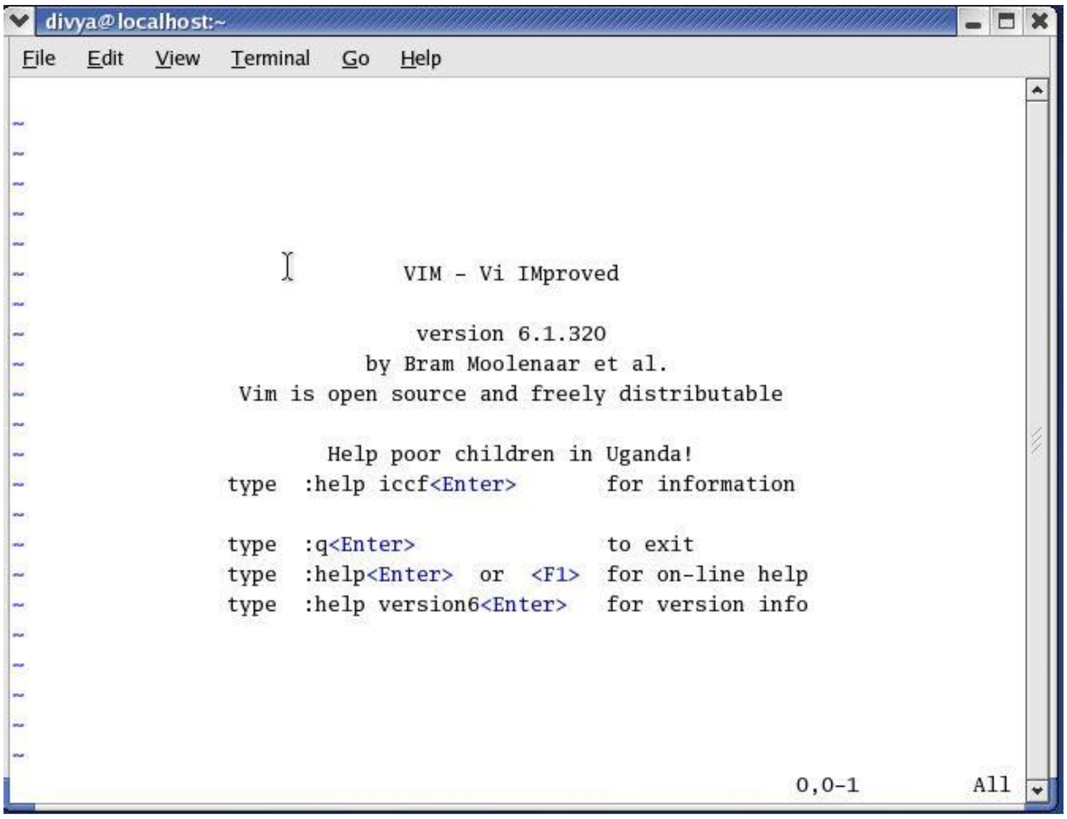
Start vim with the following command to create and edit a file named practice: $ vim practice. The tildes in the starting indicates that the page is blank.



```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"Practice" [New File]                                    0,0-1          All
```

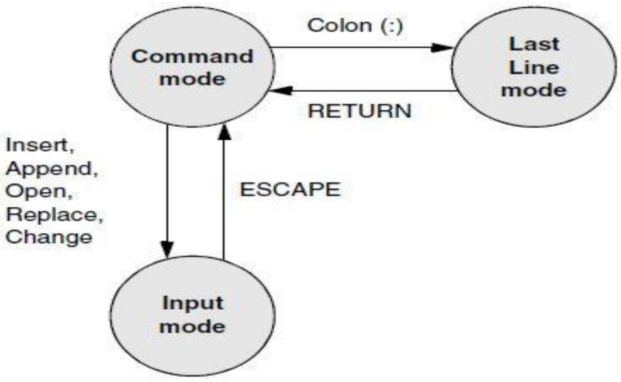
If you call vim without specifying a filename on the command line, vim assumes that you are a novice and tells you how to get started.

```
$ vim
```



```
~
~
~
~
~
~                          VIM - Vi IMproved
~
~                          version 6.1.320
~                      by Bram Moolenaar et al.
~             Vim is open source and freely distributable
~
~                    Help poor children in Uganda!
~            type  :help iccf<Enter>       for information
~
~            type  :q<Enter>               to exit
~            type  :help<Enter>  or  <F1>  for on-line help
~            type  :help version6<Enter>   for version info
~
~
~
~
~
                                                0,0-1         All
```

7.10 Modes in vim



Two of vim's modes of operation are Command mode (also called Normal mode) and Input mode. While vim is in Command mode, you can give vim commands. For example, you can delete text or exit from vim. You can also command vim to enter Input mode. In Input mode, vim accepts anything you enter as text and displays it on the screen.The colon (:) in the preceding command puts vim into another mode, Last Line mode. While in this mode, vim keeps the cursor on the bottom line of the screen. When you press RETURN to finish entering the command, vim restores the cursor to its place in the text. Press ESCAPE to return vim to Command mode. By default, the vim editor informs you about which mode it is in: It displays INSERT at the lower-left corner of the screen while it is in Insert mode.

Entering Text

When you start vim, you must put it in Input mode before you can enter text. To put vim in Input mode, press the i (insert before cursor) key or the a (append after cursor) key. If you are not sure whether vim is in Input mode, press the ESCAPE key; vim returns to Command mode if it is in Input mode or beeps, flashes, or does nothing if it is already in Command mode. You can put vim back in Input mode by pressing the i or a key again. While vim is in Input mode, you can enter text by typing on the keyboard. If the text does not appear on the screen as you type, vim is not in Input mode.

Correcting Text

The keys that back up and correct a shell command line serve the same functions when vim is in Input mode.

Key	Results
CONTROL-H	Erase
CONTROl-U	Line kill
CONTROL-W	Word kill
Dd, dw, x	Remove the incorrect text
I, a, o, O	Insert the correct text

To change the word pressing to hitting in, you might use the ARROW keys to move the cursor until it is on top of the p in pressing. Then give the command dw to delete the word pressing. Put vim in Input mode by giving an i command, enter the word hitting followed by a SPACE, and press ESCAPE. The word is changed, and vim is in Command mode, waiting for another command. A shorthand for the two commands dw followed by the i command is cw (Change word). The cw command puts vim into Input mode.

Ending the Editing Session

While you are editing, vim keeps the edited text in an area named the Work buffer. Use the ZZ command (you must use uppercase Zs) to write the newly entered text to the disk. You can exit with :q! if you do not want to save your work.

Moving the Cursor

To delete, insert, and correct text, you need to move the cursor on the screen. While vim is in Command mode, you can use the RETURN key, the SPACE bar, and the ARROW keys to move the cursor. You can use the h, j, k, and l (lowercase "l") keys to move the cursor left, down, up, and right, respectively.

Deleting Text

If you wany to delete a single character, then press x. If a word needs to be deleted, pressdw and for the deletion of line of text, press dd.

Undoing Mistakes

Give the command u (Undo) immediately after the command you want to undo. If you give the u command again, vim will undo the command you gave before the one it just undid. You can use this technique to back up over many of your actions. If you undo a command, you did not mean to undo, give a Redo command: CONTROL-R or :redo (followed by a RETURN).As with the Undo command, you can give the Redo command many times in a row.

Entering Additional Text

When you want to insert new text within existing text, move the cursor so it is on the character that follows the new text you plan to enter. Then give the i (Insert) command to put vim in Input mode, enter the new text, and press ESCAPE to return vim to Command mode. Alternatively, you can position the cursor on the character that precedes the new text and use the a (Append) command. To enter one or more lines, position the cursor on the line above where you want the new text to go. Give the command o (Open). The vim editor opens a blank line below the line the cursor was on, puts the cursor on the new, empty line, and enters Input mode. Enter the new text, ending each line with a RETURN. When you are finished entering text, press ESCAPE to return vim to Command mode. The O command works in the same way as the o command, except it opens a blank line above the current line.

7.11 Introduction to vim Features

There are various useful features of vim which are of great help:

- Online help,
- Modes of operation,
- The work buffer,
- Emergency procedures,
- Other vim features.

Online Help

Give the command: help feature to display information about feature. As you scroll through the various help texts, you will see words with a bar on either side, such as |tutor|. These words are active links: Move the cursor on top of an active link and press CONTROL-] to jump to the linked text. Use CONTROL-o (lowercase "o") to jump back to where you were in the help text. You can also use the active link words in place of feature. For example, you might see the reference |credits|; you could enter :help credits RETURN to read more about credits. Enter :q! to close a help window. You can also give the command :help doc-file-list to view a complete list of the help files.

Modes of Operation

The current character is the character the cursor is on. The current line is the line the cursor is on. The status line is the last or bottom line of the screen. This line is reserved for Last Line mode and status information. Text you are editing does not appear on this line. The vim editor is part of the ex-editor, which has five modes of operation:

- ex Command mode
- ex Input mode
- vim Command mode
- vim Input mode
- vim Last Line mode

While in Command mode, vim accepts keystrokes as commands, responding to each command as you enter it. It does not display the characters you type in this mode. While in Input mode, vim accepts and displays keystrokes as text that it eventually puts into the file you are editing. All commands that start with a colon (:) put vim in Last Line mode. The colon moves the cursor to the status line of the screen, where you enter the rest of the command. When you give a command in Command mode, you do not terminate the command with a RETURN. In contrast, you must terminate all Last Line mode commands with a RETURN. When an editing session begins, vim is in Command mode. Several commands, including Insert and Append, put vim in Input mode. When you press the ESCAPE key, vim always reverts to Command mode. The Change and Replace commands combine the Command and Input modes. The Change command deletes the text you want to change and puts vim in Input mode so you can insert new text. The Replace command deletes the character(s) you overwrite and inserts the new one(s) you enter.

The vim editor displays status information on the bottom line of the display area. This information includes error messages, information about the deletion or addition of blocks of text, and file status information. In addition, vim displays Last Line mode commands on the status line. Sometimes the screen may become garbled or overwritten. When vim puts characters on the screen, it sometimes leaves @ on a line instead of deleting the line. When output from a program becomes intermixed with the display of the Work buffer, things can get even more confusing. The output does not become part of the Work buffer but affects only the display. If the screen gets overwritten, press ESCAPE to make sure vim is in Command mode, and press CONTROL-L to redraw (refresh) the screen. If the end of the file is displayed on the screen, vim marks lines that would appear past the end of the file with a tilde (~) at the left of the screen. While vim is in Input mode, you can use the erase and line kill keys to back up over text so you can correct it. You can also use CONTROL-W to back up over words.

Work Buffer

The vim editor does all its work in the Work buffer. At the beginning of an editing session, vim reads the file you are editing from the disk into the Work buffer. During the editing session, it makes all changes to this copy of the file but does not change the file on the disk until you write the contents of the Work buffer back to the disk. Normally when you end an editing session, you tell vim to write the contents of the Work buffer, which makes the changes to the text final. If you accidentally end an editing session without writing out the contents of the Work buffer, your work is lost. However, if you unintentionally make some major changes (such as deleting the entire contents of the Work buffer), you can end the editing session without implementing the changes. To look at a file but not to change it while you are working with vim, you can use the view utility: $ view filename. Calling the view utility is the same as calling the vim editor with the –R (readonly) option.

The vim editor operates on files of any format. The total length of the file is limited only by available disk space and memory. The vim editor allows you to open, close, and hide multiple windows, each of which allows you to edit a different file. Give the command :help windows to display a complete list of windows commands.

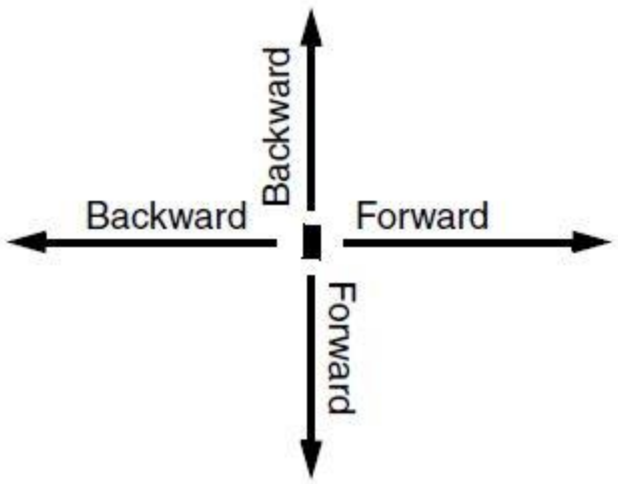
Abnormal Termination of an Editing Session and recovering after a crash (Emergency Procedures)

You can end an editing session in one of two ways: When you exit from vim, you can save the changes you made during the editing session or you can abandon those changes. You can use the ZZ or :wq command from Command mode to save the changes and exit from vim. To end an editing session without writing out the contents of the Work buffer, give the following command:
:q!.Use the :q! command cautiously. When you use this command to end an editing session, vim does not preserve the contents of the Work buffer, so you will lose any work you did since the last time you wrote the Work buffer to disk.:w filename: Use this to save the file with a name.

The vim editor temporarily stores the file you are working on in a swap file. If the system crashes while you are editing a file with vim, you can often recover its text from the swap file. If someone else is editing the file, quit or open the file as a read only file. With the –r option, vim displays a list of swap files it has saved (some may be old).If your work was saved, give the same command followed by a SPACE and the name of the file. Give the command :w filename immediately to save the salvaged copy of the Work buffer to disk under a name different from the original file; then check the recovered file to make sure it is OK.

7.12 Command Mode

While vim is in Command mode, you can position the cursor over any character on the screen.

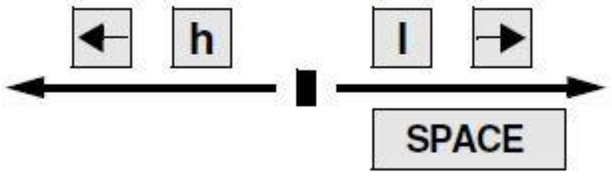


Forward means toward the right and bottom of the screen and the end of the file. Backward means toward the left and top of the screen and the beginning of the file.

Moving the Cursor

When you use a command that moves the cursor forward past the end (right) of a line, the cursor generally moves to the beginning (left) of the next line. When you move it backward past the beginning of a line, the cursor generally moves to the end of the previous line. Sometimes a line in the Work buffer may be too long to appear as a single line on the screen. In such a case vim wraps the current line onto the next line. You can move the cursor through the text by any Unit of Measure (that is, character, word, line, sentence, paragraph, or screen).If you precede a cursor-movement command with a number, called a Repeat Factor, the cursor moves that number of units through the text.

Moving the Cursor by Characters



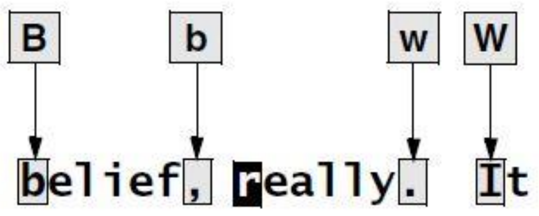
Key	Results
Space bar, l and Right Arrow Key	Forward
h and Left Arrow Key	Backward

For example, the command 7 SPACE or 7l moves the cursor seven characters to the right.

Moving the Cursor to a Specific Character

You can move the cursor to the next occurrence of a specified character on the current line by using the Find command. For example, the following command moves the cursor from its current position to the next occurrence of the character a, if one appears on the same line: fa. You can also find the previous occurrence by using a capital F. The following command moves the cursor to the position of the closest previous a in the current line: Fa. A semicolon (;) repeats the last Find command.

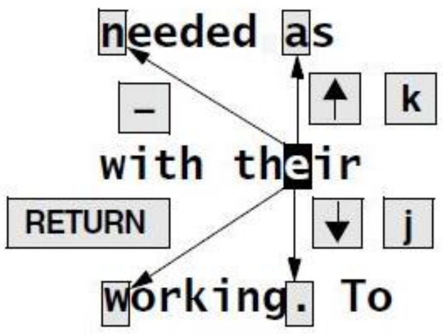
Moving the Cursor by Words



Key	Results
W and w	Moves the cursor forward
B and b	Moves the cursor backwards
E and e	Moves the cursor to the end of the next word

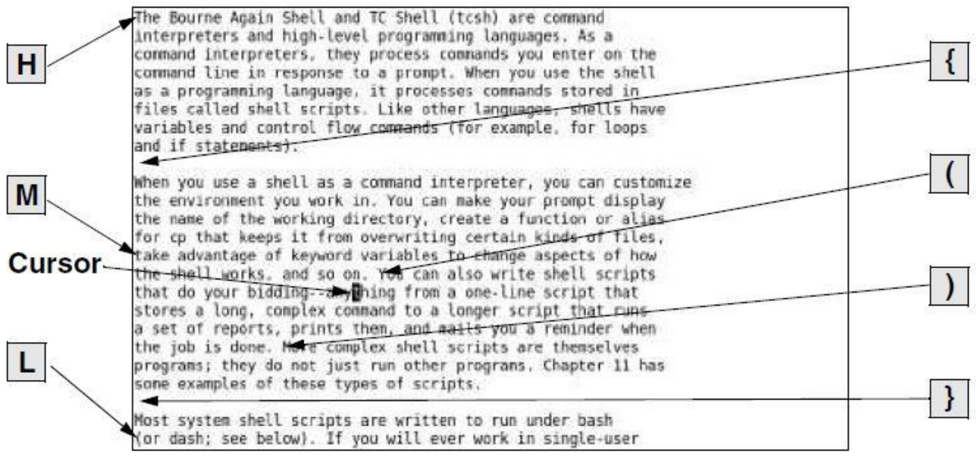
Groups of punctuation count as words. The command 15w moves the cursor to the first character of the fifteenth subsequent word. The W key is like the w key but moves the cursor by blank-delimited words, including punctuation, as it skips forward. The B key moves the cursor backward by blank-delimited words. E moves it to the end of the next blank-delimited word.

Moving the Cursor by Lines



Key	Results
RETURN	To the beginning of next line.
j/ DOWN Arrow	Down one line to the character just below the current character.
k and UP Arrow	Up one line to the character just above the current character.
-	To the end of the next line.
)	Forward to the beginning of the next sentence.
(	Forward to the beginning of the next paragraph.
}	Backward to the beginning of current sentence.
{	Backward to the beginning of current paragraph.

Moving the Cursor Within the Screen



```
The Bourne Again Shell and TC Shell (tcsh) are command
interpreters and high-level programming languages. As a
command interpreters, they process commands you enter on the
command line in response to a prompt. When you use the shell
as a programming language, it processes commands stored in
files called shell scripts. Like other languages, shells have
variables and control flow commands (for example, for loops
and if statements).

When you use a shell as a command interpreter, you can customize
the environment you work in. You can make your prompt display
the name of the working directory, create a function or alias
for cp that keeps it from overwriting certain kinds of files,
take advantage of keyword variables to change aspects of how
the shell works, and so on. You can also write shell scripts
that do your bidding--anything from a one-line script that
stores a long, complex command to a longer script that runs
a set of reports, prints them, and mails you a reminder when
the job is done. More complex shell scripts are themselves
programs; they do not just run other programs. Chapter 11 has
some examples of these types of scripts.

Most system shell scripts are written to run under bash
(or dash; see below). If you will ever work in single-user
```

Key	Result
H	To the left end of the top line of the screen
M	To the middle line
L	To the bottom line

Viewing Different Parts of the Work Buffer

The screen displays a portion of the text that is in the Work buffer. You can display the text preceding or following the text on the screen by scrolling the display. You can also display a portion of the Work buffer based on a line number. Press CONTROL-D to scroll the screen down (forward) through the file so that vim displays half a screen of new text. Use CONTROL-U to scroll the screen up (backward) by the same amount. If you precede either of these commands with a number, vim scrolls that number of lines each time you press CONTROL-D or CONTROL-U for the rest of the session (unless you again change the number of lines to scroll). The CONTROL-F (forward) and CONTROL-B (backward) keys display almost a whole screen of new text, leaving a couple of lines from the previous screen for continuity. On many keyboards you can use the PAGE DOWN and PAGE UP keys in place of CONTROL-F and CONTROL-B, respectively.

Deleting and Changing of Text

- Undoing changes
- Deleting characters
- Deleting text
- Changing text
- Replacing text
- Changing case

Undoing Changes

The u command (Undo) restores text that you just deleted or changed by mistake. A single Undo command restores only the most recently deleted text. With the compatible parameter set, vim can undo only the most recent change. The U command restores the last line you changed to the way it was before you started changing it, even after several changes.

Deleting Characters

The x command deletes the current character. You can precede the x command by a Repeat Factor to delete several characters on the current line, starting with the current character. The X command deletes the character to the left of the cursor.

Deleting Text

The d (Delete) command removes text from the Work buffer. The amount of text that d removes depends on the Repeat Factor and the Unit of Measure. After the text is deleted, vim is still in Command mode.

Command	Result
dl	Deletes current character (same as the x command)
d0	Deletes from the beginning of line
d^	Deletes from first character of line
dw	Deletes to end of word
d3w	Deletes to end of third word
db	Deletes from beginning of word
dw	Deletes to end of blank delimited word
dB	Deletes to beginning of blank delimited word
d7B	Deletes from seventh previous beginning of blank delimited word
d)	Deletes to end of sentence
d4)	Deletes to end of fourth sentence
d{	Deletes from beginning of sentence
d}	Deletes to end of paragraph
d{	Deletes from beginning of paragraph
d7{	Deletes from 7th paragraph preceding from beginning of paragraph
d/text	Deletes upto next occurrence of word text
dfc	Deletes on current line upto and including next occurrence of character c
dtc	Deletes on current line upto next occurrence of c
D	Deletes to end of line
D$	Deletes to end of line

dd	Deletes current line
5dd	Deletes five lines starting with current line
dL	Deletes through last line on screen
dH	Deletes from first line on screen
dG	Deletes through end of work buffer
D1G	Deletes from beginning of work buffer

Changing Text

The c (Change) command replaces existing text with new text. The new text does not have to occupy the same space as the existing text. You can change a word to several words, a line to several lines, or a paragraph to a single character. The C command replaces the text from the cursor position to the end of the line. The c command deletes the amount of text specified by the Repeat Factor and the Unit of Measure and puts vim in Input mode. When you finish entering the new text and press ESCAPE, the old word, line, sentence, or paragraph is changed to the new one. Pressing ESCAPE without entering new text deletes the specified text (that is, it replaces the specified text with nothing).

Command	Result
cl	Changes current character
cw	Changes to end of word
c3w	Changes to end of third word
cb	Changes from beginning of word
cW	Changes from end of blank delimited word
cB	Changes from beginning of blank delimited word
C7B	Changes from beginning of seventh previous blank delimited word
C$	Changes to end of line
c0	Changes from beginning of line
c)	Changes to end of sentence
c4)	Changes to end of fourth sentence
c(	Changes from beginning of sentence
c}	Changes to end of paragraph
c{	Changes from beginning of paragraph
c7{	Changes from beginning of seventh preceding paragraph
ctc	Changes of current line upto next occurrence of c
C	Changes to end of line
cc	Changes current line

Replacing Text

The s and S (Substitute) commands also replace existing text with new text.The s command deletes the current character and puts vim into Input mode. It has the effect of replacing the current character with whatever you type until you press ESCAPE. The S command does the same thing as the cc command: It changes the current line. The s command replaces characters only on the current line.If you specify a Repeat Factor before an s command and this action would replace more characters than are present on the current line, s changes characters only to the end of the line (same as C).

Command	Result
s	Substitute one or more characters for current character
S	Substitute one or more characters for current line
5s	Substitute one or more characters for five characters, starting with current character

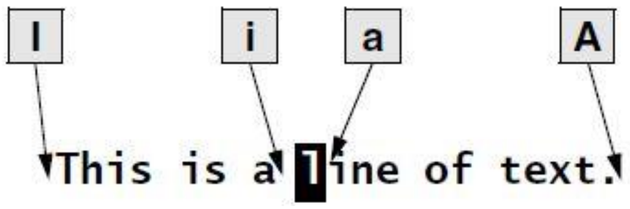
Changing Case

The tilde (~) character changes the case of the current character from uppercase to lowercase, or vice versa. You can precede the tilde with a number to specify the number of characters you want the command to affect.For example, the command 5~ transposes the next five characters starting with the character under the cursor but will not transpose characters past the end of the current line.

7.13 Input Mode

The Insert, Append, Open, Change, and Replace commands put vim in Input mode.While vim is in this mode, you can put new text into the Work buffer. To return vim to Command mode when you finish entering text, press the ESCAPE key.

Inserting Text



Key	Results
i	Put vim in the insert mode and places the text before the current character.
I	Places the text at the beginning of the current line.
a	Places the text after the current character.
A	Places the text after the current line.
o	Opens a new line below the current line.
O	Opens a new line above the current line.

Replacing Text

The r and R (Replace) commands cause the new text you enter to overwrite (replace) existing text. The single character you enter following an r command overwrites the current character. After you enter that character, vim returns to Command mode—you do not need to press the ESCAPE key. The R command causes all subsequent characters to overwrite existing text until you press ESCAPE to return vim to Command mode.

Quoting Special Characters in Input Mode

While you are in Input mode, you can use the Quote command, CONTROL-V, to enter any character into the text, including characters that normally have special meaning to vim. Among these characters are CONTROL-L (or CONTROL-R), which redraws the screen; CONTROL-W, which backs the cursor up a word to the left; CONTROL-M, which enters a NEWLINE; and ESCAPE, which ends Input mode. To insert one of these characters into the text, type CONTROL-V followed by the character. CONTROL-V quotes the single character that follows it.

7.14 Emacs

Emacs is one of the oldest and most versatile text editors available for Linux and UNIX-based systems. It's been around for a long time (more than twenty years for GNU emacs) and is well known for its powerful and rich editing features. The emacs editor, which is coded in C, contains a complete Lisp interpreter and fully supports the X Window System and mouse interaction. Version 22 has significant internationalization upgrades: an extended UTF-8 internal character set four times bigger than Unicode, along with fonts and keyboard input methods for more than 30 languages. Also, the user interface is moving in the direction of a WYSIWYG (what you see is what you get) word processor, which makes it easier for beginners to use the editor. You never need to switch emacs between Input and Command modes, emacs is a modeless editor.

emacs Versus vim

1) Like vim, emacs is a display editor: It displays on the screen the text you are editing and changes the display as you type each command or insert new text.

2) Unlike vim, emacs does not require you to keep track of whether you are in Command mode or Insert mode: Commands always use CONTROL or other special keys.

3) The emacs editor inserts ordinary characters into the text you are editing (as opposed to using ordinary characters as commands), another trait of modeless editing. For many people this approach is convenient and natural.

4) As with vim, you use emacs to edit a file in a work area, or buffer, and have the option of writing this buffer back to the file on the disk when you are finished. With emacs, however, you can have many work buffers and switch among them without having to write the buffer out and read it back in.

5) Like vim, emacs has a rich, extensive command set for moving about in the buffer and altering text. This command set is not "cast in concrete"—you can change or customize commands at any time.

6) Finally, and very unlike vim, emacs allows you to use Lisp to write new commands or override old ones. This feature is called online extensibility,

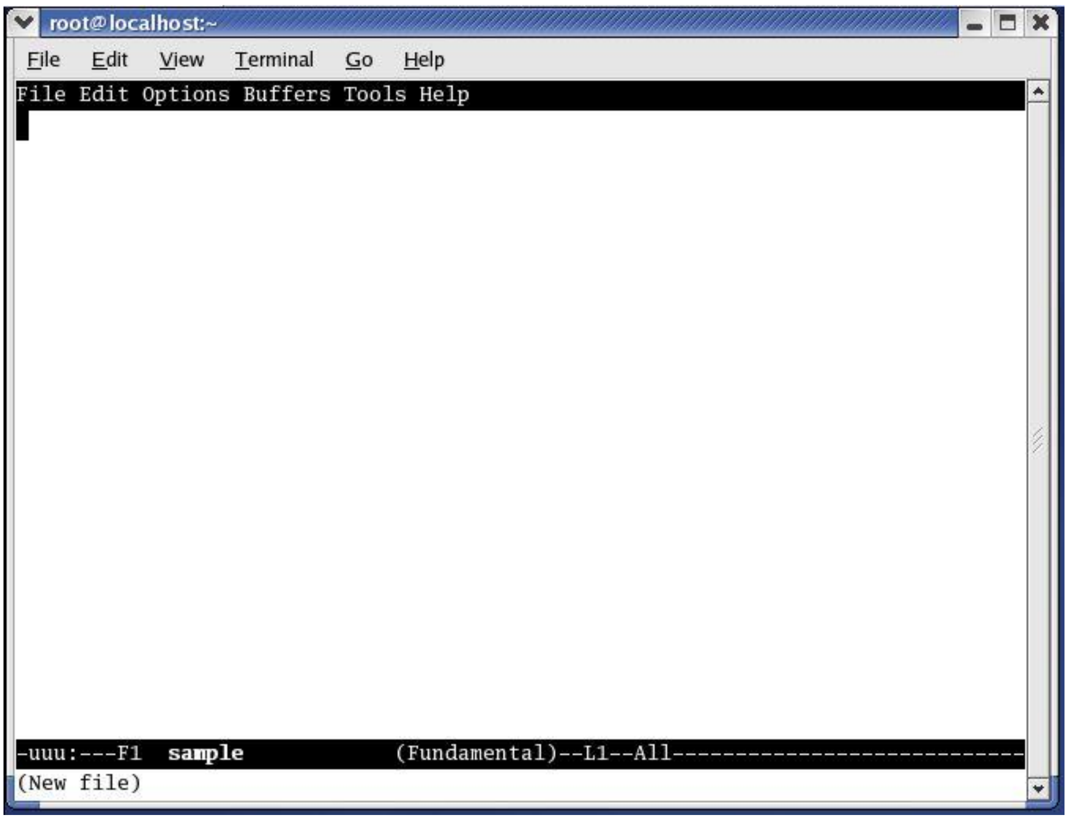
Getting Started with emacs

To edit a file named sample using emacs as a text-based editor, enter the following command:$ emacs -nw -q sample.

- **–nw option:** It must be the first option on the emacs command line, tells emacs not to use its X interface (GUI).

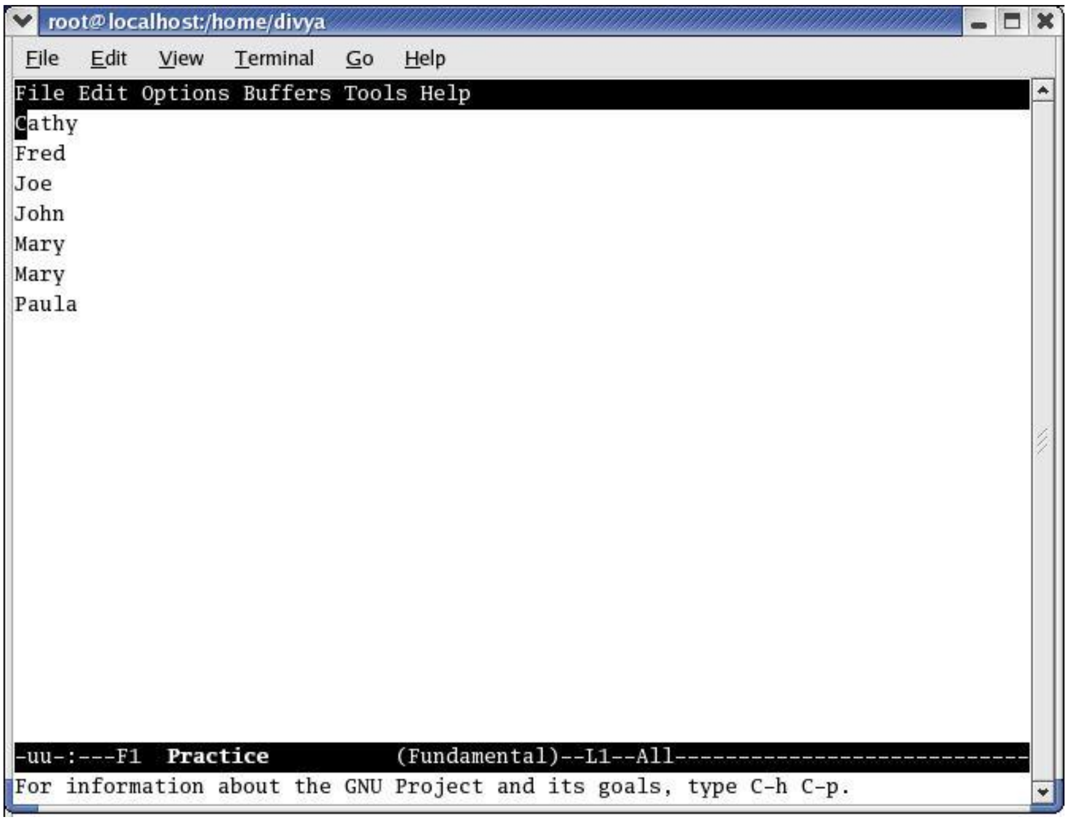
- **–q option: It** tells emacs not to read the ~/.emacs startup file. Not reading this file guarantees that emacs will behave in a standard manner and can be useful for beginners or for other users who want to bypass a .emacs file.

The command $ emacs -nw -q samplestarts emacs, reads the file named sample into a buffer, and displays its contents on the screen or window. If no file has this name, emacs displays a blank screen with (New File) at the bottom.


root@localhost:~
File Edit View Terminal Go Help
File Edit Options Buffers Tools Help

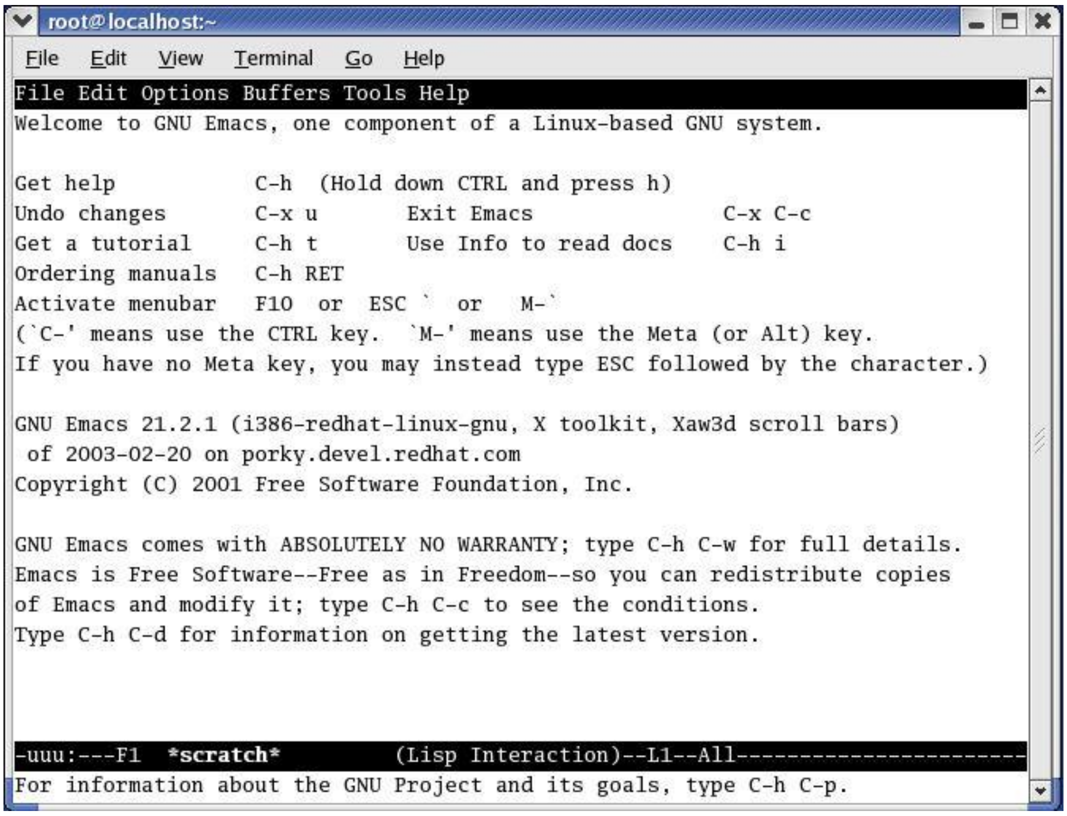
```
-uuu:---F1  sample          (Fundamental)--L1--All---------------------------
(New file)
```

If the file exists, emacs displays the file and a different message.

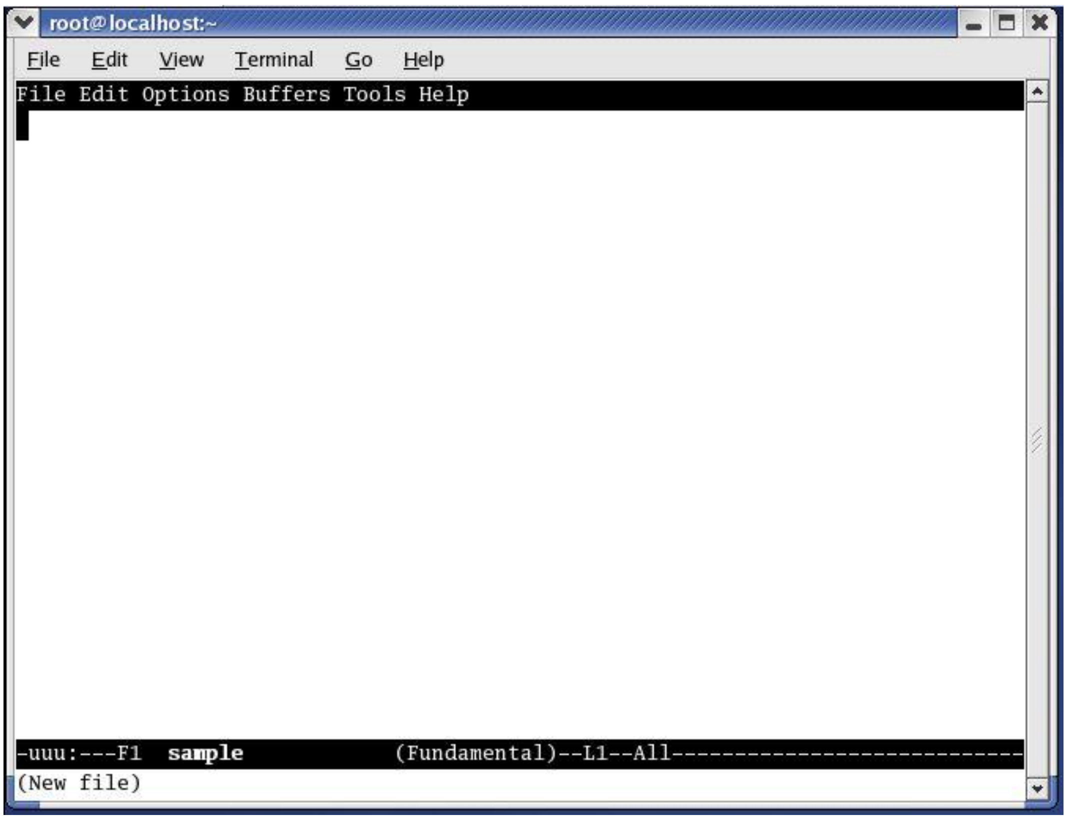

root@localhost:/home/divya
File Edit View Terminal Go Help
File Edit Options Buffers Tools Help

```
Cathy
Fred
Joe
John
Mary
Mary
Paula

-uu-:---F1  Practice        (Fundamental)--L1--All---------------------------
For information about the GNU Project and its goals, type C-h C-p.
```

If you start emacs without naming a file on the command line, it displays a welcome screen that includes usage information and a list of basic commands.



```
File Edit Options Buffers Tools Help
Welcome to GNU Emacs, one component of a Linux-based GNU system.

Get help           C-h  (Hold down CTRL and press h)
Undo changes       C-x u       Exit Emacs              C-x C-c
Get a tutorial     C-h t       Use Info to read docs   C-h i
Ordering manuals   C-h RET
Activate menubar   F10  or  ESC `  or   M-`
(`C-' means use the CTRL key.  `M-' means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the character.)

GNU Emacs 21.2.1 (i386-redhat-linux-gnu, X toolkit, Xaw3d scroll bars)
 of 2003-02-20 on porky.devel.redhat.com
Copyright (C) 2001 Free Software Foundation, Inc.

GNU Emacs comes with ABSOLUTELY NO WARRANTY; type C-h C-w for full details.
Emacs is Free Software--Free as in Freedom--so you can redistribute copies
of Emacs and modify it; type C-h C-c to see the conditions.
Type C-h C-d for information on getting the latest version.



-uuu:---F1  *scratch*         (Lisp Interaction)--L1--All----------------------
For information about the GNU Project and its goals, type C-h C-p.
```



```
File Edit Options Buffers Tools Help


-uuu:---F1  sample            (Fundamental)--L1--All----------------------------
(New file)
```

Initially, emacs displays a single window. At the top of the window is a reverse-video menubar that you can access using a mouse or keyboard.From the keyboard, F10, META-' (back tick), or META-x tmm-menubar RETURN displays the Menubar Completion List window. At the bottom of the emacs window is a reverse-video titlebar called the Mode Line. At a minimum, the Mode Line shows which buffer the window is viewing, whether the buffer has been changed, which major and minor modes are in effect, and how far down the buffer the window is positioned.When multiple windows appear on the screen, one Mode Line appears in each window. At the bottom of the screen, emacs leaves a single line open. This Echo Area and Minibuffer line (they coexist on one line) is used for messages and special one-line commands.If you make an error while you are typing in the Minibuffer, emacs displays the error message in the Echo Area. The error message overwrites the command you were typing, but emacs restores the command in a few seconds.A cursor is either in the window or in the Minibuffer. All input and nearly all editing take place at the cursor. As you type ordinary characters, emacs inserts them at the cursor position. If characters are under the cursor or to its right, they are pushed to the right as you type, so no characters are lost.

Exiting

The command to exit from emacs is CONTROL-X CONTROL-C.If you want to cancel a half-typed command or stop a running command before it is done, press CONTROL-G. The emacs editor displays Quit in the Echo Area and waits for another command.

Inserting Text

Typing an ordinary (printing) character pushes the cursor and any characters to the right of the cursor one position to the right and inserts the new character in the space opened by moving the characters.

Deleting Characters

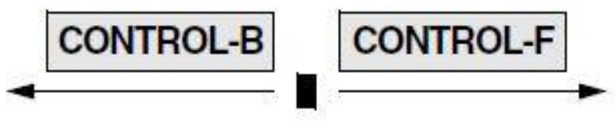
Depending on the keyboard and the emacs startup file, different keys may delete characters in different ways.

- **CONTROL-D** typically deletes the character under the cursor, as do DELETE and DEL.
- **BACKSPACE** typically deletes the character to the left of the cursor.

Moving the Cursor

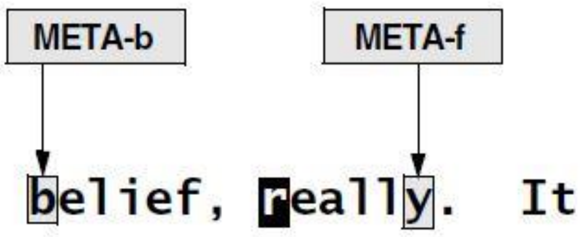
You can position the cursor over any character in the emacs window and move the window, so it displays any portion of the buffer. You can move the cursor forward or backward through the text various textual units—for example, characters, words, sentences, lines, and paragraphs.

Moving the Cursor by Characters



Key	Results
Control – B / Left Arrow Key	Moves the cursor backward one character.
Control – F / Right Arrow Key	Moves the cursor forward one character.

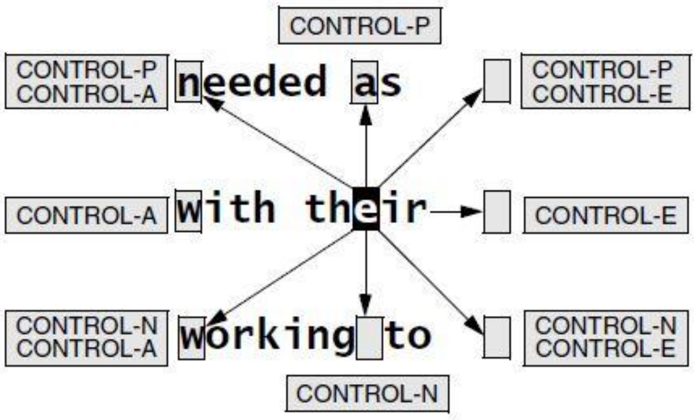
Moving the Cursor by Words



META-f: Moves the cursor forward one word. To invoke this command, holddown the META or ALT key while you press f. If the keyboard you are using does not have either of these keys, press ESCAPE, release it, and then press **f.**

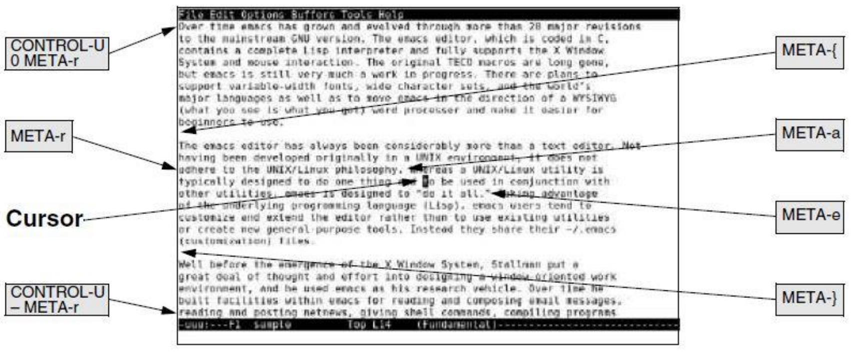
META-b: Moves the cursor backward one word, leaving the cursor on the first letter of the word it started on.

Moving the Cursor by Lines



Key	Result
CONTROL-A/CONTROL-P	Moves the cursor to the beginning of the line it is on
CONTROL-E/CONTROL-P	Moves it to the end
UP ARROW key or CONTROL-P	Moves the cursor up one line to the position directly above where the cursor started
DOWN ARROW key or CONTROL-N	Moves the cursor down one line to the position directly above where the cursor started

Moving the Cursor by Sentences, Paragraphs, and Window Position



```
Over time emacs has grown and evolved through more than 20 major revisions
to the mainstream GNU version. The emacs editor, which is coded in C,
contains a complete Lisp interpreter and fully supports the X Window
System and mouse interaction. The original TECO macros are long gone,
but emacs is still very much a work in progress. There are plans to
support variable-width fonts, wide character sets, and the world's
major languages as well as to move emacs in the direction of a WYSIWYG
(what you see is what you get) word processor and make it easier for
beginners to use.

The emacs editor has always been considerably more than a text editor. Not
having been developed originally in a UNIX environment, it does not
adhere to the UNIX/Linux philosophy. Whereas a UNIX/Linux utility is
typically designed to do one thing and to be used in conjunction with
other utilities, emacs is designed to "do it all." Taking advantage
of the underlying programming language (Lisp), emacs users tend to
customize and extend the editor rather than to use existing utilities
or create new general-purpose tools. Instead they share their ~/.emacs
(customization) files.

Well before the emergence of the X Window System, Stallman put a
great deal of thought and effort into designing a window-oriented work
environment, and he used emacs as his research vehicle. Over time he
built facilities within emacs for reading and composing email messages,
reading and posting netnews, giving shell commands, compiling programs
-uuu:---F1  sample          Top L14    (Fundamental)----------------------
```

Key	Results
META-a	It moves the cursor to the beginning of the sentence the cursor is on
META-e	It moves the cursor to the end of the sentence the cursor is on.
META-{	It moves the cursor to the beginning of the paragraph the cursor is on.
META-}	It moves it to the end of the paragraph the cursor is on.
META-r	It moves the cursor to the beginning of the middle line of the window.
CONTROL-U META-r	It moves the cursor to the beginning of the top line (line zero) in the window.
CONTROL-U– (minus sign)	The command moves the cursor to the beginning of the last line of the window

Editing at the Cursor Position

Entering text requires no commands once you position the cursor in the window at the location you want to enter text. When you type text, emacs displays that text at the position of the cursor. Any text under or to the right of the cursor is pushed to the right.Pressing BACKSPACE removes characters to the left of the cursor. The cursor and the remainder of the text on this line both move to the left each time you press BACKSPACE. To join a line with the line above it, position the cursor on the first character of the second line and press BACKSPACE. Press CONTROL-D to delete the character under the cursor.

Saving and Retrieving the Buffer

No matter what changes you make to a buffer during an emacs session, the associated file does not change until you save the buffer. If you leave emacs without saving the buffer (emacs allows you to do so if you are persistent), the file is not changed and emacs discards the work you did during the session. The command CONTROL-X CONTROL-S saves the current buffer in its associated file. The emacs editor confirms a successful save by displaying an appropriate message in the Echo Area.

The emacs GUI

Full mouse support was introduced in version 19. The emacs editor is still evolving toward full internationalization, accessibility, and a simplified user experience that appeals to the widest possible audience. New features include a colorful menubar and toolbar, mixed text and graphics, tooltips, drag and drop editing, and new GUIs for browsing directories and selecting fonts.

Basic Editing Commands

Command functions in emacs usually involve two or three keys. The most common is the Ctrl key, followed by the Alt or Esc key. In emacs literature, Ctrl is shown in short form as “C”.So if you see something like C-x C-c, it means “press the Ctrl key and x together, then press Ctrl and c”.Similarly, if you see C-h t, it means “press Ctrl and h together, then release both keys and press t”.Alt and Esc keys are referred to as “meta” key in emacs lingo. So if you see a notation like M-x, it means “press Alt/Esc/Option/Edit key and x together”. The Enter key is shown as RET (short for “Return”) and The Esc key is often shown as E.

Marking Text Regions

To mark a text region (like selecting text in popular word processors), follow these steps:

1) Move the cursor to the position where you would like the selection to start Press **C-Space** (Ctrl + Space Bar) or **C-@** to set a mark. The mini buffer will show a status message of Mark set.

2) Move the cursor to the position where you want the region to end.

3) The text will be highlighted up to the point where your cursor is now located.

4) If you want to "un-mark" the highlighted text, press C-Space or C-@ twice The mini buffer will show a status message of Mark deactivated.

Cut, Copy & Paste

Once you have the text region marked, you can copy or cut the text and paste it elsewhere. For copying the text, press **E-w.** For cutting the text, press **C-w.** Move your cursor to the position where the text needs to be pasted. Press **C-y** (y stands for "yank" - you are yanking the text from one position to another). The contents will be pasted here.

Deleting Text

For deleting, Backspace and Delete keys work just the way you would expect them to work. For deleting a whole word, move the cursor at the beginning of a word and press **M-d**. For deleting multiple words, hold the meta key down and keep pressing d. Words will start deleting one by one. For deleting a whole line, position the cursor where you want it to be and press **C-k**. This would delete the text right up to the end of the line on screen. For deleting a sentence, press **M-k**

Undo & Redo

Undoing the last operation is simple. Press **C-x u**. You can keep repeating this to go backwards. Another key combination is **C-/** (Ctrl + /) or **C-_** (Ctrl + _). For redoing your last undo, press **C-g**, followed by **C-_** (that's Ctrl + Shift+ Underscore). Another way to do the same thing would be to press C-x C-u again (Undoing the Undo).

Search & Replace

There are two search directions in emacs: forward and backward. In forward search, the word you specify will be searched forward from the current cursor position. For backward search, it's the other way round. Press **C-s** for forward search and press **C-r** for backward search.

For replacing text, follow these steps:

1. Press **M-%** (that's Alt + Shift + %). The mini buffer shows the prompt for the text to be searched (Query replace:).
2. Type the text and press Enter. The mini buffer displays a message like (Query replace <search_word> with:). Type the replacing text and press Enter.
3. For each match emacs finds, it will ask whether you would like to make a replacement (Query replacing <search_word> with <replace_word>: (C-h for help)). You can take any of the following actions:Press y to replace the current match found/ press n to skip to the next match/ press q to exit without any replacements (basically escaping)/ press ! to do a global replace without any prompts. (emacs will show a message like replaced n occurrences)

Left, Right and Centre Alignment

For justifying a selected text region, follow these steps:

1. Create a text region to highlight the text you wish to justify
2. Press **M-x**. The mini buffer will await a response
3. Start typing **set-justifiction-** and press Tab.
4. You will be given completion options likeset-justification-center, set-justification-left, set-justification-right,set-justification-none and set-justification-full

5. Complete the justification command (for example set-justification-right) and press Enter.
6. The selected text will be justified.

Converting Case

Command	Result
M-c (c for Capitalize)	Capitalizing a word after the cursor
M-l (l for Lower case)	Converting a word to lower case
M-u (u for Upper case)	Converting a word to upper case
Block select, then C-x C-u	Converting a paragraph to upper case
Block select, then C-x C-l	Converting a paragraph to lower case

Summary

- Kernel is the innermost part of an operating system; a shell is the outermost one.
- The line that contains the command, including any arguments, is called the command line.
- By default, the shell directs standard output from a command to the screen.
- In addition to standard input and standard output, a running program normally has a place to send error messages: standard error.
- The redirect output symbol (>) instructs the shell to redirect the output of a command to the specified file instead of to the screen.
- The redirect input symbol (<) instructs the shell to redirect a command's input to come from the specified file instead of from the keyboard.
- A filter is a command that processes an input stream of data to produce an output stream of data.
- When you run a command in the foreground, the shell waits for it to finish before displaying another prompt and allowing you to continue. When you run a command in the background, you do not have to wait for the command to finish before running another command.
- You can suspend a foreground job (stop it from running without aborting the job) by pressing the suspend key, usually CONTROL-Z.
- The vim editor is not a text formatting program. It is a sophisticated text editor meant to be used to write code (C, HTML, Java, etc.), short notes, and input to a text formatting system, such as groff or troff.
- If you call vim without specifying a filename on the command line, vim assumes that you are a novice and tells you how to get started.
- There are three modes in vim: command mode, input mode and last line mode.
- To put vim in Input mode, press the i (insert before cursor) key or the a (append after cursor) key.
- In vim to correct text, use dd, dw, or x to remove the incorrect text. Then use i, a, o, or O to insert the correct text.
- There are various features of vim: online help, modes of operation, work buffer, emergency procedures, etc.

- The emacs editor, which is coded in C, contains a complete Lisp interpreter, and fully supports the X Window System and mouse interaction. You never need to switch emacs between Input and Command modes, emacs is a modeless editor.

Keywords

- **Kernel:**Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible.
- **Shell**: A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output.
- **Token:**On the command line each sequence of nonblank characters is called a token or word.
- **Standard output:** It is a place a program can send information, such as text.
- **Standard input:** It is a place that a program gets information from.
- **Redirection:** The term redirection encompasses the various ways you can cause the shell to alter where standard input of a command comes from and where standard output goes to.
- **noclobber:**The shell provides the noclobber feature that prevents overwriting a file using redirection.
- **Bit bucket:**The /dev/null device is a data sink, commonly referred to as a bit bucket.
- **Pipe:** The shell uses a pipe to connect standard output of one command to standard input of another command.
- **tee utility:**The tee utility copies its standard input both to a file and to standard output.
- **Job:** A job is a series of one or more commands that can be connected by pipes. You can have only one foreground job in a window or on a screen, but you can have many background jobs.
- **? special character:**The question mark (?) is a special character that causes the shell to generate filenames.It matches any single character in the name of an existing file.
- *** special character:**The asterisk (*) performs a function similar to that of the question mark but matches any number of characters, including zero characters, in a filename.
- **Builtin:**A builtin is a utility (also called a command) that is built into a shell.

Self Assessment

1. Which of these is used for erasing a word?

A. CTRL-H
B. CTRL-W
C. CTRL-U
D. None of the above

2. Standard _______ is a place that a program gets information from.

A. Output
B. Input
C. Error
D. None of the above

3. Which symbol is used for appending the standard output to a file?

A. >>

B. <<

C. !!

D. <>

4. Which of these components of operating system is the outermost part?

A. Kernel

B. Shell

C. CPU

D. None of the above

5. The redirection input symbol is

A. >

B. <

C. !

D. None of the above

6. What is another name of pathname expansion?

A. Local-ling

B. Globing

C. Met-forcing

D. None of the above

7. A _____________ within a bracket defines a range.

A. Hyphen

B. Underscore

C. Asterisk

D. None of the above

8. How many jobs we can run in foreground?

A. 0

B. 1

C. 2

D. 3

9. Which of these is a suspend key which can suspend a foreground job?

A. CTRL-Z

B. CTRL-D

C. CTRL-C

D. CTRL-A

10. Which of these symbols puts the vim in last line mode?

A. :

B. ;

C. "
D. ?

11. Which of these modes are available in vim?
A. Command mode
B. Input mode
C. Last line mode
D. All of the above mentioned

12. Which of these characters delete a character?
A. x
B. y
C. z
D. None of the above

13. Which of these keys can be used to move the cursor backward by one character?
A. h
B. i
C. j
D. k

14. For backward search, press
A. C-s
B. C-b
C. C-r
D. C-c

15. The shortcut for exiting from emacs is
A. CTRL-X CTRL-C
B. CTRL-X CTRL-X
C. CTRL-C CTRL-C
D. CTRL-C CTRL-X

Answers for Self Assessment

1.	B	2.	B	3.	A	4.	B	5.	B
6.	B	7.	A	8.	B	9.	A	10.	A
11.	D	12.	A	13.	A	14.	C	15.	A

Review Questions:

1. What is an operating system? Describe its main components: shell and kernel in detail.

2. What is a command? What are the rules for writing a command? Explain the components of a command.
3. What are standard input and standard output? Explain redirection and noclobber.
4. Explain pipes and filters.
5. How can we run a command in background? Explain with commands. How to move a job from foreground to background?
6. What is filename generation? Which special characters are used in this? Explain the difference of usage of each character.
7. What is a builtin? How can we list out bash and tschbuiltins?
8. What is vim? How can we use vim? Explain the starting, entering of text, moving of cursor, correction of text and exiting in vim with examples and commands.
9. What are modes in vim? Explain its usage.
10. Explain the various features of vim in detail.
11. What is command mode? How can we move the cursor in this?
12. How can we delete and change text in command mode? Explain with commands.
13. What is emacs? Explain its difference with vim.
14. What are basic editing commands in emacs? Explain.

Further Readings

https://www.informit.com/articles/article.aspx?p=2854374&seqNum=5

https://unix.stackexchange.com/questions/986/what-are-the-pros-and-cons-of-vim-and-emacs

https://stackoverflow.com/questions/1430164/differences-between-emacs-and-vim

Unit 08: The Bourne Again Shell and Tc Shell

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand the basics of shell, parameters and variables of a shell
- Understand the special characters
- Understand about the process
- Understand the re-execution and editing of commands
- Understand the aliases and functions in Linux
- Know how to control bash, how to enter and leave the TC shell
- Understand the features common to Bourne Again and TC Shells

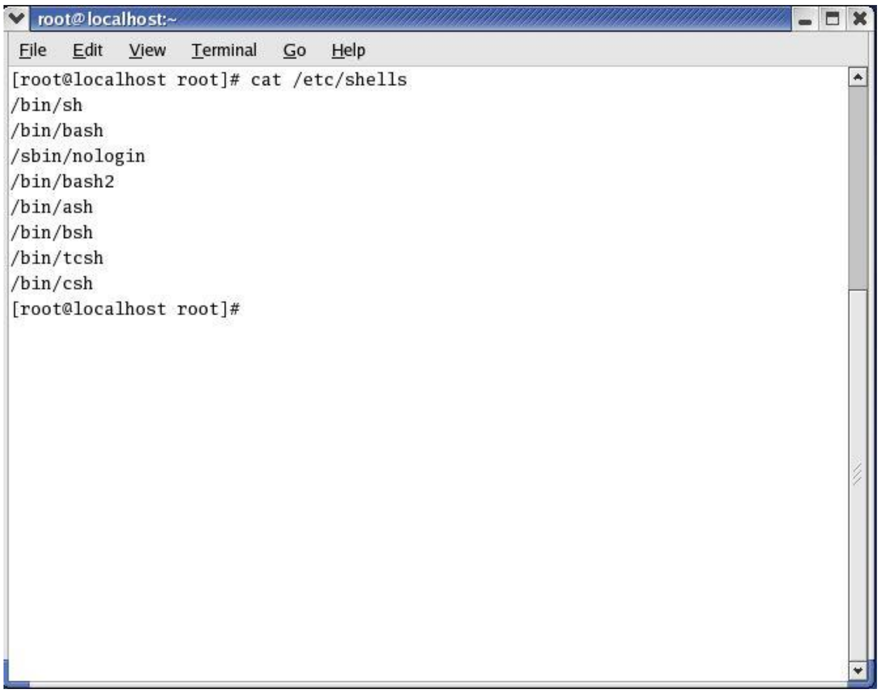
Introduction

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finish executing, it displays that program's output. Shell is an environment in which we can run our commands, programs, and shell scripts.

There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions. There are two major types of shells −

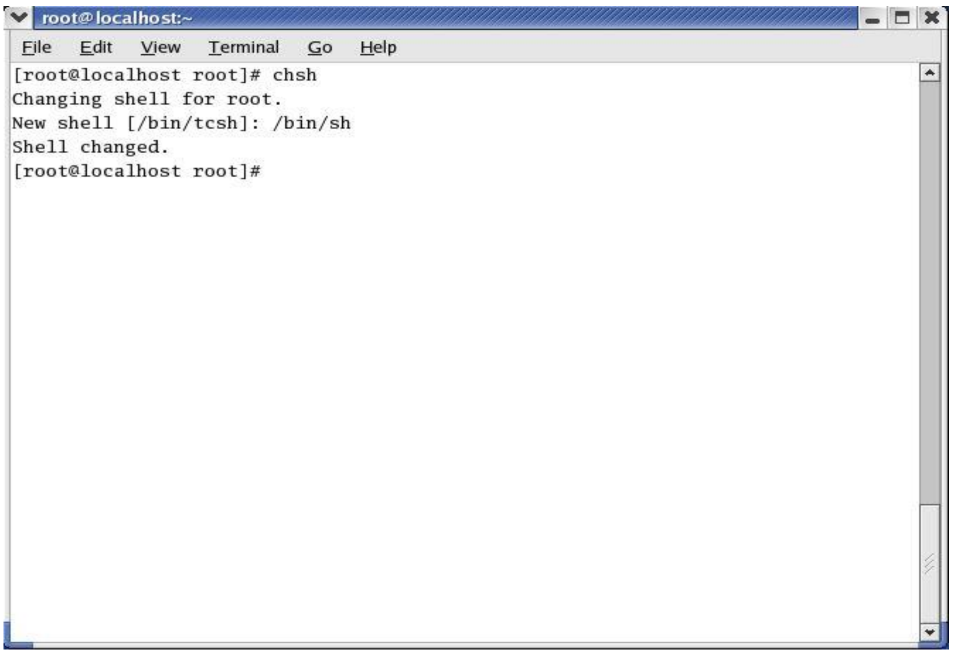
- Bourne Again shell
- TC shell

The Bourne Again Shell and TC Shell are command interpreters and high-level programming languages. As command interpreters, they process commands you enter on the command line in response to a prompt. When you use the shell as a programming language, it processes commands stored in files called shell scripts.Give the command **echo "$SHELL"**. Under most Linux distributions, bash is the default shell. You can run any shell you like once you are logged in. When the command **cat /etc/shells** is given, it shows which shells are available to use in the system.

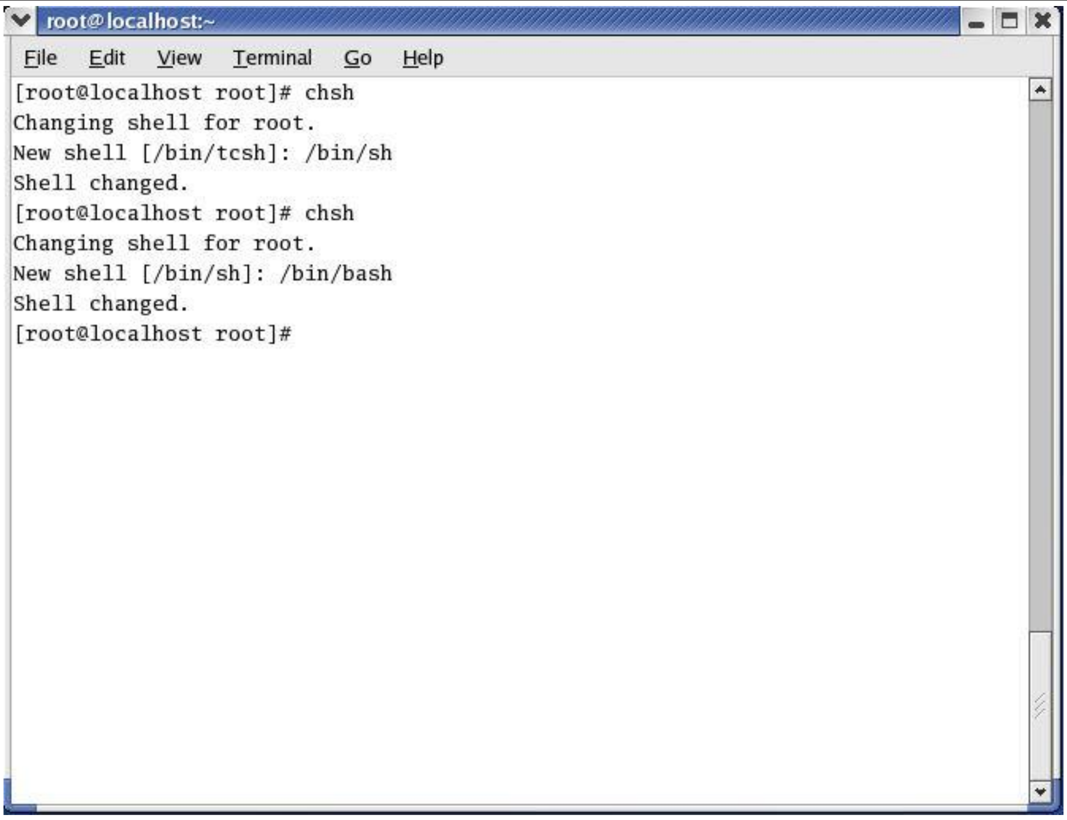


```
[root@localhost root]# cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/bash2
/bin/ash
/bin/bsh
/bin/tcsh
/bin/csh
[root@localhost root]#
```

Enter the name of the shell you want to use (bash, tcsh, or another shell) and press RETURN; the next prompt will be that of the new shell. Give an exit command to return to the previous shell.



```
[root@localhost root]# chsh
Changing shell for root.
New shell [/bin/tcsh]: /bin/sh
Shell changed.
[root@localhost root]#
```



```
[root@localhost root]# chsh
Changing shell for root.
New shell [/bin/tcsh]: /bin/sh
Shell changed.
[root@localhost root]# chsh
Changing shell for root.
New shell [/bin/sh]: /bin/bash
Shell changed.
[root@localhost root]#
```

8.1 Startup Files

When a shell starts, it runs startup files to initialize itself. Which files the shell runs depends on whether it is a login shell, an interactive shell that is not a login shell, or a noninteractive shell. You must have read access to a startup file to execute the commands in it.

Login Shells

The shells that you start with the bash --login option. Login shells are, by their nature, interactive.

1) /etc/profile: The shell first executes the commands in /etc/profile. A user working with root privileges can set up this file to establish system-wide default characteristics for users running bash.

2) .bash_profile, .bash_login, .profile: Next the shell looks for ~/.bash_profile, ~/.bash_login, and ~/.profile (~/ is shorthand for your home directory), in that order, executing the commands in the first of these files it finds. You can put commands in one of these files to override the defaults set in /etc/profile.

3) .bash_logout: When you log out, bash executes commands in the ~/.bash_logout file. This file often holds commands that clean up after a session, such as those that remove temporary files.

Interactive Nonlogin Shells

The commands in these startup files are not executed by interactive, nonlogin shells. However, these shells inherit values from the login shell variables that are set by these startup files.

1) /etc/bashrc: Although not called by bash directly, many ~/.bashrc files call /etc/bashrc. This setup allows a user working with root privileges to establish systemwide default characteristics for nonlogin bash shells.

2) **.bashrc:** An interactive nonlogin shell executes commands in the ~/.bashrc file. Typically, a startup file for a login shell, such as .bash_profile, runs this file, so both login and nonlogin shells run the commands in .bashrc.

However, the noninteractive shells inherit login shell variables that are set by these startup files. Noninteractive shells look for the environment variable BASH_ENV (or ENV if the shell is called as sh) and execute commands in the file named by this variable.

There are various commands that are used as symbols. The Bourne Again Shell uses these symbols as shown in the below table. A command can send error messages to standard error to keep them from getting mixed up with the information it sends to standard output.

Symbol	**Command**
()	Subshell
$( )	Command Substitution
(( ))	Arithmetic evaluation
$(( ))	Arithmetic expression
[ ]	The test command
[[ ]]	Conditional expression; similar to [ ] but adds string comparisons

8.2 File Descriptors

A file descriptor is the place a program sends its output to and gets its input from. When you execute a program, Linux opens three file descriptors for the program:

- 0 (standard input), (0<)
- 1 (standard output), (1>)
- and 2 (standard error), (2>)

Opening a File Descriptor

The Bourne Again Shell opens files using the exec built in as follows:

- exec n>outfile: The line opens outfile for output and holds it open, associating it with file descriptor n.
- exec m<infile: The line opens infile for input and holds it open, associating it with file descriptor m.

Duplicating a file descriptor

The <& token duplicates an input file descriptor; >& duplicates an output file descriptor.

Redirection Operators

There are various redirection operators which can be used. These are shown in the table below.

Operator	Meaning
< filename	Redirects standard input from filename
>filename	Redirects standard output to filename unless filename exists and noclobber is set. If noclobber is not set, this redirection creates filename if it does not exist and overwrites it if it does exist.
>| filename	Redirects standard output to filename, even if the file exists and noclobber is set.
>> filename	Redirects and appends standard output to filename unless filename exists and noclobber is set. If noclobber is not set, this redirection creates filename if it does not exist.
&> filename	Redirects standard output and standard error to filename.
<&m	Duplicates standard input from file descriptor m
[n] >&m	Duplicates standard output or file descriptor n if specified from file descriptor m
[n] <&-	Closes standard input or file descriptor n if specified
[n] >&-	Closes standard output or file descriptor n if specified.

8.3 Writing a Simple Shell Script

A shell script is a file that holds commands that the shell can execute. The commands in a shell script can be any commands you can enter in response to a shell prompt. For example, a command in a shell script might run a Linux utility, a compiled program, or another shell script. Like the commands you give on the command line, a command in a shell script can use ambiguous file references and can have its input or output redirected from or to a file or sent through a pipe. You can also use pipes and redirection with the input and output of the script itself.

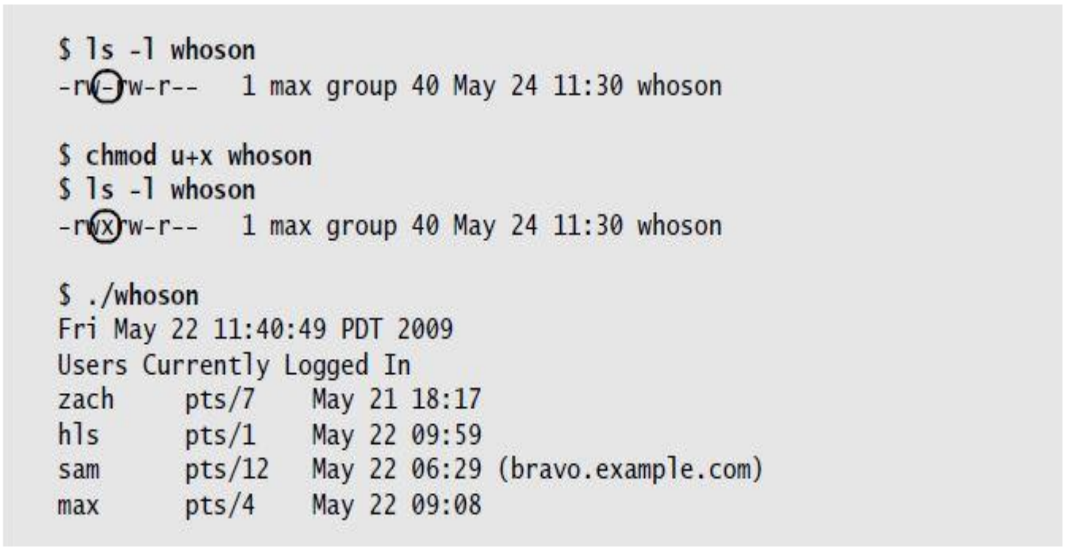
chmod: Makes a File Executable

To execute a shell script by giving its name as a command, you must have permission to read and execute the file that contains the script. Read permission enables you to read the file that holds the script. Execute permission tells the shell and the system that the owner, group, and/or public has permission to execute the file; it implies that the content of the file is executable. When you create a shell script using an editor, the file does not typically have its execute permission set. The following example shows a file named whoson that contains a shell script:

```
$ cat whoson
date
echo "Users Currently Logged In"
```

who

$ **./whoson**



```
$ ls -l whoson
-rw-rw-r--   1 max group 40 May 24 11:30 whoson

$ chmod u+x whoson
$ ls -l whoson
-rwxrw-r--   1 max group 40 May 24 11:30 whoson

$ ./whoson
Fri May 22 11:40:49 PDT 2009
Users Currently Logged In
zach     pts/7    May 21 18:17
hls      pts/1    May 22 09:59
sam      pts/12   May 22 06:29 (bravo.example.com)
max      pts/4    May 22 09:08
```

#! Specifies a Shell

You can put a special sequence of characters on the first line of a shell script to tell the operating system which shell (or other program) should execute the file. Because the operating system checks the initial characters of a program before attempting to execute it using exec, these characters save the system from making an unsuccessful attempt.

# Begins a Comment

Comments make shell scripts and all code easier to read and maintain by you and others. The comment syntax is common to both the Bourne Again and the TC Shells.

Executing a Shell Script

A command on the command line causes the shell to fork a new process, creating a duplicate of the shell process (a subshell). The new process attempts to exec (execute) the command. Like fork, the exec routine is executed by the operating system (a system call).

Separating and Grouping Commands

Whether you give the shell commands interactively or write a shell script, you must separate commands from one another.

8.4 Job Control

A job is a command pipeline. For example: you run a simple job whenever you give the shell a command. If you type date on the command line and press RETURN, you have run a job. You can also create several jobs with multiple commands on a single command line. For example:

$ **find . -print | sort | lpr& grep -l max /tmp/* >maxfiles&**

[1] 18839

[2] 18876

jobs: Lists Jobs

The jobs built in lists all background jobs. Following, the sleep command runs in the background and creates a background job that jobs reports on:

```
$ sleep 60 &
[1] 7809
$ jobs
[1] + Running sleep 60 &
```

fg: Brings a Job to the Foreground

The shell assigns a job number to each command you run in the background. For each job run in the background, the shell lists the job number and PID number immediately, just before it issues a prompt:

```
$ xclock&
[1] 1246
$ date &
[2] 1247
$ Tue Dec 2 11:44:40 PST 2008
[2]+ Done date
$ find /usr -name ace -print >findout&
[2] 1269
$ jobs
[1]- Running xclock&
[2]+ Running find /usr -name ace -print >findout&
```

To move a background job to the foreground, use the fgbuiltin followed by the job number. Alternatively, you can give a percent sign (%) followed by the job number as a command. Either of the following commands moves job 2 to the foreground. When you move a job to the foreground, the shell displays the command it is now executing in the foreground.

```
$ fg 2
find /usr -name ace -print >findout
```

or

```
$ %2
find /usr -name ace -print >findout
```

Suspending a Job

Pressing the suspend key (usually CONTROL-Z) immediately suspends (temporarily stops) the job in the foreground and displays a message that includes the word Stopped.

```
CONTROL-Z
[2]+ Stopped find /usr -name ace -print >findout
```

bg: Sends a Job to the Background

To move the foreground job to the background, you must first suspend the job using CONTROL-Z. You can then use the bg built in to resume execution of the job in the background.

```
$ bg
```

```
[2]+ find /usr -name ace -print >findout&
```

8.5 Manipulating the Directory Stack

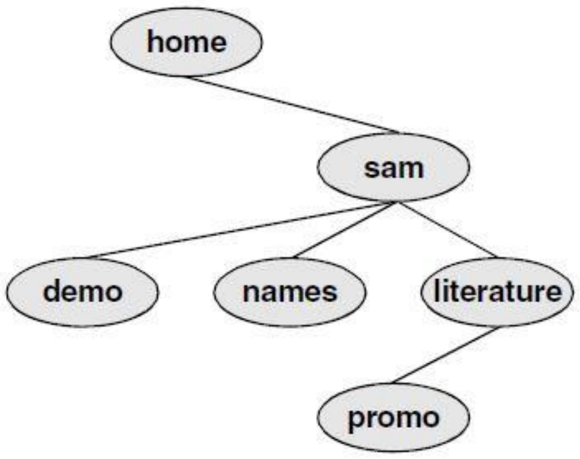
The Bourne Again Shell allows you to store a list of directories you are working with, enabling you to move easily among them. This list is referred to as a stack. It implements LIFO rule.

dirs: Displays the Stack

The dirs. Built in displays the contents of the directory stack. If you call dirs when the directory stack is empty, it displays the name of the working directory: $ dirs

~/literature

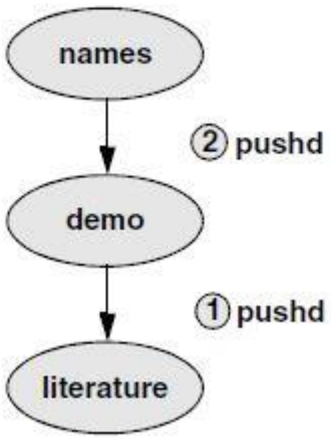
The dirsbuiltin uses a tilde (~) to represent the name of the home directory.



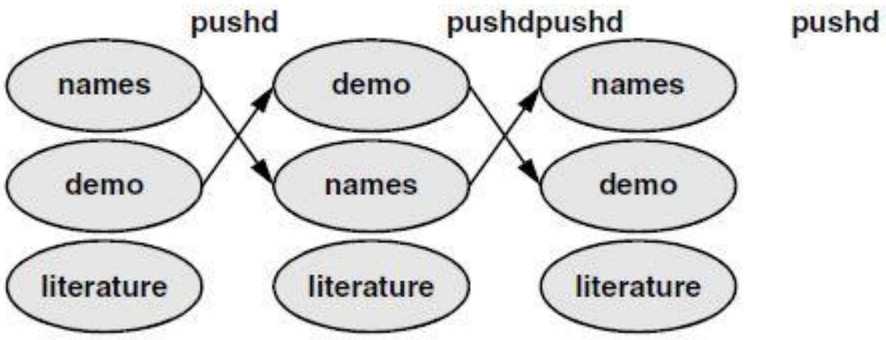
pushd: Pushes a Directory on the Stack

When you supply the pushd (push directory) builtin with one argument, it pushes the directory specified by the argument on the stack, changes directories to the specified directory, and displays the stack.

```
$ pushd ../demo
~/demo ~/literature
$ pwd
/home/sam/demo
$ pushd ../names
~/names ~/demo ~/literature
$ pwd
/home/sam/names
```



When you use pushd without an argument, it swaps the top two directories on the stack, makes the new top directory (which was the second directory) the new working directory, and displays the stack

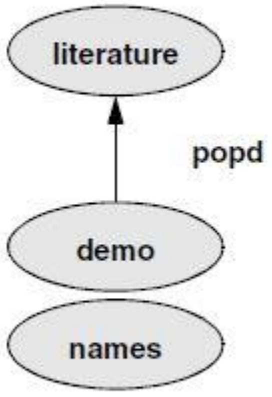


Using pushd in this way, you can easily move back and forth between two directories. You can also use cd – to change to the previous directory, whether you have explicitly created a directory stack. To access another directory in the stack, call pushd with a numeric argument preceded by a plus sign. The directories in the stack are numbered starting with the top directory, which is number 0.

popd: Pops a Directory Off the Stack

To remove a directory from the stack, use the popd (pop directory) builtin.

```
$ dirs
~/literature ~/demo ~/names
$ popd
~/demo ~/names
$ pwd
/home/sam/demo
```



To remove a directory other than the top one from the stack, use popd with a numeric argument preceded by a plus sign.

```
$ dirs
~/literature ~/demo ~/names
$ popd +1
~/literature ~/names
```

8.6 Shell Variables

Within a shell, a shell parameter is associated with a value that is accessible to the user. There are several kinds of shell parameters. The parameters whose names consist of letters, digits, and underscores are often referred to as shell variables, or simply variables. A variable name must start with a letter or underscore, not with a number. Thus A76, MY_CAT, and ___X___ are valid variable names, whereas 69TH_STREET (starts with a digit) and MY-NAME (contains a hyphen) are not.

Keyword variables

Keyword shell variables (or simply keyword variables) have special meaning to the shell and usually have short, mnemonic names. When you start a shell (by logging in, for example), the shell inherits several keyword variables from the environment. Among these variables are HOME, which identifies your home directory, and PATH, which determines which directories the shell searches and in what order to locate commands that you give the shell. The shell creates and initializes (with default values) other keyword variables when you start it. Still other variables do not exist until you set them. You can change the values of most keyword shell variables. It is usually not necessary to change the values of keyword variables initialized in the **/etc/profile** or **/etc/csh.cshrc**systemwide startup files. If you need to change the value of a bash keyword variable, do so in one of your startup files. Just as you can make user-created variables global, so you can make keyword variables global—a task usually done automatically in startup files. You can also make a keyword variable read only.

Positional and special parameters

The names of positional and special parameters do not resemble variable names. Most of these parameters have one-character names (for example, 1, ?, and #) and are referenced (as are all variables) by preceding the name with a dollar sign ($1, $?, and $#). The values of these parameters reflect different aspects of your ongoing interaction with the shell. Whenever you give a command, each argument on the command line becomes the value of a positional parameter. Positional parameters enable you to access command-line arguments, a capability that you will often require when you write shell scripts. The set builtin enables you to assign values to positional parameters. Other frequently needed shell script values, such as the name of the last command executed, the number of command-line arguments, and the status of the most recently executed command, are available as special parameters. You cannot assign values to special parameters.

```
$ person=max
$ echo person
person
$ echo $person
max
```

User-created variables

Shell variables that you name and assign values to are user-created variables. You can change the values of user-created variables at any time, or you can make them read only so that their values cannot be changed.

User created global variables

You can also make user-created variables global. A global variable (also called an environment variable) is available to all shells and other programs you fork from the original shell. One naming convention is to use only uppercase letters for global variables and to use mixed-case or lowercase letters for other variables.

Assigning values

To assign a value to a variable in the Bourne Again Shell, use the following syntax: **VARIABLE=value.** There can be no whitespace on either side of the equal sign (=). An example assignment follows: $ **myvar=abc.** Under the TC Shell the assignment must be preceded by the word set and the SPACEs on either side of the equal sign are optional: $ **set myvar = abc**. The Bourne Again Shell permits you to put variable assignments on a command line. This type of assignment creates a variable that is local to the command shell—that is, the variable is accessible only from the program the command runs.

```
$ cat my_script
echo $TEMPDIR
$ TEMPDIR=/home/sam/temp ./my_script
/home/sam/temp
$ echo $TEMPDIR
$
```

The my_script shell script displays the value of TEMPDIR. The following command runs my_script with TEMPDIR set to /home/sam/temp. The echo builtin shows that the interactive shell has no value for TEMPDIR after running my_script. If TEMPDIR had been set in the interactive shell, running my_script in this manner would have had no effect on its value.

Parameter substitution

Because the echo builtin copies its arguments to standard output, you can use it to display the values of variables.

```
$ person=max
$ echo person
person
$ echo $person
max
```

The second line of the example shows that person does not represent max. Instead, the string person is echoed as person. The shell substitutes the value of a variable only when you precede the name of the variable with a dollar sign ($). Because of the leading $, the shell recognizes that $person is the name of a variable, substitutes the value of the variable, and passes that value to echo.

Quoting the $

You can prevent the shell from substituting the value of a variable by quoting the leading $. Double quotation marks do not prevent the substitution; single quotation marks or a backslash (\) do.

```
$ echo $person
max
$ echo "$person"
max
$ echo '$person'
$person
$ echo \$person
$person
```

Spaces

To assign a value that contains SPACEs or TABs to a variable, use double quotation marks around the value.

```
$ person="max and zach"
$ echo $person
max and zach
$ person=max and zach
bash: and: command not found
```

If you do not quote the variable, the shell collapses each string of blank characters into a single SPACE before passing the variable to the utility:

```
$ person="max and zach"
$ echo $person
max and zach
$ echo "$person"
max and zach
```

Pathname expansion in assignments

When you execute a command with a variable as an argument, the shell replaces the name of the variable with the value of the variable and passes that value to the program being executed. If the value of the variable contains a special character, such as * or ?, the shell may expand that variable.

```
$ memo=max*
$ echo "$memo"
max*
```

The first line assigns the string max* to the variable memo. The Bourne Again Shell does not expand the string because bash does not perform pathname expansion when it assigns a value to a variable. All shells process a command line in a specific order. Within this order bash (but not tcsh) expands variables before it interprets commands. The echo command line, the double quotation marks quote the asterisk (*) in the expanded value of $memo and prevent bash from performing pathname expansion on the expanded memo variable before passing its value to the echo command.

```
$ ls
max.report
max.summary
$ echo $memo
max.reportmax.summary
```

All shells interpret special characters as special when you reference a variable that contains an unquoted special character. The shell expands the value of the memo variable because it is not quoted.Here the shell expands the **$memo variable to max*, expands max*** to**max.report**and **max.summary,** and passes these two values to echo.

User-Created Variables - unset: Removes a Variable

Unless you remove a variable, it exists as long as the shell in which it was created exists. To remove the value of a variable but not the variable itself, assign a null value to the variable (use set person = in tcsh):

```
$ person=
$ echo $person
$
```

You can remove a variable using the unset builtin. The following command removes the variable person:

```
$ unset person
```

8.7 Variable Attributes

readonly: Makes the Value of a Variable Permanent

You can use the readonlybuiltin (not in tcsh) to ensure that the value of a variable cannot be changed.

```
$ person=zach
$ echo $person
zach
$ readonly person
$ person=helen
bash: person: readonly variable
```

If you use the readonlybuiltin without an argument, it displays a list of all readonly shell variables. This list includes keyword variables that are automatically set as readonly as well as keyword or user-created variables that you have declared as readonly.

declare and typeset: Assign Attributes to Variables

The declare and typeset builtins (two names for the same command, neither of which is available in tcsh) set attributes and values for shell variables.

Attribute	Meaning
-a	Declares a variable as an array
-f	Declares a variable to be a function name
-I	Declares a variable to be of type integer
-r	Makes a variable readonly
-x	Exports a varibale, makes it global

$ **declare person1=max:** The first line declares person1 and assigns it a value of max. This command has the same effect with or without the word declare.

$ **declare -r person2=zach:** The readonly and export builtins are synonyms for the commands declare –r and declare –x, respectively.

$ **declare -rx person3=helen**

$ **declare -x person4**: You can declare a variable without assigning a value to it, as the preceding declaration of the variable person4 illustrates. This declaration makes person4 available to all subshells (i.e., makes it global). Until an assignment is made to the variable, it has a null value. You can list the options to declare separately in any order. The following is equivalent to the preceding declaration of person3:$ declare -x -r person3=helen. Use the + character in place of – when you want to remove an attribute from a variable. You cannot remove the readonly attribute. After the following command is given, the variable person3 is no longer exported but it is still readonly.$ declare +x person3. You can use typeset instead of declare.

Listing variable attributes

Without any arguments or options, declare lists all shell variables. The same list is output when you run set without any arguments.

```
$ declare -r
declare -ar BASH_VERSINFO='([0]="3" [1]="2" [2]="39" [3]="1" ... )'
declare -ir EUID="500"
declare -ir PPID="936"
declare -r SHELLOPTS="braceexpand:emacs:hashall:histexpand:history:..."
declare -ir UID="500"
declare -r person2="zach"
declare -rx person3="helen"
```

If you use a declare builtin with options but no variable names as arguments, the command lists all shell variables that have the indicated attributes set. For example, the command declare –r displays a list of all read only shell variables. This list is the same as that produced by the read only command without any arguments.

Integer

By default the values of variables are stored as strings. When you perform arithmetic on a string variable, the shell converts the variable into a number, manipulates it, and then converts it back to a string. A variable with the integer attribute is stored as an integer. Assign the integer attribute as follows:

```
$ declare -i COUNT
```

8.8 Keyword Variables

Keyword variables either are inherited or are declared and initialized by the shell when it starts. You can assign values to these variables from the command line or from a startup file. Typically, you want these variables to apply to all subshells you start as well as to your login shell. For those variables not automatically exported by the shell, you must use export or setenv to make them available to child shells.

HOME: Your Home Directory

By default, your home directory is the working directory when you log in. Your home directory is established when your account is set up; under Linux its name is stored in the **/etc/passwd file.**

```
$ pwd
/home/max/laptop
$ echo $HOME
/home/max
$ cd
$ pwd
/home/max
```

When you log in, the shell inherits the pathname of your home directory and assigns it to the variable HOME. When you give a cd command without an argument, cd makes the directory whose name is stored in HOME the working directory. This example shows the value of the HOME variable and the effect of the cd builtin. After you execute cd without an argument, the pathname of the working directory is the same as the value of HOME: your home directory.

Tilde

The shell uses the value of HOME to expand pathnames that use the shorthand tilde (~) notation to denote a user's home directory. It uses ls to list the files in Max's laptop directory, which is a subdirectory of his home directory:

```
$ echo ~
/home/max
$ ls ~/laptop
tester count lineup
```

PATH: Where the Shell Looks for Programs

If the file with the pathname you specified does not exist, the shell reports command not found. If the file exists as specified but you do not have execute permission for it, or in the case of a shell script you do not have read and execute permission for it, the shell reports Permission denied. If you give a simple filename as a command, the shell searches through certain directories (your search path) for the program you want to execute. It looks in several directories for a file that has the same name as the command and that you have execute permission for (a compiled program) or read and execute permission for (a shell script). The PATH shell variable controls this search. The system directories include /bin and /usr/bin and other directories appropriate to the local system. The PATH variable specifies the directories in the order the shell should search them. Each directory must be separated from the next by a colon. The following command sets PATH so that a search for an executable file starts with the **/usr/local/bin** directory. If it does not find the file in this directory, the shell looks next in **/bin,** and then in **/usr/bin.** If the search fails in those directories, the shell looks in the **~/bin** directory, a subdirectory of the user's home directory. Finally, the shell looks in the working directory. The exporting PATH makes its value accessible to subshells:$ **export PATH=/usr/local/bin:/bin:/usr/bin:~/bin:**A null value in the string indicates the working directory.

MAIL: Where Your Mail Is Kept

The MAIL variable (mail under tcsh) contains the pathname of the file that holds your mail (your mailbox, usually **/var/mail/name**, where name is your username).If MAIL is set and MAILPATH (next) is not set, the shell informs you when mail arrives in the file specified by MAIL. In a graphical environment you can unset MAIL so the shell does not display mail reminders in a terminal emulator window. The MAIL variable and other mail-related shell variables do not do anything unless you have a local mail server.

- **MAILPATH**: The MAILPATH variable (not available under tcsh) contains a list of filenames separated by colons. If this variable is set, the shell informs you when any one of the files is modified.
- **MAILCHECK**: This variable (not available under tcsh) specifies how often, in seconds, the shell checks for new mail. The default is 60 seconds. If you set this variable to zero, the shell checks before each prompt.

PS1: User Prompt (Primary)

The default Bourne Again Shell prompt is a dollar sign ($). When you run bash with root privileges, bash typically displays a pound sign (#) prompt. The PS1 variable holds the prompt string that the shell uses to let you know that it is waiting for a command. When you change the value of PS1 or prompt, you change the appearance of your prompt. You can customize the prompt displayed by PS1. For example, the assignment

```
$ PS1="[\u@\h \W \!]$ "
```

displays the following prompt:

[user@host directory event]$

where user is the username, host is the hostname up to the first period, directory is the basename of the working directory, and event is the event number of the current command.

Symbol	Display in Prompt
\$	# is the user is running with root priviliges; otherwise, $
\w	Pathname of the working directory
\W	Basename of the working directory
\!	Current event (history) number

\d	Date in Weekday Month Date format
\h	Machine hostname, without the domain
\H	Full machine hostname, including the domain
\u	Username of the current user
\@	Current time of day in 12-hour, AM/PM format
\T	Current time of day in 12-hour HH:MM:SS format
\A	Current time of day in 24-hour HH:MM format
\t	Current time of day in 24-hour HH:MM:SS format

PS2: User Prompt (Secondary)

The PS2 variable holds the secondary prompt. On the first line, an unclosed quoted string follows echo. The shell assumes the command is not finished and, on the second line, gives the default secondary prompt (>). This prompt indicates the shell is waiting for the user to continue the command line. The shell waits until it receives the quotation mark that closes the string. Only then does it execute the command:

```
$ echo "demonstration of prompt string
>2"
demonstration of prompt string
2
$ PS2="secondary prompt: "
$ echo "this demonstrates
secondary prompt: prompt string 2"
this demonstrates
prompt string 2
```

The second command changes the secondary prompt to secondary prompt: followed by a SPACE. A multiline echo demonstrates the new prompt.

PS3: Menu Prompt

The PS3 variable holds the menu prompt for the select control structure.

PS4: Debugging Prompt

The PS4 variable holds the bash debugging symbol.

Keyword Variables

Variable	Value
BASH_ENV	The pathname of the startup file for noninteractive shells
CDPATH	The cd search path
COLUMNS	The width of the display used by **select**
FCEDIT	The name of the editor that fc uses by default
HISTFILE	The pathname of the file that holds the history list
HISTFILESIZE	The maximum number of entries saved in **HISTFILE**
HISTSIZE	The maximum number of entries saved in the history list
HOME	The pathname of the user's home directory; used as the default argument for cd and in tilde expansion
IFS	Internal Field Separator; used for word splitting
INPUTRC	The pathname of the Readline startup file
LANG	The locale category when that category is not specifically set with an **LC_***variable
LC_*	A group of variables that specify locale categories including **LC_COLLATE, LC_CTYPE, LC_MESSAGES, and LC_NUMERIC; use the locale builtin to display** a complete list with values
LINES	The height of the display used by **select**
MAIL	The pathname of the file that holds a user's mail
MAILCHECK	How often, in seconds, bash checks for mail
MAILPATH	A colon-separated list of file pathnames that bash checks for mail in
PATH	A colon-separated list of directory pathnames that bash looks for commands

PROMPT_COMMAND	A command that bash executes just before it displays the primary prompt
PS1	Prompt String 1; the primary prompt
PS2	Prompt String 2; the secondary prompt
PS3	The prompt issued by **select**
PS4	The bash debugging symbol
REPLY	Holds the line that read accepts also used by **select**

Special Characters

Characters	**Use**
NEWLINE	Initiates the execution of a command page
;	Separates commands
()	Group commands for execution by a subshell or identifies a function
(( ))	Expands an arithmetic expression
&	Executes a command in the background
|	Pipe
>	Redirects standard output
>>	Appends standard output
<	Redirects standard input
<<	Here document
*	Any string of zero or more characters in an ambiguous file reference
?	Any single character in an ambiguous file reference
\	Quotes the following character

‘	Quotes a string, preventing all substitution
“	Quotes a string, allowing only variable and command substitution
‘...’	Performs command substitution
[ ]	Character class in an ambiguous file reference
$	Reference a variable
.	Executes a command
#	Begins a comment
{ }	Surrounds the contents of a function
:	Returns true
&&	Boolean AND
| |	Boolean OR
!	Boolean NOT
$( )	Performs command substitution
[ ]	Evaluates an arithmetic expression

8.9 Processes

A process is the execution of a command by the Linux kernel. The shell that starts when you log in is a command, or a process, like any other. When you give the name of a Linux utility on the command line, you initiate a process. When you run a shell script, another shell process is started and additional processes are created for each command in the script. Depending on how you invoke the shell script, the script is run either by the current shell or, more typically, by a subshell (child) of the current shell. A process is not started when you run a shell builtin, such as cd.

Process Structure

Like the file structure, the process structure is hierarchical, with parents, children, and even a root. A parent process forks a child process, which in turn can fork other processes. The term fork indicates that, as with a fork in the road, one process turns into two. Initially the two forks are identical except that one is identified as the parent and one as the child. You can also use the term spawn; the words are interchangeable. The operating system routine, or system call, that creates a new process is named fork().When Linux begins execution when a system is started, it starts init, a single process called a spontaneous process, with PID number 1. This process holds the same position in the process structure as the root directory does in the file structure: It is the ancestor of all processes the system and users work with. When a command-line system is in multiuser mode, init runs getty or mingetty processes, which display login: prompts on terminals. When a user responds to the prompt and presses RETURN, getty hands control over to a utility named login, which checks the username and password combination. After the user logs in, the login process becomes the user's shell process.

Process Identification

Linux assigns a unique PID number at the inception of each process. As long as a process exists, it keeps the same PID number. During one session the same process is always executing the login shell. When you fork a new process—for example, when you use an editor—the PID number of the new (child) process is different from that of its parent process. When you return to the login shell, it is still being executed by the same process and has the same PID number as when you logged in.

- $ **sleep 10 &**

```
[1] 22789
```

- $ **ps -f**

```
UID PID PPID C STIME TTY TIME CMD
max 21341 21340 0 10:42 pts/16 00:00:00 bash
max 22789 21341 0 17:30 pts/16 00:00:00 sleep 10
max 22790 21341 0 17:30 pts/16 00:00:00 ps -f
```

The following example shows that the process running the shell forked (is the parent of) the process running ps. When you call it with the –f option, ps displays a full listing of information about each process. The line of the ps display with bash in the CMD column refers to the process running the shell. The column headed by PID identifies the PID number. The column headed PPID identifies the PID number of the parent of the process. From the PID and PPID columns you can see that the process running the shell (PID 21341) is the parent of the process running sleep (PID 22789).The parent PID number of sleep is the same as the PID number of the shell (21341).A second pair of sleep and ps –f commands shows that the shell is still being run by the same process but that it forked another process to run sleep:

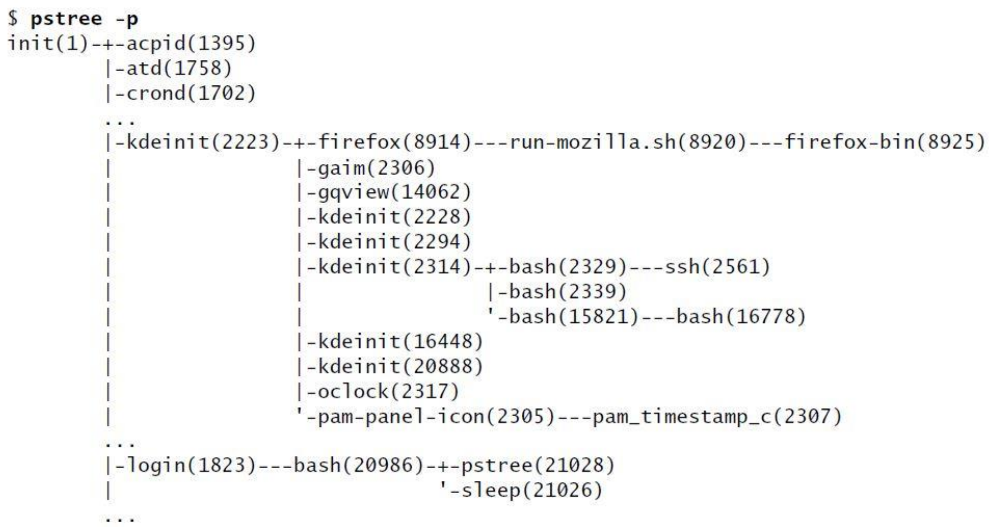
- $ **sleep 10 &**

```
[1] 22791
```

- $ **ps -f**

```
UID PID PPID C STIME TTY TIME CMD
max 21341 21340 0 10:42 pts/16 00:00:00 bash
max 22791 21341 0 17:31 pts/16 00:00:00 sleep 10
max 22792 21341 0 17:31 pts/16 00:00:00 ps -f
```

You can also use pstree (or ps ––forest, with or without the –e option) to see the parent–child relationship of processes



```
$ pstree -p
init(1)-+-acpid(1395)
        |-atd(1758)
        |-crond(1702)
        ...
        |-kdeinit(2223)-+-firefox(8914)---run-mozilla.sh(8920)---firefox-bin(8925)
        |               |-gaim(2306)
        |               |-gqview(14062)
        |               |-kdeinit(2228)
        |               |-kdeinit(2294)
        |               |-kdeinit(2314)-+-bash(2329)---ssh(2561)
        |               |               |-bash(2339)
        |               |               '-bash(15821)---bash(16778)
        |               |-kdeinit(16448)
        |               |-kdeinit(20888)
        |               |-oclock(2317)
        |               '-pam-panel-icon(2305)---pam_timestamp_c(2307)
        ...
        |-login(1823)---bash(20986)-+-pstree(21028)
        |                           '-sleep(21026)
        ...
```

The line that starts with –kdeinit shows a graphical user running many processes, including firefox, gaim, and oclock. The line that starts with –login shows a textual user running sleep in the background and running pstree in the foreground.

Executing a Command

When you give the shell a command, it usually forks [spawns using the fork() system call] a child process to execute the command. While the child process is executing the command, the parent process sleeps [implemented as the sleep() system call]. While a process is sleeping, it does not use any computer time; it remains inactive, waiting to wake up. When the child process finishes executing the command, it tells its parent of its success or failure via its exit status and then dies. The parent process (which is running the shell) wakes up and prompts for another command. When you run a process in the background by ending a command with an ampersand (&), the shell forks a child process without going to sleep and without waiting for the child process to run to completion. The parent process, which is executing the shell, reports the job number and PID number of the child process and prompts for another command. The child process runs in the background, independent of its parent. Although the shell forks a process to run most of the commands you give it, some commands are built into the shell. The shell does not need to fork a process to run builtins.

Variables Within a given process, such as your login shell or a subshell, you can declare, initialize, read, and change variables. By default, however, a variable is local to a process. When a process forks a child process, the parent does not pass the value of a variable to the child. You can make the value of a variable available to child processes (global) by using the export builtin under bash or the setenvbuiltin under tcsh.

8.10 History

The history mechanism, a feature adapted from the C Shell, maintains a list of recently issued command lines, also called events. There are various features of this:

- It provides a quick way to reexecute any of the events in the list.
- Execute variations of previous commands and reuses arguments from them.
- Replicates complicated commands and arguments that you used earlier in this login session or in a previous one.
- Enter a series of commands that differ from one another in minor ways.
- Serves as a record of what you have done.
- Keeps a record of a procedure that involved a series of commands.

Variables That Control History

The TC Shell's history mechanism is similar to bash's but uses different variables and has other differences.

Variable	Default	Function
HISTSIZE	500 events	Maximum number of events saved during a session
HISTFILE	~/.bash_history	Location of the history file
HISTFILESIZE	500 events	Maximum number of events saved between session

- **HISTSIZE:** The value of the HISTSIZE variable determines the number of events preserved in the history list during a session. A value in the range of 100 to 1,000 is normal.
- **HISTFILE:** When you exit from the shell, the most recently executed commands are saved in the file whose name is stored in the HISTFILE variable (the default is ~/.bash_history). The next time you start the shell, this file initializes the history list.
- **HISTFILESIZE:** The value of the HISTFILESIZE variable determines the number of lines of history saved in HISTFILE.

The Bourne Again Shell assigns a sequential event number to each command line. You can display this event number as part of the bash prompt by including \! in PS1. Give the following command manually, or place it in **~/.bash_profile**to affect future sessions, to establish a history list of the 100 most recent events: $ **HISTSIZE=100.** The following command causes bash to save the 100 most recent events across login sessions: $ **HISTFILESIZE=100.** After you set HISTFILESIZE, you can log out and log in again, and the 100 most recent events from the previous login session will appear in your history list. Give the command **history** to display the events in the history list. This list is ordered so that the oldest events appear at the top. A tcsh history list includes the time the command was executed. The following history list includes a command to modify the bash prompt so it displays the history event number. The last event in the history list is the history command that displayed the list.

```
32 $ history | tail
23 PS1="\! bash$ "
24 ls -l
25 cat temp
26 rm temp
27 vim memo
28 lpr memo
29 vim memo
30 lpr memo
31 rm memo
32 history | tail
```

As you run commands and your history list becomes longer, it may run off the top of the screen when you use the history builtin. Pipe the output of history through less to browse through it, or give the command history 10 or history | tail to look at the ten most recent commands.

History alias

Creating the following aliases makes working with history easier. The first allows you to give the command h to display the ten most recent events. The second alias causes the command hg string to display all events in the history list that contain string. Put these aliases in your **~/.bashrc file** to make them available each time you log in.

- $ alias **'h=history | tail'**

```
$ alias 'hg=history | grep'
```

Reexecuting and Editing Commands

You can reexecute any event in the history list. This feature can save you time, effort, and aggravation. Not having to reenter long command lines allows you to reexecute events more easily, quickly, and accurately than you could if you had to retype the command line in its entirety. You can recall, modify, and reexecute previously executed events in three ways:

- fc builtin,
- exclamation point commands ,
- Readline Library

fc: Displays, Edits, and Reexecutes Commands

The fc (fix command) builtin (not in tcsh) enables you to display the history list and to edit and reexecute previous commands. It provides many of the same capabilities as the command-line editors. When you call fc with the –l option, it displays commands from the history list. Without any arguments, fc –l lists the 16 most recent commands in a numbered list, with the oldest appearing first:

```
$ fc -l
1024 cd
1025 view calendar
1026 vim letter.adams01
1027 aspell -c letter.adams01
1028 vim letter.adams01
1029 lpr letter.adams01
1030 cd ../memos
1031 ls
1032 rm *0405
1033 fc -l
1034 cd
1035 whereisaspell
1036 man aspell
1037 cd /usr/share/doc/*aspell*
1038 pwd
1039 ls
1040man-htm
```

The fc builtin can take zero, one, or two arguments with the –l option. The arguments specify the part of the history list to be displayed: fc –l **[first [last]]**. The fc builtin lists commands beginning with the most recent event that matches first. The argument can be an event number, the first few characters of the command line, or a negative number, which is taken to be the nth previous command. Without last, fc displays events through the most recent. If you include last, fc displays commands from the most recent event that matches first through the most recent event that matches last. The next command displays the history list from event 1030 through event 1035: $ **fc -l 1030 1035**

```
1030 cd ../memos

1031 ls

1032 rm *0405

1033 fc -l

1034 cd

1035 whereisaspell
```

This command lists the most recent event that begins with view through the most recent command line that begins with whereis: $ **fc -l view whereis**

```
1025 view calendar

1026 vim letter.adams01

1027 aspell -c letter.adams01

1028 vim letter.adams01

1029 lpr letter.adams01

1030 cd ../memos

1031 ls

1032 rm *0405

1033 fc -l

1034 cd

1035 whereisaspell
```

To list a single command from the history list, use the same identifier for the first and second arguments. The following command lists event 1027: $ **fc -l 1027 1027**

```
1027 ell -c letter.adams01
```

You can use fc to edit and reexecute previous commands:**fc [–e editor] [first [last]]**. When you call fc with the –e option followed by the name of an editor, fc calls the editor with event(s) in the Work buffer. By default, fc invokes the nano editor. Without first and last, it defaults to the most recent command: $ **fc -e vi**. The fc builtin uses the stand-alone vim editor. If you set the FCEDIT variable, you do not need to use the –e option to specify an editor on the command line. Because the value of FCEDIT has been changed to /usr/bin/emacs and fc has no arguments, the following command edits the most recent command using the emacs editor:

```
$ export FCEDIT=/usr/bin/emacs

$ fc
```

If you call it with a single argument, fc invokes the editor on the specified command. This command starts the editor with event 1029 in the Work buffer. When you exit from the editor, the shell executes the command: $ **fc 1029.** You can identify commands with numbers or by specifying the first few characters of the command name. This command calls the editor to work on events from the most recent event that begins with the letters vim through event 1030:$ **fc vim 1030.** You can reexecute previous commands without using an editor. If you call fc with the –s option, it skips the editing phase and reexecutes the command. This command reexecutes event 1029: $ **fc -s 1029**

```
lpr letter.adams01
```

This command reexecutes the previous command:

```
$ fc -s
```

When you reexecute a command, you can tell fc to substitute one string for another.This command substitutes the string john for the string adams in event 1029 and executes the modified event:

```
$ fc -s adams=john 1029
lpr letter.john01
```

Using an Exclamation Point (!) to Reference Events

The C Shell history mechanism uses an exclamation point to reference events. For example, the **!! command** reexecutes the previous event, and the shell replaces the!**$** token with the last word on the previous command line. You can reference an event by using its absolute event number, its relative event number, or the text it contains. All references to events, called event designators, begin with an exclamation point (!). One or more characters follow the exclamation point to specify an event. You can put history events anywhere on a command line. To escape an exclamation point so that the shell interprets it literally instead of as the start of a history event, precede the exclamation point with a backslash (\) or enclose it within single quotation marks.

Event Designators

Designator	Meaning
!	Starts a history event unless followed immediately by SPACE, NEWLINE, = or (.
!!	The previous command
!n	Command number n in the history list
!-n	The nth preceding command
!string	The most recent command line that started with string
!?string[?]	The most recent command that contained string. The last ? Is optional
!#	The current command
!{event}	The event is an event designator. The braces isolate event from the surrounding text.

An event designator specifies a command in the history list.

You can reexecute the previous event by giving a **!!** command. In the following example, event 45 reexecutes event 44:

```
44 $ ls -l text
-rw-rw-r-- 1 max group 45 Apr 30 14:53 text
```

```
45 $ !!
ls -l text
-rw-rw-r-- 1 max group 45 Apr 30 14:53 text
```

A number following an exclamation point refers to an event. If that event is in the history list, the shell executes it. Otherwise, the shell displays an error message. A negative number following an exclamation point references an event relative to the current event. For example, the command !–3 refers to the third preceding event. After you issue a command, the relative event number of a given event changes (event –3 becomes event –4). Both of the following commands reexecute event 44:

```
51 $ !44
ls -l text
-rw-rw-r-- 1 max group 45 Apr 30 14:53 text
52 $ !-8
ls -l text
-rw-rw-r-- 1 max group 45 Apr 30 14:53 text
```

When a string of text follows an exclamation point, the shell searches for and executes the most recent event that began with that string. If you enclose the string within question marks, the shell executes the most recent event that contained that string. The final question mark is optional if a RETURN would immediately follow it.

The Readline Library

Command-line editing under the Bourne Again Shell is implemented through the Readline Library, which is available to any application written in C. Any application that uses the Readline Library supports line editing that is consistent with that provided by bash. Programs that use the Readline Library, including bash, read **~/.inputrc** for key binding information and configuration settings. The **––noediting** command-line option turns off command-line editing in bash.You can choose one of two editing modes when using the Readline Library in bash: emacs or vi(m). Both modes provide many of the commands available in the standalone versions of the emacs and vim editors. You can also use the ARROW keys to move around. Up and down movements move you backward and forward through the history list. In addition, Readline provides several types of interactive word completion. The default mode is emacs; you can switch to vi mode with the following command: $ **set -o vi.** This command switches back to emacs mode:$ **set -o emacs**

Before you start, make sure the shell is in vi mode. When you enter bash commands while in vi editing mode, you are in Input mode. As you enter a command, if you discover an error before you press RETURN, you can press ESCAPE to switch to vim Command mode.

Unlike the vim editor, emacs is modeless. You need not switch between Command mode and Input mode because most emacs commands are control characters, allowing emacs to distinguish between input and commands. Like vim, the emacs command-line editor provides commands for moving the cursor on the command line and through the command history list and for modifying part or all of a command. However, in a few cases, the emacs command-line editor commands differ from those in the stand-alone emacs editor. In emacs you perform cursor movement by using both CONTROL and ESCAPE commands.

You can use the TAB key to complete words you are entering on the command line. This facility, called completion, works in both vi and emacs editing modes and is similar to the completion facility available in tcsh. Several types of completion are possible, and which one you use depends on which part of a command line you are typing when you press TAB.

Command Completion

If you are typing the name of a command (usually the first word on the command line), pressing TAB initiates command completion, in which bash looks for a command whose name starts with the part of the word you have typed. If no command starts with the characters you entered, bash beeps. If there is one such command, bash completes the command name. If there is more than one choice, bash does nothing in vi mode and beeps in emacs mode. Pressing TAB a second time causes bash to display a list of commands whose names start with the prefix you typed and allows you to continue typing the command name. In the following example, the user types bz and presses TAB. The shell beeps (the user is in emacs mode) to indicate that several commands start with the letters bz. The user enters another TAB to cause the shell to display a list of commands that start with bz followed by the command line as the user had entered it so far:

$ **bz → TAB** ***(beep) → TAB***

bzcatbzdiff bzip2 bzless

bzcmpbzgrep bzip2recover bzmore

$ bz■

Next the user types c and presses TAB twice. The shell displays the two commands that start with bzc. The user types a followed by TAB. At this point the shell completes the command because only one command starts with bzca.

$ bz**c → TAB (beep) → TAB**

bzcatbzcmp

$ bzc**a → TAB →t ■**

Pathname Completion

Pathname completion, which also uses TABs, allows you to type a portion of a pathname and have bash supply the rest. If the portion of the pathname you have typed is sufficient to determine a unique pathname, bash displays that pathname. If more than one pathname would match it, bash completes the pathname up to the point where there are choices so that you can type more. When you are entering a pathname, including a simple filename, and press TAB, the shell beeps (if the shell is in emacs mode—in vi mode there is no beep). It then extends the command line as far as it can.

$ **cat films/dar → TAB** ***(beep) cat films/dark_***

In the films directory every file that starts with dar has k_ as the next characters, so bash cannot extend the line further without making a choice among files. The shell leaves the cursor just past the _ **character.** At this point you can continue typing the pathname or press TAB twice. In the latter case bash beeps, displays your choices, redisplays the command line, and again leaves the cursor just after the _ character.

$ **cat films/dark_ → TAB** ***(beep) → TAB***

dark_passagedark_victory

$ cat films/dark_

When you add enough information to distinguish between the two possible files and press TAB, bash displays the unique pathname. If you enter p followed by TAB after the _ character, the shell completes the command line: $ cat films/dark_p → TAB → assage. Because there is no further ambiguity, the shell appends a SPACE so you can finish typing the command line or just press RETURN to execute the command. If the complete pathname is that of a directory, bash appends a slash (/) in place of a SPACE.

Variable Completion

When you are typing a variable name, pressing TAB results in variable completion, wherein bash attempts to complete the name of the variable. In case of an ambiguity, pressing TAB twice displays a list of choices:

$ **echo $HO → TAB → TAB**

$HOME $HOSTNAME $HOSTTYPE

$ echo $HOM → **TAB** → **E**

.inputrc: Configuring the Readline Library

The Bourne Again Shell and other programs that use the Readline Library read the file specified by the INPUTRC environment variable to obtain initialization information. If INPUTRC is not set, these programs read the ~/.inputrc file. They ignore lines of .inputrc that are blank or that start with a pound sign (#).You can set variables in .inputrc to control the behavior of the Readline Library using the following syntax:set **variable value.**

Readline variables

Variable	Effect
editing-mode	Set to vi to start Readline in vi mode. Set to emacs to start Readline in emacs mode (the default). Similar to the set –o vi and set –o emacs shell commands
horizontal-scroll-mode	Set to on to cause long lines to extend off the right edge of the display area. Moving the cursor to the right when it is at the right edge of the display area shifts the line to the left so you can see more of the line. You can shift the line back by moving the cursor back past the left edge. The default value is off, which causes long lines to wrap onto multiple lines of the display.
mark-directories	Set to off to cause Readline not to place a slash (/) at the end of directory names it completes. The default value is on.
mark-modified-lines	Set to on to cause Readline to precede modified history lines with an asterisk. The default value is off.

8.11 Alias

An alias is a usually short name that the shell translates into another usually longer name or complex command. Aliases allow you to define new commands by substituting a string for the first token of a simple command. They are typically placed in the ~/.bashrc (bash) or ~/.tcshrc (tcsh) startup files so that they are available to interactive subshells. Under bash the syntax of the alias builtin is:alias [name[=value]]. Under tcsh the syntax is: alias [name[value]].

In the bash syntax no SPACEs are permitted around the equal sign. If value contains SPACEs or TABs, you must enclose value within quotation marks. Unlike aliases under tcsh, a bash alias does not accept an argument from the command line in value. Use a bash function when you need to use an argument. An alias does not replace itself, which avoids the possibility of infinite recursion in handling an alias such as the following: $ alias ls='ls -F'. You can nest aliases. Aliases are disabled for noninteractive shells (that is, shell scripts). To see a list of the current aliases, give the command alias. To view the alias for a particular name, give the command alias followed by the name of the alias. You can use the unalias builtin to remove an alias. When you give an alias builtin command without any arguments, the shell displays a list of all defined aliases:

```
$ alias

alias ll='ls -l'

alias l='ls -ltr'

alias ls='ls -F'

alias zap='rm -i'
```

Give an alias command to see which aliases are in effect. You can delete the aliases you do not want from the appropriate startup file.

Single Versus Double Quotation Marks in Aliases

The choice of single or double quotation marks is significant in the alias syntax when the alias includes variables. If you enclose value within double quotation marks, any variables that appear in value are expanded when the alias is created. If you enclose value within single quotation marks, variables are not expanded until the alias is used. The PWD keyword variable holds the pathname of the working directory (Shown in next slide).Max creates two aliases while he is working in his home directory. Because he uses double quotation marks when he creates the dirA alias, the shell substitutes the value of the working directory when he creates this alias. The alias dirA command displays the dirA alias and shows that the substitution has already taken place:

```
$ echo $PWD

/home/max

$ alias dirA="echo Working directory is $PWD"

$ alias dirA

alias dirA='echo Working directory is /home/max'
```

When Max creates the dirB alias, he uses single quotation marks, which prevent the shell from expanding the $PWD variable. The alias dirB command shows that the dirB alias still holds the unexpanded $PWD variable:

```
$ alias dirB='echo Working directory is $PWD'

$ alias dirB

alias dirB='echo Working directory is $PWD'
```

After creating the dirA and dirB aliases, Max uses cd to make cars his working directory and gives each of the aliases as commands. The alias he created using double quotation marks displays the name of the directory he created the alias in as the working directory (which is wrong). In contrast, the dirB alias displays the proper name of the working directory.

```
$ cd cars

$ dirA

Working directory is /home/max

$ dirB

Working directory is /home/max/cars
```

Examples of Aliases

The following alias allows you to type r to repeat the previous command: $ alias r='fc -s'. If you use the command ls –ltr frequently, you can create an alias that substitutes ls –ltr when you give the command l:

```
$ alias l='ls -ltr'
$ l
total 41
-rw-r--r-- 1 max group 30015 Mar 1 2008 flute.ps
-rw-r----- 1 max group 3089 Feb 11 2009 XTerm.ad
-rw-r--r-- 1 max group 641 Apr 1 2009 fixtax.icn
-rw-r--r-- 1 max group 484 Apr 9 2009 maptax.icn
drwxrwxr-x 2 max group 1024 Aug 9 17:41 Tiger
drwxrwxr-x 2 max group 1024 Sep 10 11:32 testdir
-rwxr-xr-x 1 max group 485 Oct 21 08:03 floor
drwxrwxr-x 2 max group 1024 Oct 27 20:19 Test_Emacs
```

Another common use of aliases is to protect yourself from mistakes. The following example substitutes the interactive version of the rm utility when you give the command zap:

```
$ alias zap='rm -i'
$ zap f*
rm: remove 'fixtax.icn'? n
rm: remove 'flute.ps'? n
rm: remove 'floor'? n
```

The –i option causes rm to ask you to verify each file that would be deleted, thereby helping you avoid deleting the wrong file. You can also alias rm with the rm –i command: alias rm='rm –i'.This aliases cause the shell to substitute ls –l each time you give an ll command and ls –F each time you use ls:

```
$ alias ls='ls -F'
$ alias ll='ls -l'
```

```
$ ll
total 41
drwxrwxr-x 2 max group 1024 Oct 27 20:19 Test_Emacs/
drwxrwxr-x 2 max group 1024 Aug 9 17:41 Tiger/
-rw-r----- 1 max group 3089 Feb 11 2009 XTerm.ad
-rw-r--r-- 1 max group 641 Apr 1 2009 fixtax.icn
-rw-r--r-- 1 max group 30015 Mar 1 2008 flute.ps
-rwxr-xr-x 1 max group 485 Oct 21 08:03 floor*
-rw-r--r-- 1 max group 484 Apr 9 2009 maptax.icn
drwxrwxr-x 2 max group 1024 Sep 10 11:32 testdir/
```

The –F option causes ls to print a slash (/) at the end of directory names and an asterisk (*) at the end of the names of executable files. In this example, the string that replaces the alias ll (ls –l) itself contains an alias (ls). When it replaces an alias with its value, the shell looks at the first word of the replacement string to see whether it is an alias. In the preceding example, the replacement string contains the alias ls, so a second substitution occurs to produce the final command ls –F –l.When given a list of aliases without the =value or value field, the alias builtin responds by displaying the value of each defined alias. The alias builtin reports an error if an alias has not been defined:

```
$ alias ll l lszapwx
alias ll='ls -l'
alias l='ls -ltr'
alias ls='ls -F'
alias zap='rm -i'
bash: alias: wx: not found
```

You can avoid alias substitution by preceding the aliased command with a backslash (\):

```
$ \ls
Test_Emacs XTerm.ad flute.ps maptax.icn
Tiger fixtax.icn floor testdir
```

Because the replacement of an alias name with the alias value does not change the rest of the command line, any arguments are still received by the command that gets executed:

```
$ ll f*
-rw-r--r-- 1 max group 641 Apr 1 2009 fixtax.icn
-rw-r--r-- 1 max group 30015 Mar 1 2008 flute.ps
-rwxr-xr-x 1 max group 485 Oct 21 08:03 floor*
```

You can remove an alias with the unalias builtin. When the zap alias is removed, it is no longer displayed with the alias builtin and its subsequent use results in an error message:

```
$ unalias zap
$ alias
alias ll='ls -l'
alias l='ls -ltr'
alias ls='ls -F'
$ zap maptax.icn
```

8.12 Functions

A bash shell function (tcsh does not have functions) is similar to a shell script in that it stores a series of commands for execution at a later time. However, because the shell stores a function in the computer's main memory (RAM) instead of in a file on the disk, the shell can access it more quickly than the shell can access a script. The shell also preprocesses (parses) a function so that it starts up more quickly than a script. Finally the shell executes a shell function in the same shell that called it. If you define too many functions, the overhead of starting a subshell can become unacceptable. You can declare a shell function in the ~/.bash_profile startup file, in the script that uses it, or directly from the command line. You can remove functions with the unset builtin. The shell does not retain functions after you log out.

Removing variables and functions

If you have a shell variable and a function with the same name, using unset removes the shell variable. If you then use unset again with the same name, it removes the function. The syntax that declares a shell function is

[function] function-name ()

{

```
commands
}
```

The word function is optional, function-name is the name you use to call the function, and commands comprise the list of commands the function executes when you call it. The commands can be anything you would include in a shell script, including calls to other functions. There are some features of functions:

- The opening brace ({) can appear on the same line as the function name.
- Aliases and variables are expanded when a function is read, not when it is executed.
- You can use the break statement within a function to terminate its execution.
- Shell functions are useful as a shorthand as well as to define special commands.
- It creates a simple function that displays the date, a header, and a list of the people who are logged in on the system. This function runs the same commands as the whoson script.
- The greater than (>) signs are secondary shell prompts (PS2); do not enter them.

```
$ function whoson ()
> {
> date
> echo "Users Currently Logged On"
> who
> }
$ whoson
Sun Aug 9 15:44:58 PDT 2009
Users Currently Logged On
hls console Aug 8 08:59 (:0)
max pts/4 Aug 8 09:33 (0.0)
zach pts/7 Aug 8 09:23 (bravo.example.com)
```

Functions in startup files

If you want to have the whoson function always be available without having to enter it each time you log in, put its definition in ~/.bash_profile. Then run .bash_profile, using the . (dot) command to put the changes into effect immediately:

```
$ cat ~/.bash_profile
export TERM=vt100
stty kill '^u'
whoson ()
{
date
echo "Users Currently Logged On"
who
}
$ . ~/.bash_profile
```

You can specify arguments when you call a function. Within the function these arguments are available as positional parameters. The following example shows the arg1 function entered from the keyboard:

```
$ arg1 ( ) {
> echo "$1"
> }
$ arg1 first_arg
first_arg
```

8.13 Command Line Options

Two kinds of command-line options are available: short and long. Short options consist of a hyphen followed by a letter. Long options have two hyphens followed by multiple characters. Long options must appear before short options on a command line that calls bash.

Option	**Explanation**	**Syntax**
Help	Displays a usage message.	--help
No edit	Prevents users from using the Readline Library to edit command lines in an interactive shell.	--noediting
No profile	Prevents reading these startup files: **/etc/profile, ~/.bash_profile, ~/.bash_login, and ~/.profile.**	--noprofile
No rc	Prevents reading the **~/.bashrc**startup file. This option is on by default if the shell is called as **sh.**	--norc
POSIX	Runs bash in POSIX mode.	--posix
Version	Displays bash version information and exits.	--version
Login	Causes bash to run as though it were a login shell.	-l (lowercase "l")
Shopt	Runs a shell with the opt shopt option . A –O (uppercase "O") sets the option; +O unsets it.	[±] 0 [opt]
End of options	On the command line, signals the end of options. Subsequent tokens are treated as arguments even if they begin with a hyphen **(–).**	--

8.14 Shell Features

You can control the behavior of the Bourne Again Shell by turning features on and off. Different features use different methods to turn features on and off. The set builtin controls one group of features, while the shoptbuiltin controls another group. You can also control many features from the command line you use to call bash.

set ±o: Turns Shell Features On and Off

The bash set builtin (there is a set builtin in tcsh, but it works differently), when used with the –o or +o option, enables, disables, and lists certain bash features. For example, the following command turns on the noclobber feature: $ set -o noclobber. You can turn this feature off (the default) by giving the command: $ set +o noclobber. The command set –o without an option lists each of the features controlled by set, followed by its state (on or off).The command set +o without an option lists the same features in a form you can use as input to the shell.

shopt: Turns Shell Features On and Off

The shopt (shell option) builtin (not available in tcsh) enables, disables, and lists certain bash features that control the behavior of the shell. For example, the following command causes bash to include filenames that begin with a period (.) when it expands ambiguous file references (the –s stands for set): $ shopt -s dotglob. You can turn this feature off (the default) by giving the following command (the **–u** stands for unset): $ **shopt -u dotglob.** The shell displays how a feature is set if you give the name of the feature as the only argument to shopt:$ **shoptdotglob**dotglob off. The command shopt without any options or arguments lists the features controlled by shopt and their state. The command shopt –s without an argument lists the features controlled by shopt that are set or on. The command shopt –u lists the features that are unset or off.

bash features

Feature	Description	Syntax	Alternate Syntax
allexport	Automatically exports all variables and functions you create or modify after givingthis command.	**set –o allexport**	**set –a**
braceexpand	Causes bash to perform brace expansion	**set –o braceexpand**	**set –B**
cdspell	Corrects minor spelling errors in directorynames used as arguments to cd.	**shopt –s cdspell**	
cmdhist	Saves all lines of a multiline command inthe same history entry, adding semicolonsas needed.	**shopt –s cmdhist**	

dotglob	Causes shell special characters (wildcards) in an ambiguous file referenceto match a leading period in a filename. By default, special characters do not match aleading period. You must always specify the filenames . and .. explicitly because nopattern ever matches them.	**shopt –s dotglob**	
emacs	Specifies emacs editing mode for command-line editing	**set –o emacs**	
errexit	Causes bash to exit when a simple command (not a control structure) fails.	**set –o errexit**	**set –e**
execfail	Causes a shell script to continue running when it cannot find the file that is given as an argument to exec. By default a script terminates when exec cannot find the file that is given as its argument.	**shopt –s execfail**	
expand_ aliases	Causes aliases to be expanded(by default it is on for interactive shells and off for noninteractive shells).	**shopt –s expand_alias**	
hashall	Causes bash to remember where commands it has found using **PATH** are located (default).	**set –o hashall**	**set –h**
histappend	Causes bash to append the history list to the file named by **HISTFILE** when the shell exits. By default bash overwrites this file.	**shopt –s histappend**	
histexpand	Turns on the history mechanism (which uses exclamation points by default. Turn this feature off to turn off history expansion.	**set –o histexpand**	**set –H**
history	Enables command history	**set –o history**	
huponexit	Specifies that bash send a SIGHUP signal to all jobs when an interactive login shell exits.	**shopt –s huponexit**	
ignoreeof	Specifies that bash must receive ten EOF characters before it exits. Useful on noisy dial-up lines.	**set –o ignoreeof**	

monitor	Enables job control	**set –o monitor**	set –m
nocaseglob	Causes ambiguous file references to match filenames withoutregard to case (off by default).	**shopt –s nocaseglob**	
noclobber	Helps prevent overwriting files	**set –o noclobber**	set –C
noglob	Disables pathname expansion	**set –o noglob**	set –f
notify	With job control enabled, reports the termination status of background jobs immediately. The default behavior is to display the status just before the next prompt.	**set –o notify**	set –b
nounset	Displays an error and exits from a shell script when you use an unset variable in an interactive shell. The default is to display a null value for an unset variable.	**set –o nounset**	**set –u**
nullglob	Causes bash to expand ambiguous file References that do not match a filename to a null string. By default bash passes these file references without expanding them.	**shopt –s nullglob**	
posix	Runs bash in POSIX mode.	**set –o posix**	
verbose	Displays command lines as bash readsthem.	**set –o verbose**	**set –v**
vi	Specifies vi editing mode for commandline Editing	**set –o vi**	
xpg_echo	Causes the echo builtin to expand backslash escape sequences without the need for the –e option	**shopt –s xpg_echo**	
xtrace	Turns on shell debugging	**set –o xtrace**	**set –x**

8.15 The TC Shell

The TC Shell (tcsh) provides an interface between you and the Linux operating system. The TC Shell is an interactive command interpreter as well as a high-level programming language. You use only one shell at any given time. The TC Shell is an expanded version of the C Shell. The "T" in TC Shell comes from the TENEX and TOPS-20 operating systems. A number of features not found in csh are present in tcsh, including file and username completion, command-line editing, and spelling correction.

Assignment statement

The tcsh assignment statement has the following syntax:set variable = value. Having SPACEs on either side of the equal sign, although illegal in bash, is allowed in tcsh. The default tcsh prompt is a greater than sign (>), but it is frequently set to a single $ character followed by a SPACE.

Shell Scripts

The TC Shell can execute files containing tcsh commands. If the first character of a shell script is a pound sign (#) and the following character is not an exclamation point (!), the TC Shell executes the script under tcsh. If the first character is anything other than #, tcsh calls the sh link to dash or bash to execute the script. The tcsh echo builtin accepts either a –n option or a trailing \c to get rid of the RETURN that echo normally displays at the end of a line.

Checking shell

- **ps**: If you are not sure which shell you are using, use the ps utility to find out. It shows whether you are running tcsh, bash, sh (linked to bash), or possibly another shell.
- **finger**: The finger command followed by your username displays the name of your login shell, which is stored in the /etc/passwd file.

Entering the TC Shell

You can execute tcsh by giving the command tcsh.If you want to use tcsh as a matter of course, you can use the chsh (change shell) utility to change your login shell:

```
bash $ chsh

Changing shell for sam.

Password:

New shell [/bin/bash]: /bin/tcsh

Shell changed.

bash $
```

Leaving the tc shell

You can leave tcsh in several ways. The approach you choose depends on two factors:whether the shell variable ignoreeof is set and whether you are using the shell that you logged in on (your login shell) or another shell that you created after you logged in.If you are not sure how to exit from tcsh, press CONTROL-D on a line by itself with no leading SPACEs, just as you would to terminate standard input to a program. You will either exit or receive instructions on how to exit.If you have not set ignoreeof and it has not been set for you in a startup file, you can exit from any shell by using CONTROL-D (the same procedure you use to exit from the Bourne Again Shell).When ignoreeof is set, CONTROL-D does not work. The ignoreeof variable causes the shell to display a message telling you how to exit. You can always exit from tcsh by giving an exit command. A logout command allows you to exit from your login shell only.

Startup Files

When you log in on the TC Shell, it automatically executes various startup files. You must have read access to a startup file to execute the commands in it.When you log in on the TC Shell, it automatically executes various startup files. You must have read access to a startup file to execute the commands in it.

1) **etc/csh.cshrc and /etc/csh.login:**

The shell first executes the commands in /etc/csh.cshrc and /etc/csh.login. A user working with root privileges can set up these files to establish systemwide default characteristics for tcsh users.They contain systemwide configuration information, such as the default path, the location to check for mail, and so on.

2) **.tcshrc and .cshrc:**

Next the shell looks for ~/.tcshrc or, if it does not exist, ~/.cshrc.You can use these files to establish variables and parameters that are local to your shell. Each time you create a new shell, tcsh reinitializes these variables for the new shell.

3) **.history**

Login shells rebuild the history list from the contents of ~/.history. If the histfile variable exists, tcsh uses the file that histfile points to in place of .history.

4)**.login**

Login shells read and execute the commands in ~/.login. This file contains commands that you want to execute once, at the beginning of each session.

5) **/etc/csh.logout**and **.logout**

The TC Shell runs the /etc/csh.logout and ~/.logout files, in that order, when you exit from a login shell.

Features Common to the Bourne Again and TC Shells

Most of the features common to both bash and tcsh are derived from the original C Shell:

- Command-line expansion (also called substitution)
- History
- Aliases
- Job control
- Filename substitution
- Directory stack manipulation
- Command substitution

Command-Line Expansion (Substitution)

The tcsh man page uses the term substitution instead of expansion; the latter is used by bash. The TC Shell scans each token for possible expansion in the following order:

1. History substitution

2. Alias substitution

3. Variable substitution

4. Command substitution

5. Filename substitution

6. Directory stack substitution

1) History substitution:

The TC Shell assigns a sequential event number to each command line. You can display this event number as part of the tcsh prompt. As in bash, the tcsh history builtin displays the events in your history list. The list of events is ordered with the oldest events at the top. The last event in the history list is the history command that displayed the list. In the following history list, which is limited to ten lines by the argument of 10 to the history command, command 23 modifies the tcsh prompt to display the history event number. The time each command was executed appears to the right of the event number.

```
32 $ history 10

23 23:59 set prompt = "! $ "

24 23:59 ls -l

25 23:59 cat temp

26 0:00 rm temp

27 0:00 vim memo

28 0:00 lpr memo

29 0:00 vim memo

30 0:00 lpr memo

31 0:00 rm memo

32 0:00 history
```

The same event and word designators work in both shells. For example, !! refers to the previous event in tcsh, just as it does in bash. The command !328 executes event number 328; !?txt? executes the most recent event containing the string txt.

- Few tcsh word modifiers not found in bash.

Modifier	Function
u	Converts the first lowercase letter into uppercase
l	Converts the first uppercase letter into lowercase
a	Applies the next modifier globally within a single word

- Variables to control history

Variable	Default	Function
history	100 words	Maximum number of events saved during a session
histfile	**~/.history**	Location of the history file
savehist	not set	Maximum number of events saved between sessions

2) Aliases Substitution

The alias builtin has a slightly different syntax: alias name value. The following command creates an alias for ls: tcsh $ alias ls "ls -lF". The tcsh alias allows you to substitute command-line arguments, whereas bash does not:

```
$ alias nam "echo Hello, \!^ is my name"

$ nam Sam

Hello, Sam is my name
```

The string \!* within an alias expands to all command-line arguments:$ alias sortprint "sort \!* | lpr"

The next alias displays its second argument:$ alias n2 "echo \!:2"

3) Job Control

Job control is similar in both bash and tcsh. You can move commands between the foreground and the background, suspend jobs temporarily, and get a list of the current jobs. The % character references a job when it is followed by a job number or a string prefix that uniquely identifies the job. You will see a minor difference when you run a multiple-process command line in the background from each shell. Whereas bash displays only the PID number of the last background process in each job, tcsh displays the numbers for all processes belonging to a job.

4) Filename Substitution

The TC Shell expands the characters *, ?, and [ ] in a pathname just as bash does. * matches any string of zero or more characters, ? matches any single character, [ ] defines a character class, which is used to match single characters appearing within a pair of brackets. The TC Shell expands command-line arguments that start with a tilde (~) into filenames in much the same way that bash does, with the ~ standing for the user's home directory or the home directory of the user whose name follows the tilde. The bash special expansions ~+ and ~– are not available in tcsh.Brace expansion is available in tcsh. Like tilde expansion, it is regarded as an aspect of filename substitution even though brace expansion can generate strings that are not the names of actual files.

5) Manipulating the Directory Stack

Directory stack manipulation in tcsh does not differ much from that in bash. The dirsbuiltin displays the contents of the stack, and the pushd and popdbuiltins push directories onto and pop directories off of the stack.

6) Command Substitution

The $(...) format for command substitution is not available in tcsh. In its place you must use the original '...' format. Otherwise, the implementation in bash and tcsh is identical.

Summary

- The Bourne Again Shell and TC Shell are command interpreters and high-level programming languages.
- Login shells are, by their nature, interactive.
- Pressing the suspend key (usually CONTROL-Z) immediately suspends (temporarily stops) the job in the foreground and displays a message that includes the word Stopped.
- Keyword shell variables have special meaning to the shell and usually have short, mnemonic names.
- Like the file structure, the process structure is hierarchical, with parents, children, and even a root. A parent process forks a child process, which in turn can fork other processes.
- You can declare a shell function in the ~/.bash_profile startup file, in the script that uses it, or directly from the command line.
- You can remove functions with the unset builtin. The shell does not retain functions after you log out.
- The shopt (shell option) builtin (not available in tcsh) enables, disables, and lists certain bash features that control the behavior of the shell.
- The TC Shell is an interactive command interpreter as well as a high-level programming language.

Keywords

- **BASH_ENV:** Noninteractive shells look for the environment variable BASH_ENV (or ENV if the shell is called as sh) and execute commands in the file named by this variable.
- **File Descriptors:** A file descriptor is the place a program sends its output to and gets its input from.
- **Shell Script:**A shell script is a file that holds commands that the shell can execute. The commands in a shell script can be any commands you can enter in response to a shell prompt.
- **Positional Parameters:**Positional parameters enable you to access command-line arguments, a capability that you will often require when you write shell scripts.
- **MAIL:**The MAIL variable (mail under tcsh) contains the pathname of the file that holds your mail (your mailbox, usually **/var/mail/name**, where name is your username).
- **MAILPATH:**The MAILPATH variable (not available under tcsh) contains a list of filenames separated by colons. If this variable is set, the shell informs you when any one of the files is modified.
- **MAILCHECK:** This variable (not available under tcsh) specifies how often, in seconds, the shell checks for new mail. The default is 60 seconds. If you set this variable to zero, the shell checks before each prompt.
- **Process:**A process is the execution of a command by the Linux kernel.
- **Events:**The history mechanism, a feature adapted from the C Shell, maintains a list of recently issued command lines, also called events.

- **HISTSIZE:** The value of the HISTSIZE variable determines the number of events preserved in the history list during a session. A value in the range of 100 to 1,000 is normal.
- **HISTFILE:** When you exit from the shell, the most recently executed commands are saved in the file whose name is stored in the HISTFILE variable (the default is ~/.bash_history). The next time you start the shell, this file initializes the history list.
- **HISTFILESIZE:** The value of the HISTFILESIZE variable determines the number of lines of history saved in HISTFILE.
- **Alias:**An alias is a usually short name that the shell translates into another usually longer name or complex command.
- **ps**: If you are not sure which shell you are using, use the ps utility to find out. It shows whether you are running tcsh, bash, sh (linked to bash), or possibly another shell.
- **finger**: The finger command followed by your username displays the name of your login shell, which is stored in the /etc/passwd file.

Self Assessment

1. The shoptbuiltin _________________ the features that control the bash.

A. Enables
B. Disables
C. Lists
D. All of the above mentioned

2. The short command line options consists of

A. Hyphen
B. A letter
C. Hyphen followed by a letter
D. None of the above

3. Using ______, we can recall, modify and re-execute previously executed events.

A. Fc builtin
B. Exclamation point commands
C. Readline libraries
D. All of the above mentioned

4. Event designators start with

A. &
B. !
C. @
D. #

5. Which key is used for pathname and command completion?

A. CTRL
B. TAB
C. RETURN

D. SHIFT

6. Which of these variables gives the location of history file?
A. HISTSIZE
B. HISTFILE
C. HISTFILESIZE
D. None of the above

7. Which builtin sets the attributes and values for shell variables?
A. declare
B. typeset
C. Both declare and typeset
D. None of the above mentioned

8. PS4 is ____________
A. Primary prompt
B. Secondary prompt
C. Prompt issued by select
D. Bash debugging symbol

9. What is the naming convention for global variables?
A. Only lowercase letters
B. Only uppercase letters
C. Mixed case letters
D. Only numbers

10. What is the correct syntax for assigning a value to a variable in Bourne Again Shell?
A. VARIABLE=value
B. VARIABLE = value
C. VARIABLE= value
D. VARIABLE =value

11. Which keyword holds the pathname of the working directory?
A. pwd
B. work
C. dir
D. key

12. You can specify arguments when you call a function. Within the function these arguments are available as _____________
A. Special parameters
B. Positional parameters
C. Uni parameters

D. None of the above

13. Which built in is used to make the value of a variable available to the child processes?

A. echo

B. export

C. cat

D. avail

14. What is used to see the parent-child relationship?

A. pstree

B. treeps

C. trpsee

D. eesptr

15. The directory stack implements _________ rule.

A. FIFO

B. LIFO

C. RIRO

D. None of the above

16. To remove a directory from the stack, use the ________ builitin.

A. popd

B. pushd

C. remd

D. None of the above

17. Which variable gives the maximum number of events saved between the session?

A. history

B. histfile

C. savehist

D. None of the above

18. Which of these features are common in bash and tcsh?

A. Aliases

B. Job control

C. Command substitution

D. All of the above mentioned

Answers for Self Assessment

1.	D	2.	C	3.	D	4.	B	5.	B
6.	B	7.	C	8.	D	9.	B	10.	A
11.	A	12.	B	13.	B	14.	A	15.	A
16.	A	17.	C	18.	D				

Review Questions

1) What are startup files? Explain in detail.
2) What are file descriptors? Explain.
3) How can we manipulate a directory stack? Explain the operations.
4) What are shell variables? Explain its types in details.
5) What are keyword variables? Explain with examples.
6) What is a process in Linux? Explain it.
7) What is history feature in Linux? Write its features. Which variables that control history?
8) How can we re-execute and edit commands? Write the different ways to do this.
9) What is an alias? Write its syntax. What is the use of single and double quotation marks in alias? Explain with example.
10) What are shell features? Write bash features.
11) What is TC shell? How can we enter and leave the TC shell? Write its startup files.
12) Write the features common to Bourne again shell and TC shell.

Further Readings

Mark G. Sobell, A Practical Guide to Linux Commands, Editors, and Shell Programming, Second Edition, Prentice Hall, Pearson Education, Inc.

Web Links

https://www.bottomupcs.com/file_descriptors.xhtml

https://www.ibm.com/docs/en/aix/7.2?topic=concepts-shell-features

Unit 09: Programming the Bourne Again Shell

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand the control structures
- Understand the file descriptors
- Know the parameters and variables
- Understand the builtins
- Understand the expressions

Introduction

The programming languages are used for writing the programs. A simple program executes a sequence of statements without any jump or condition. In a programming language, we have various kinds of control structures which basically interrupts the flow of statements based upon conditions. The control flow commands alter the order of execution of commands within a shell script. It specifies the order in which computations are performed.

9.1 Control Structures

There are various control structures:

- if...then,
- for...in,
- while,
- until,
- break,

- Continue
- case statements

if...then

This control structure is used to express the decisions. The if...then control structure has the following syntax:

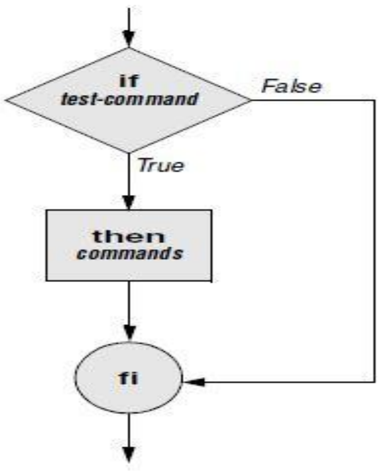
iftest-command

then

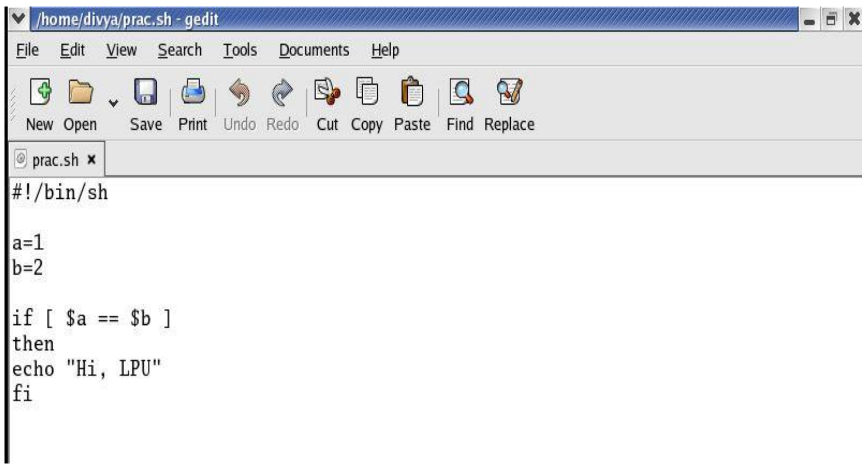
commands

fi

If the statement given in test_command turns out to be true, then the commands given must be executed. In this control structure, there is no command that needs to be printed when the test_command turns out to be false. The flow of if … then control structure is shown below:

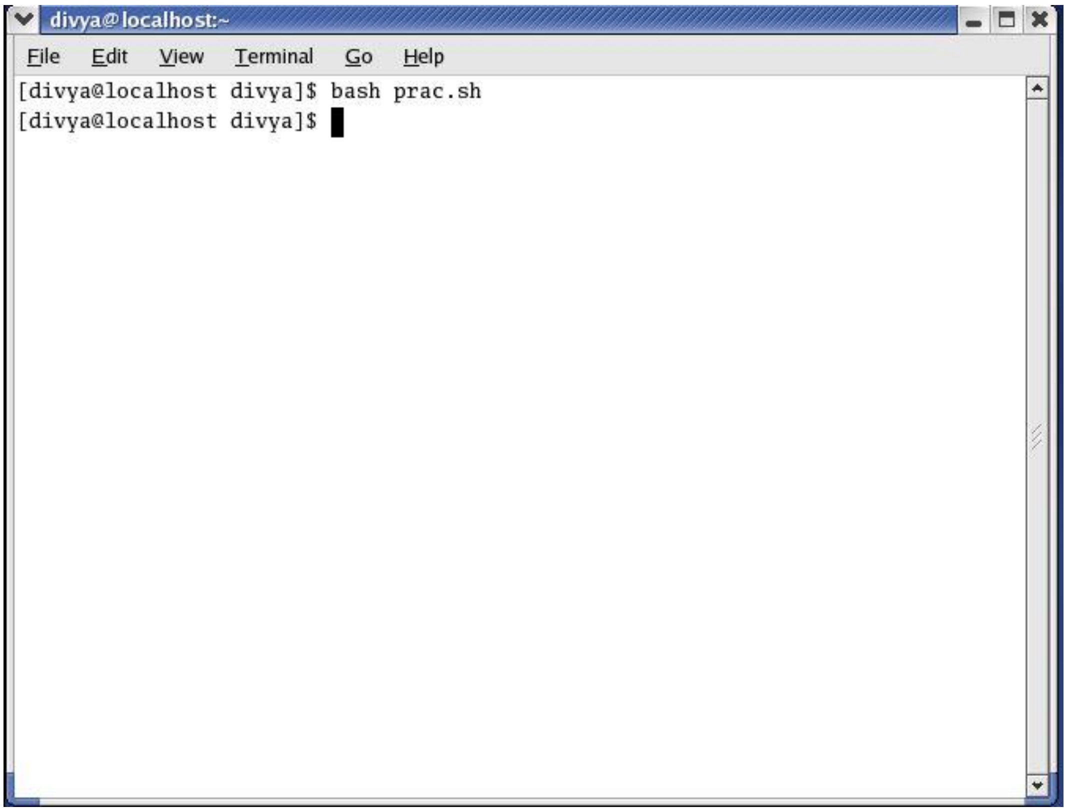


Here, one example is taken in which the value of a is 1 and b is 2 and the test condition is to check whether both values are equal or not. If the condition is true, then only the statement "Hi, LPU" will be printed. Otherwise, the control will exit. The control structure is ended using fi.


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/sh

a=1
b=2

if [ $a == $b ]
then
echo "Hi, LPU"
fi
```

The output of the program is " ". It will print nothing as the condition is false here. The control will just exit.



```
[divya@localhost divya]$ bash prac.sh
[divya@localhost divya]$ 
```

Task: Write a shell script to display square of a number

if...then...else

This control structure is also used to represent the decisions. Here the else part can be optional. The test-command will be evaluated. If this turns out to be true, then commands after that will be printed. If the test-commands turns out to be false, then the statements given in else will be printed. The **if...then...else control structure** has the following syntax:

if ***test-command***

then

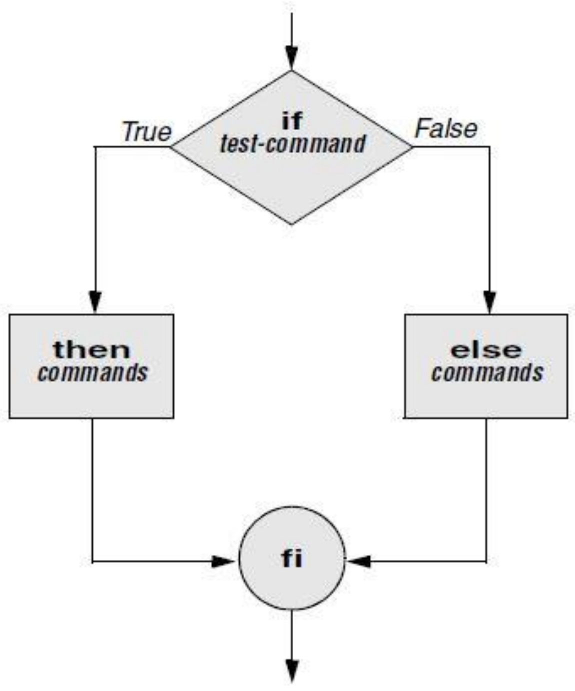
commands

else

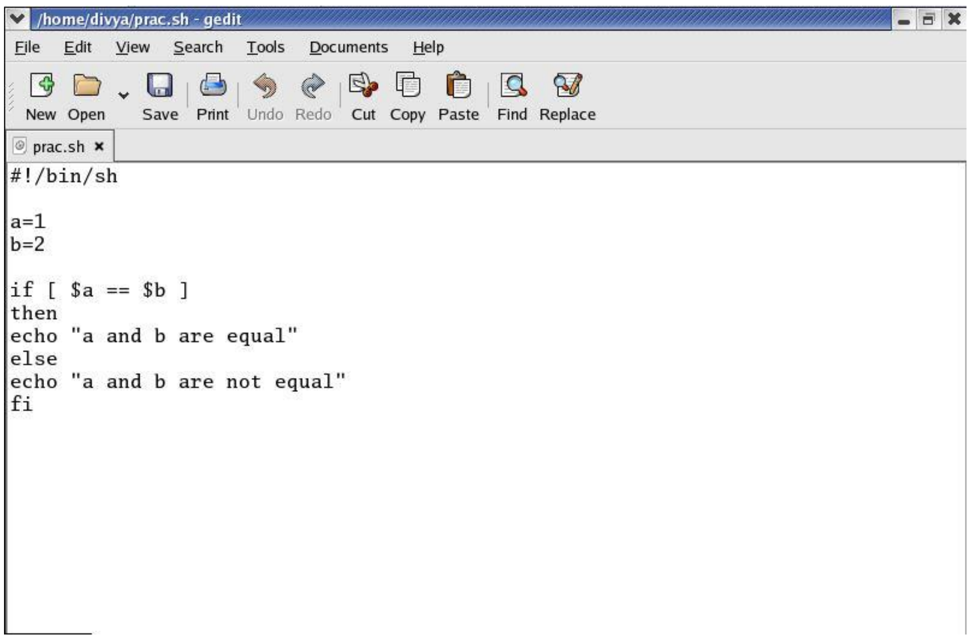
commands

fi

The flow of if… then… else is shown below:

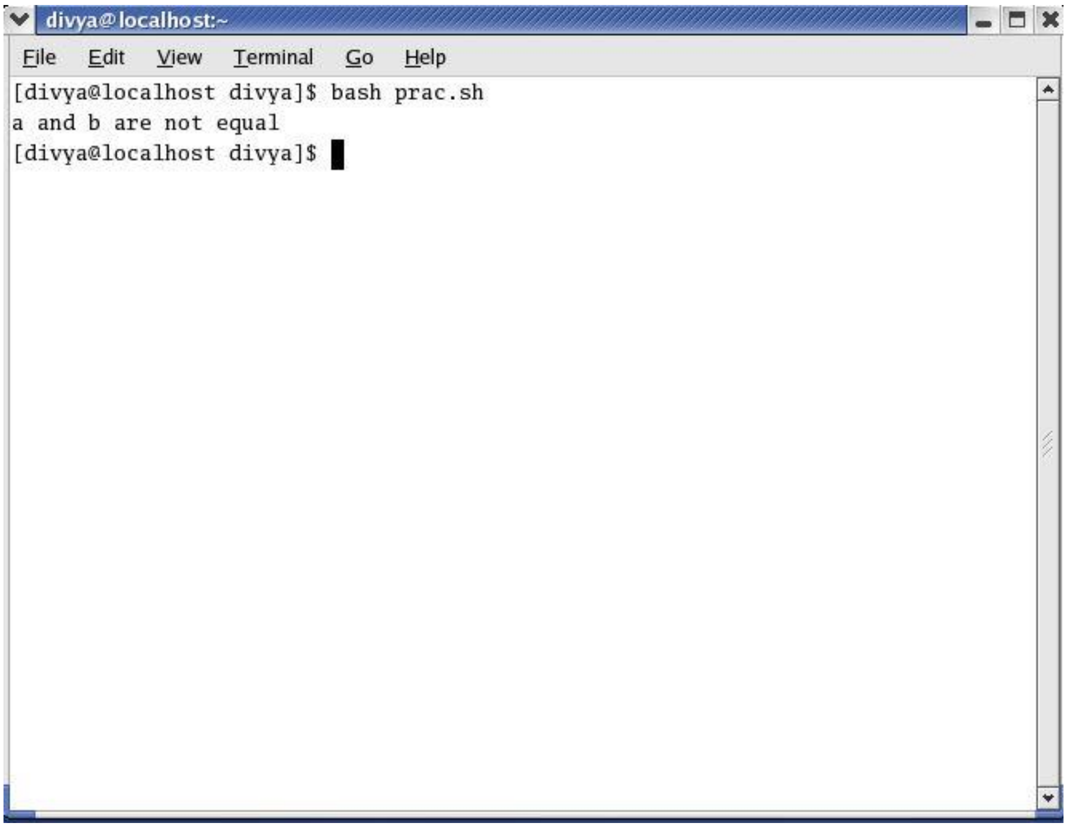


Here in this example, two variables are taken, i.e., a and b. If the values of a and b are equal, then the statement "a and b are equal" otherwise "a and b are not equal" is printed. The structure is ended with fi.


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/sh

a=1
b=2

if [ $a == $b ]
then
echo "a and b are equal"
else
echo "a and b are not equal"
fi
```

The output of the program is: The value of a is 1 and b is 2. It will print "a and b are not equal".



```
[divya@localhost divya]$ bash prac.sh
a and b are not equal
[divya@localhost divya]$
```

Task: Write a shell script to check whether a person is eligible to vote or not.

if...then...elif

This sequence of if statements is the most general way of writing a multi-way decision. The expressions are evaluated in order; if an expression is true, the statement associated with it is executed, and this terminates the whole chain. As always, the code for each statement is either a single statement, or a group of them in braces. The last else part handles the ``none of the above'' or default case where none of the other conditions is satisfied.The **if...then...elif control structure *has the following*** syntax:

if ***test-command***

then

commands

elif ***test-command***

then

commands

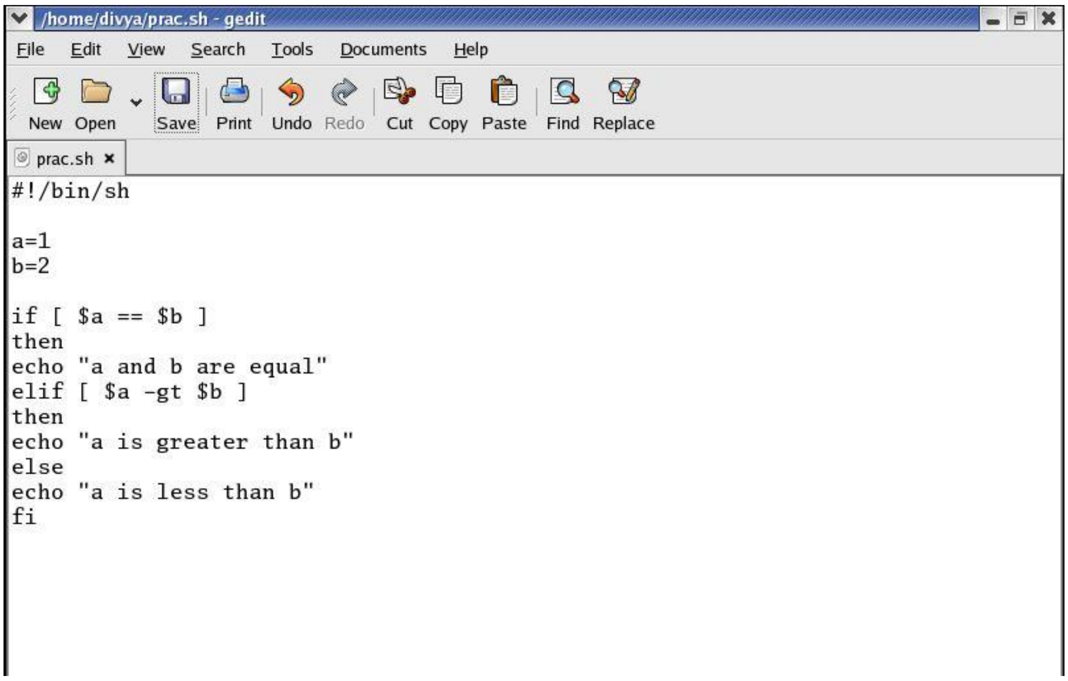
. . .

else

commands

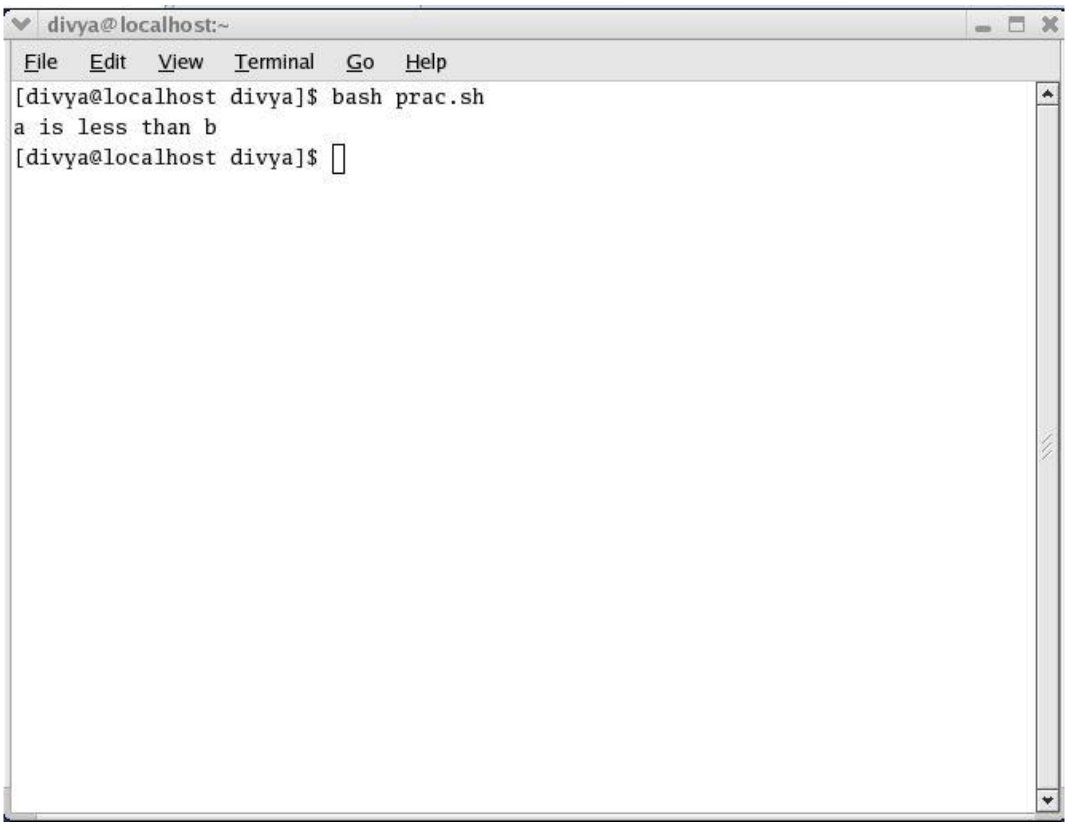
fi

The values of a and b are taken. The condition is to check the value of and b. If both are equal, then it should print "a and b are equal", If not, it should check if a is greater than b, if yes, then it should print "a is greater than b". Otherwise "a is less than b".


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/sh

a=1
b=2

if [ $a == $b ]
then
echo "a and b are equal"
elif [ $a -gt $b ]
then
echo "a is greater than b"
else
echo "a is less than b"
fi
```

The value of a is 1 and b is 2. So, the output of the program is "a is less than b".



```
[divya@localhost divya]$ bash prac.sh
a is less than b
[divya@localhost divya]$ 
```

Task: Write a shell script to display grades as per the following ranges of %age:

>= 80 'A+'

>=60 &&<80 'A'

> = 50 &&< 60 'B'

> = 40 &&<50 'C'

<40 'E'.

for...in

A loop is a sequence of instructions that is continually repeated until a certain condition is reached. It is a block of code that will repeat repeatedly. The for loop operates on lists of items. It repeats a set of commands for every item in a list. The for...in control structure *has the following syntax:*

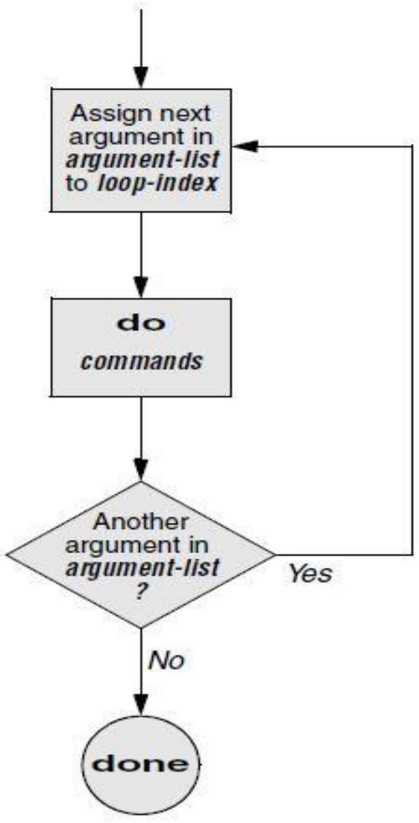
for ***loop-index in argument-list***

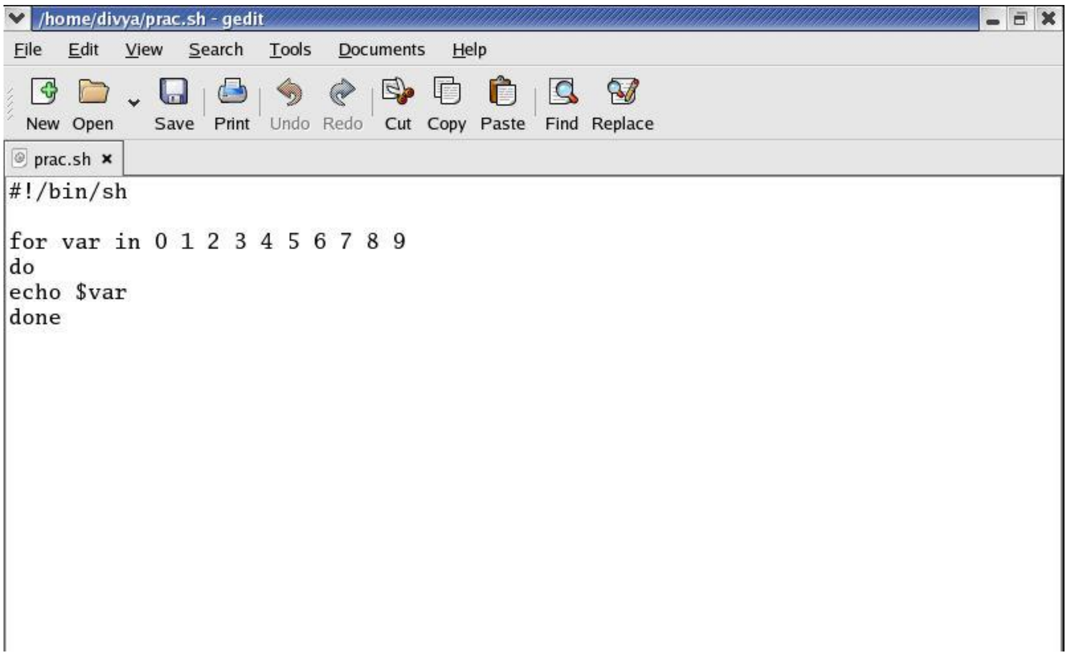
do

commands

done

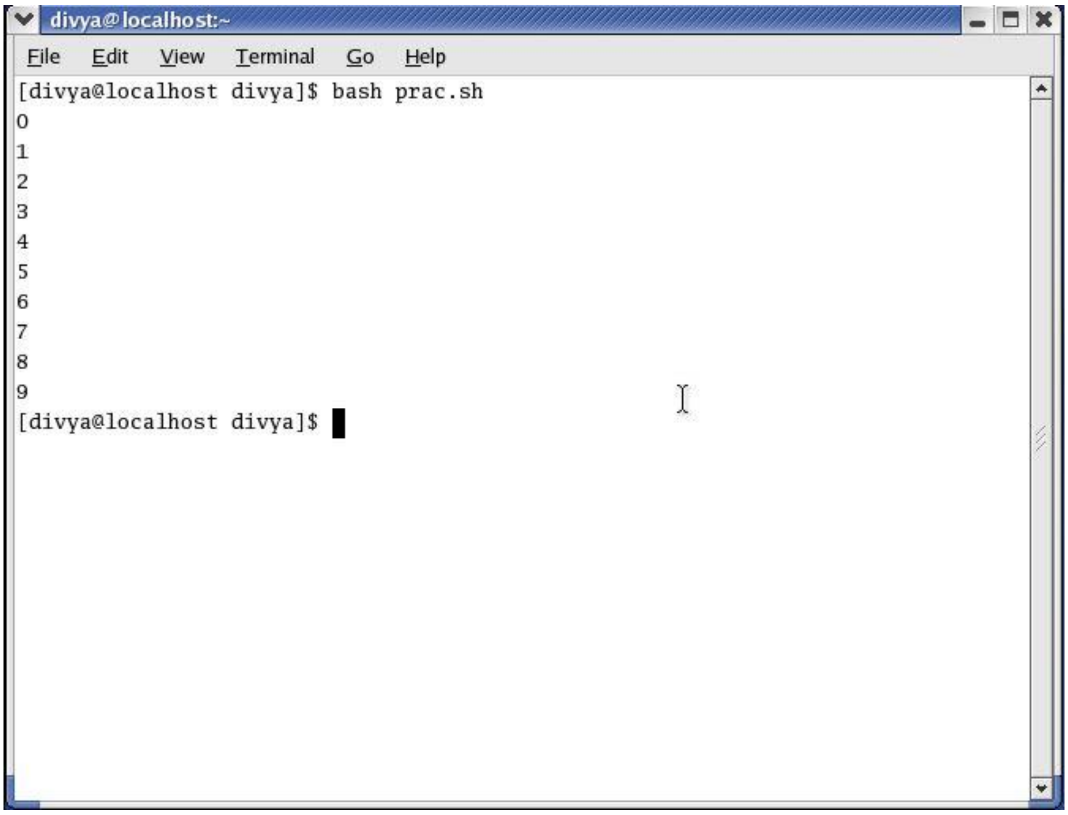
Here loop index is the variable you specify in the do section and will contain the item in the loop that you are on. The list of arguments can be anything that returns a space or newline separated list.


Assign next argument in argument-list to loop-index
do
commands
Another argument in argument-list ?
Yes
No
done


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/sh

for var in 0 1 2 3 4 5 6 7 8 9
do
echo $var
done
```

The output of the program is printing of numbers starting from 0 till 9.



```
[divya@localhost divya]$ bash prac.sh
0
1
2
3
4
5
6
7
8
9
[divya@localhost divya]$
```

Task: Write a shell script to sum of all numbers starting from 1 to 100 using for loop

While

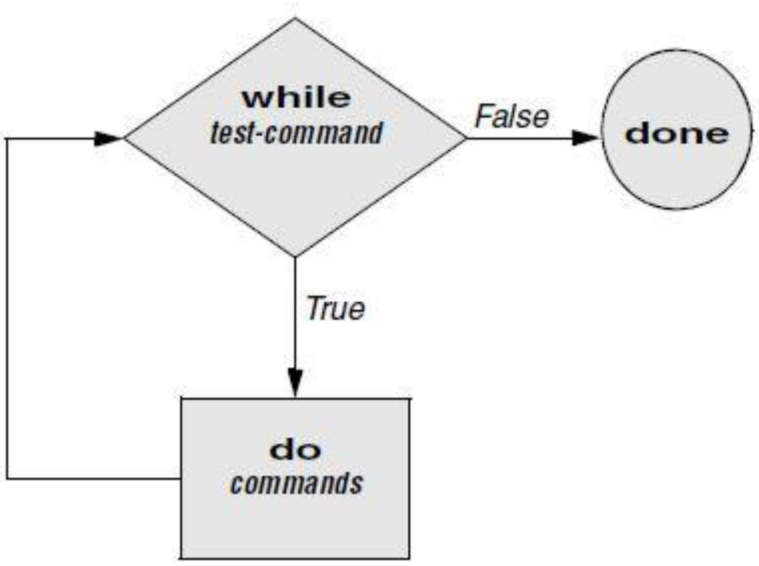
As long as the test-command (Figure 10-5) returns a true exit status, the while structure continues to execute the series of commands delimited by the do and done statements. Before each loop through the commands, the structure executes the test command. When the exit status of the test-command is false, the structure passes control to the statement after the done statement. The while control structure (not available in tcsh) has the following syntax:

while ***test-command***

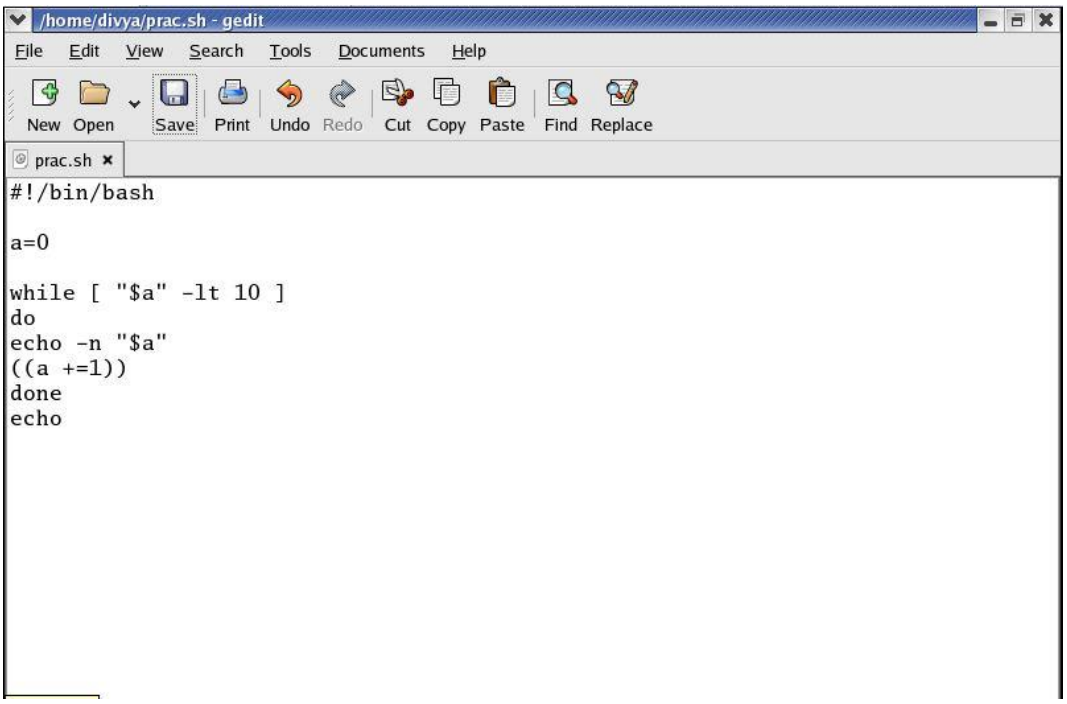
do

commands

done

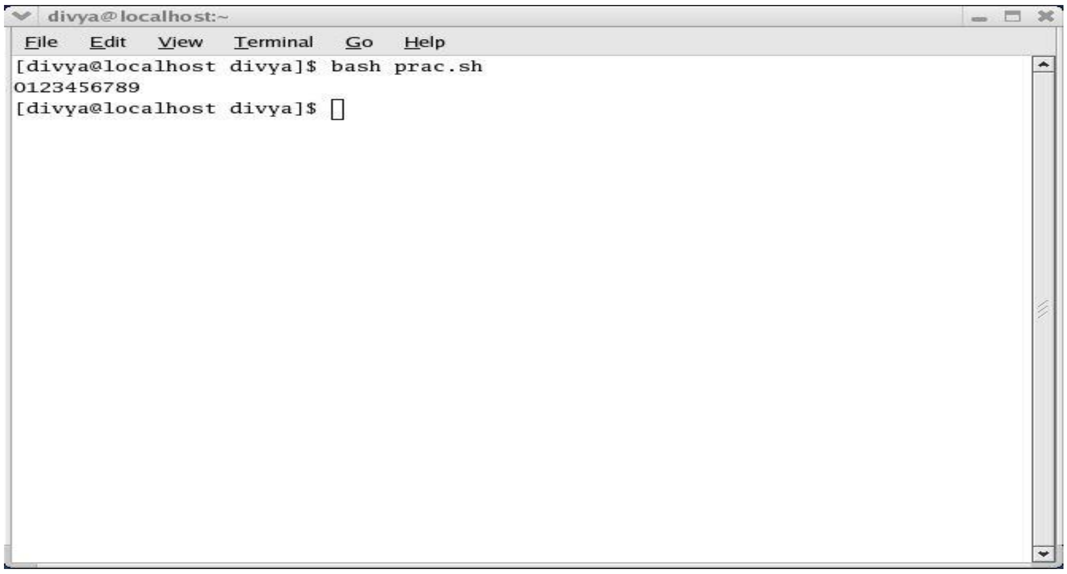


The variable a is taken and initialized with 0. Till the value of a is less than 10, it keeps on printing and incrementing the values.


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/bash

a=0

while [ "$a" -lt 10 ]
do
echo -n "$a"
((a +=1))
done
echo
```

The output of the program is: The values starting from 0 till 9 will be printed.



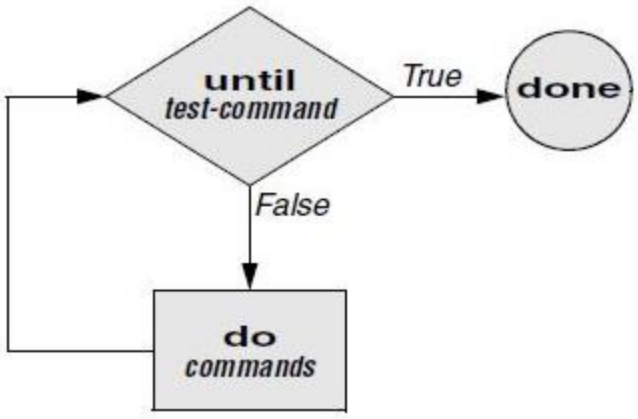
```
[divya@localhost divya]$ bash prac.sh
0123456789
[divya@localhost divya]$ 
```

Task: Write a shell script to count odd numbers from 10 to 100.

Until

The until continues to loop until the test-command returns a true exit status. The while loop is perfect for a situation where you need to execute a set of commands while some condition is true. Sometimes you need to execute a set of commands until a condition is true. The syntax of until is:

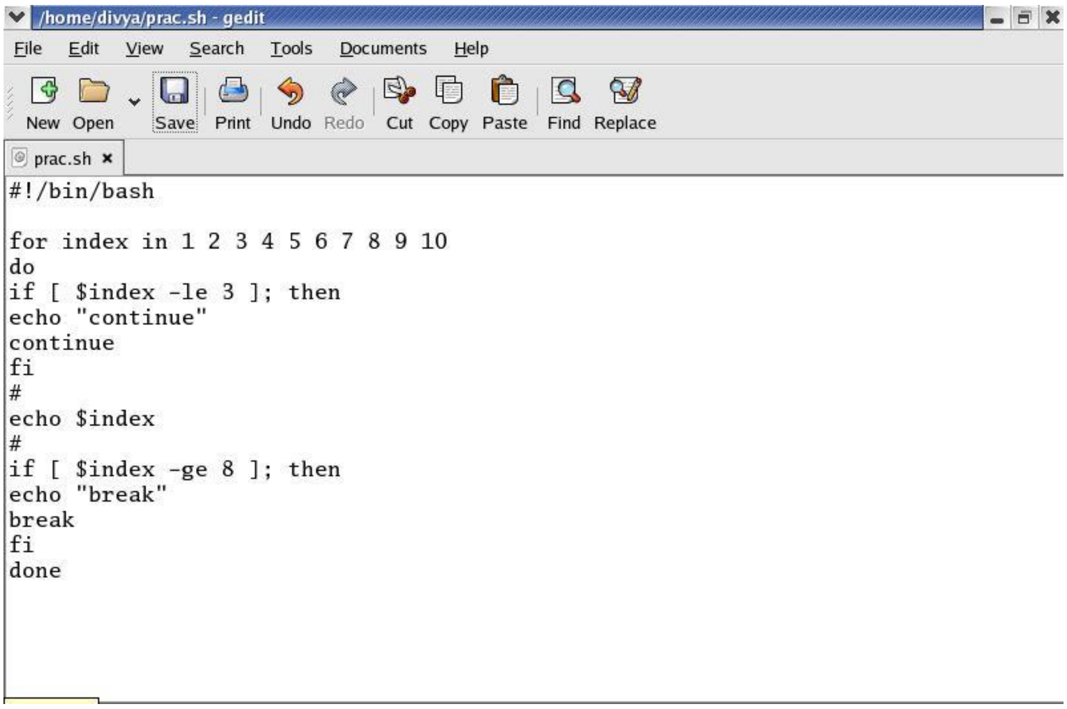
```
until command
do
Statement(s) to be executed until command is true
done
```



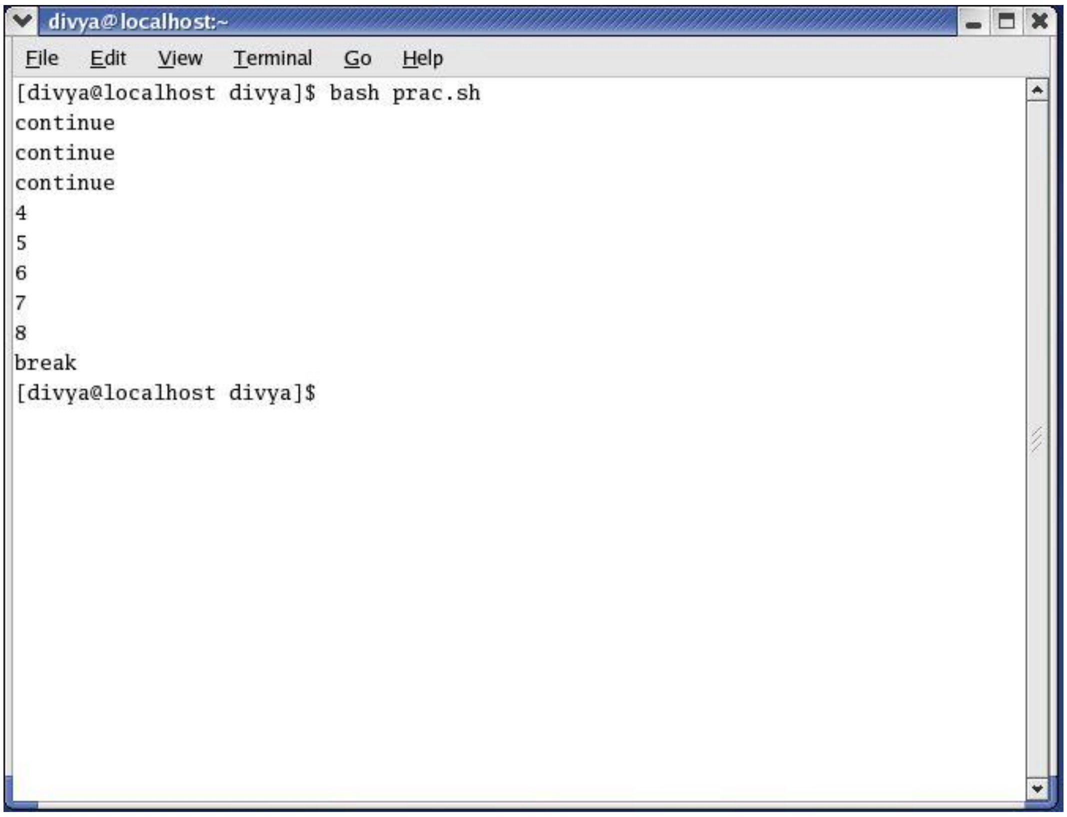
Task: Write a shell script to take input for a number and keep on counting the chances how many times user has not entered a valid number. 'Valid number is -99'.

break and continue

You can interrupt a for, while, or until loop by using a break or continue statement.The break statement transfers control to the statement after the done statement, thereby terminating execution of the loop. The continue command transfers control to the done statement, continuing execution of the loop.


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/bash

for index in 1 2 3 4 5 6 7 8 9 10
do
if [ $index -le 3 ]; then
echo "continue"
continue
fi
#
echo $index
#
if [ $index -ge 8 ]; then
echo "break"
break
fi
done
```



```
[divya@localhost divya]$ bash prac.sh
continue
continue
continue
4
5
6
7
8
break
[divya@localhost divya]$
```

Task: Write a shell script to display square of all numbers from m to n when first multiple of 10 is reached loop should terminate.

Case

You can use multiple **if...elif** statements to perform a multi way branch. However, this is not always the best solution, especially when all the branches depend on the value of a single variable. It supports **case...esac** statement which handles exactly this situation, and it does so more efficiently than repeated **if...elif** statements. The case structure is a multiple-branch decision mechanism. The path taken through the structure depends on a match or lack of a match between the test-string and one of the patterns. The pattern in the case structure is analogous to an ambiguous file reference. It can ne any special character like *, ?, […] or |.The case control structure has the following syntax:

case ***test-string in***

pattern-1)

commands-1

;;

pattern-2)

commands-2

;;

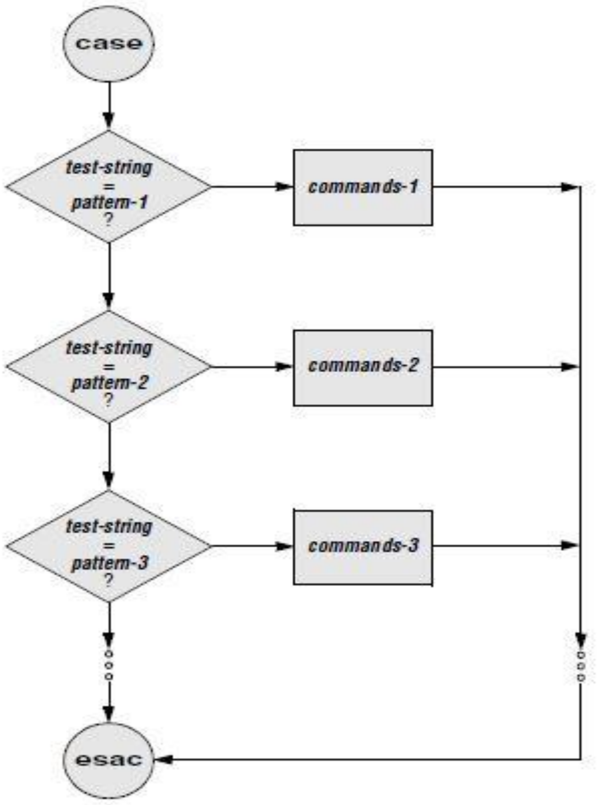
pattern-3)

commands-3

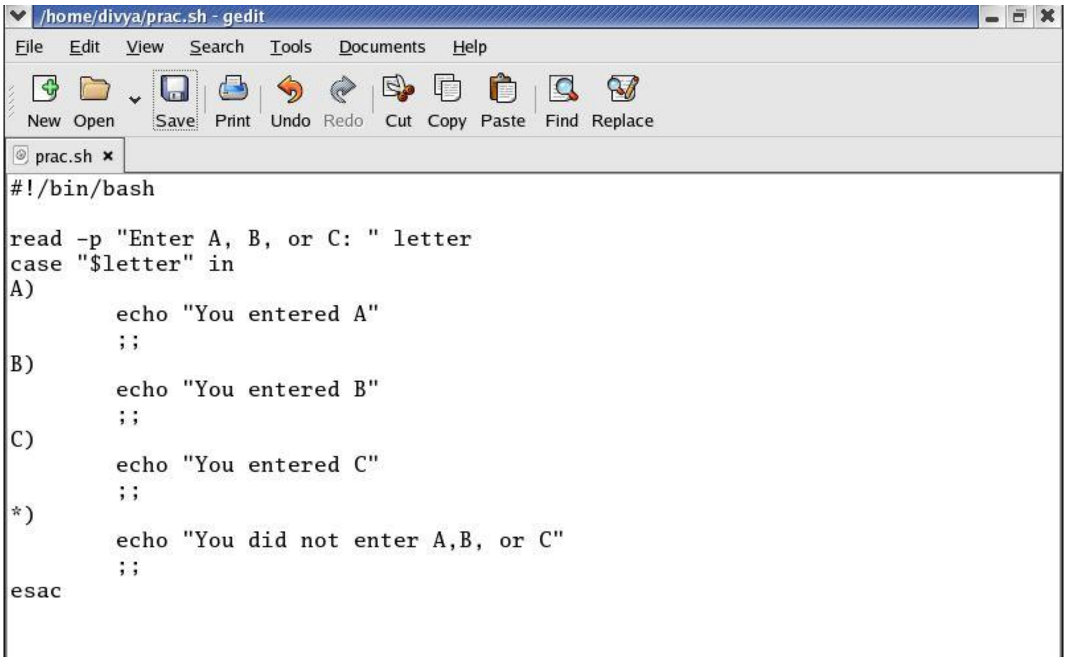
;;

. . .

esac

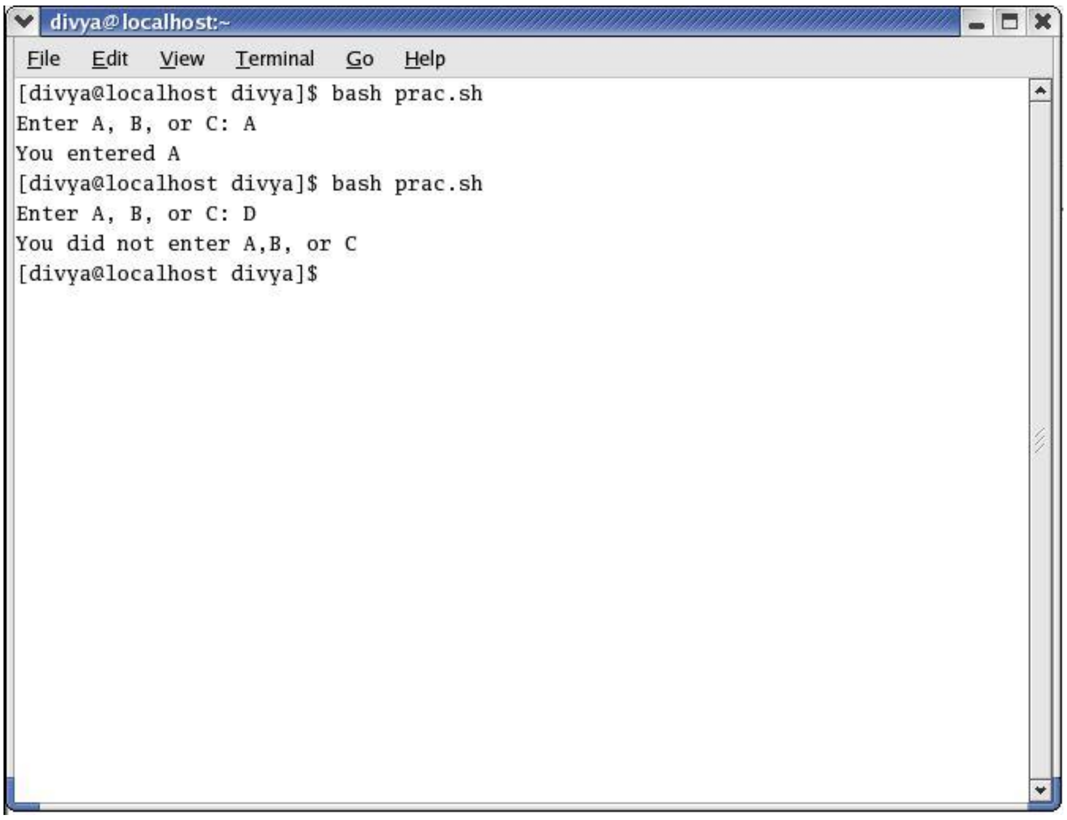

case
test-string = pattern-1 ?
commands-1
test-string = pattern-2 ?
commands-2
test-string = pattern-3 ?
commands-3
esac

The next program asks to enter any character: A,B or C. If you have entered A, it prints "You entered A". If you have entered B, it prints " You entered B". If you have entered C, it prints " You entered C". If any other character is entered, it prints " You did not enter A, B, or C". The structure ends with esac.


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/bash

read -p "Enter A, B, or C: " letter
case "$letter" in
A)
        echo "You entered A"
        ;;
B)
        echo "You entered B"
        ;;
C)
        echo "You entered C"
        ;;
*)
        echo "You did not enter A,B, or C"
        ;;
esac
```

The output of the program is:



```
[divya@localhost divya]$ bash prac.sh
Enter A, B, or C: A
You entered A
[divya@localhost divya]$ bash prac.sh
Enter A, B, or C: D
You did not enter A,B, or C
[divya@localhost divya]$
```

Task: Write a shell script to implement arithmetic calculator.

Select

The select control structure is based on the one found in the Korn Shell. It displays a menu, assigns a value to a variable based on the user's choice of items, and executes a series of commands. The select control structure has the following syntax:

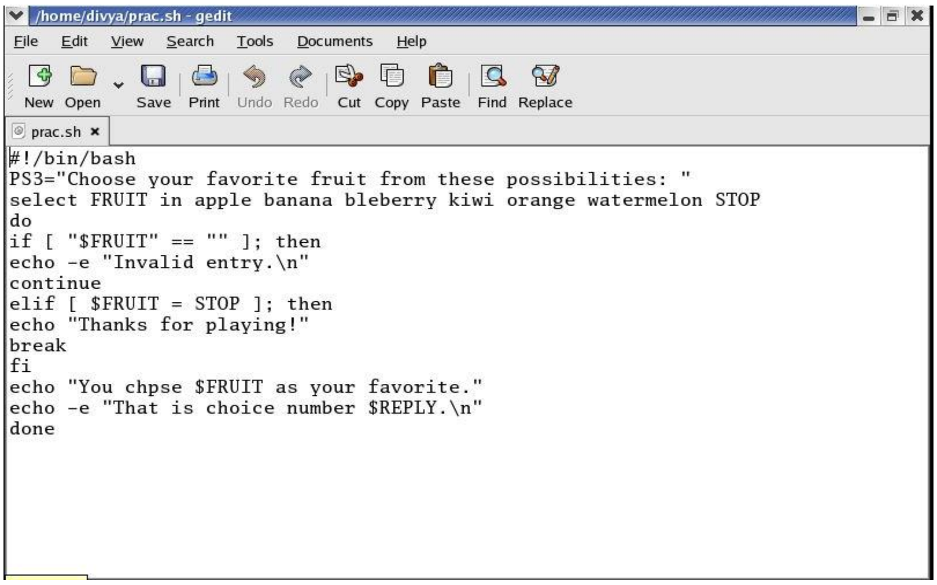
select ***varname [in arg . . .]***

do

commands

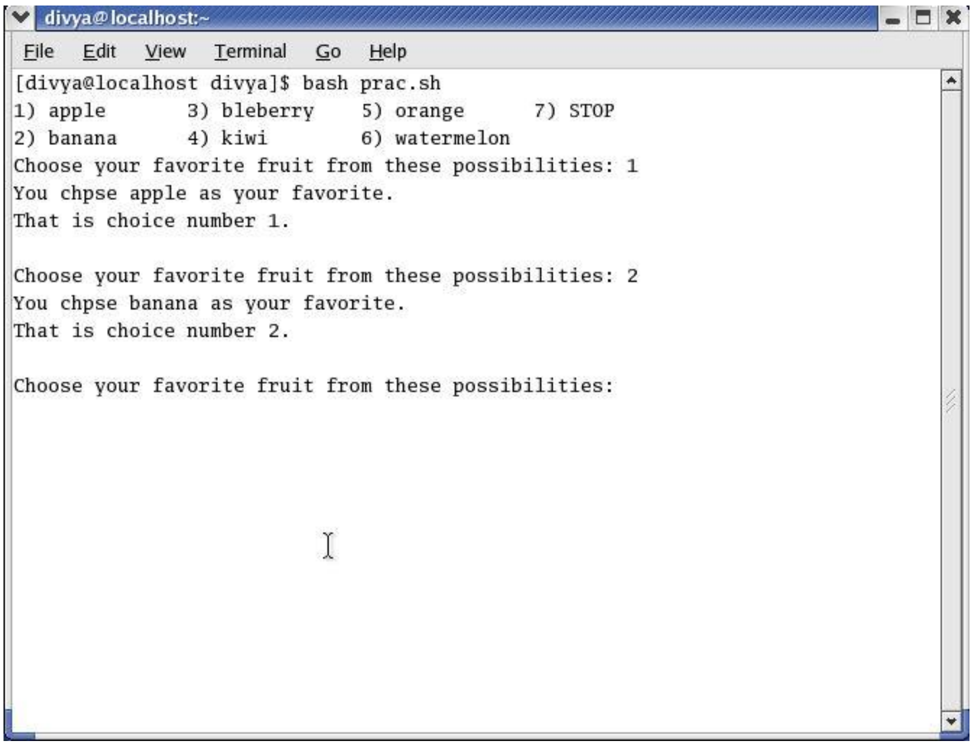
done

The select structure displays a menu of the arg items. If you omit the keyword in and the list of arguments, select uses the positional parameters in place of the arg items. The menu is formatted with numbers before each item.


/home/divya/prac.sh - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
prac.sh

```
#!/bin/bash
PS3="Choose your favorite fruit from these possibilities: "
select FRUIT in apple banana bleberry kiwi orange watermelon STOP
do
if [ "$FRUIT" == "" ]; then
echo -e "Invalid entry.\n"
continue
elif [ $FRUIT = STOP ]; then
echo "Thanks for playing!"
break
fi
echo "You chpse $FRUIT as your favorite."
echo -e "That is choice number $REPLY.\n"
done
```

The output of the program is:



```
[divya@localhost divya]$ bash prac.sh
1) apple      3) bleberry    5) orange      7) STOP
2) banana     4) kiwi        6) watermelon
Choose your favorite fruit from these possibilities: 1
You chpse apple as your favorite.
That is choice number 1.

Choose your favorite fruit from these possibilities: 2
You chpse banana as your favorite.
That is choice number 2.

Choose your favorite fruit from these possibilities:
```

9.2 File Descriptor

Before a process can read from or write to a file, it must open that file. When a process opens a file, Linux associates a number (called a file descriptor) with the file. A file descriptor is an index into the process's table of open files. Each process has its own set of open files and its own file descriptors. After opening a file, a process reads from and writes to that file by referring to its file descriptor. When it no longer needs the file, the process closes the file, freeing the file descriptor. A typical Linux process starts with three open files: standard input (file descriptor 0), standard output (file descriptor 1), and standard error (file descriptor 2). Often these are the only files the process needs.

Opening a file descriptor

The Bourne Again Shell opens files using the exec built in as follows: exec n> outfileopens outfile for output and holds it open, associating it with file descriptor n. The exec m< infileopens infile for input and holds it open, associating it with file descriptor m.

Duplicating a file descriptor

The <& token duplicates an input file descriptor; >& duplicates an output file descriptor. You can duplicate a file descriptor by making it refer to the same file as another open file descriptor, such as standard input or output. To open or redirect file descriptor n as a duplicate of file descriptor m: exec n<&m

Once you have opened a file, you can use it for input and output in two ways. First, you can use I/O redirection on any command line, redirecting standard output to a file descriptor with >&n or redirecting standard input from a file descriptor with <&n. Second, you can use the read and echo builtins. If you invoke other commands, including functions, they inherit these open files and file descriptors. When you have finished using a file, you can close it using: exec n<&–

9.3 Parameters and Variables

There are various parameters and variables which are used. These are:

- Array Variables
- Locality of Variables
- Functions
- Special Parameters
- Positional Parameters

Array Variables

The Bourne Again Shell supports one-dimensional array variables. The subscripts are integers with zero-based indexing (i.e., the first element of the array has the subscript 0). The declaration and assignment of values to an array can be done as:name=(element1 element2 ...). An example of assigning four values to the array NAMES can be done as:$ NAMES=(max helen sam zach). It references a single element of an array as follows:$ echo ${NAMES[2]}

```
sam
```

The declare builtin with the –a option displays the values of the arrays (and reminds you that bash uses zero based indexing for arrays):

```
$ A=("${NAMES[*]}")
$ B=("${NAMES[@]}")
$ declare -a
declare -a A='([0]="max helen sam zach")'
declare -a B='([0]="max" [1]="helen" [2]="sam" [3]="zach")'
```

```
...
declare -a NAMES='([0]="max" [1]="helen" [2]="sam" [3]="zach")'
```

Locality of Variables

By default, variables are local to the process in which they are declared. Thus, a shell script does not have access to variables declared in your login shell unless you explicitly make the variables available (global). Under bash, export makes a variable available to child processes. Under tcsh, setenv assigns a value to a variable and makes it available to child processes.

Locality of Variables(Without export)

```
$ cat extest1
cheese=american
echo "extest1 1: $cheese"
subtest
echo "extest1 2: $cheese"
$ cat subtest
echo "subtest 1: $cheese"
cheese=swiss
echo "subtest 2: $cheese"
$ ./extest1
extest1 1: american
subtest 1:
subtest 2: swiss
extest1 2: American
```

Locality of Variables(With export)

```
$ cat extest2
export cheese=american
echo "extest2 1: $cheese"
subtest
echo "extest2 2: $cheese"
$ ./extest2
extest2 1: american
subtest 1: american
subtest 2: swiss
extest2 2: American
```

An export builtin can optionally include an assignment:export cheese=American. The preceding statement is equivalent to the following two statements:

```
cheese=american
export cheese
```

An export builtin can optionally include an assignment:export cheese=american

The preceding statement is equivalent to the following two statements:

```
cheese=american
export cheese
```

Functions

Because functions run in the same environment as the shell that calls them, variables are implicitly shared by a shell and a function it calls.

```
$ function nam () {
>echo $myname
>myname=zach
>}
$ myname=sam
$ nam
sam
$ echo $myname
zach
```

The myname variable is set to sam in the interactive shell. The nam function then displays the value of myname (sam) and sets myname to zach. The final echo shows that, in the interactive shell, the value of myname has been changed to zach. Local variables are helpful in a function written for general use. Because the function is called by many scripts that may be written by different programmers, you need to make sure the names of the variables used within the function do not conflict with (i.e., duplicate) the names of the variables in the programs that call the function. Local variables eliminate this problem. When used within a function, the typeset builtin declares a variable to be local to the function it is defined in.

Special parameters

Special parameters enable you to access useful values pertaining to command-line arguments and the execution of shell commands. You reference a shell special parameter by preceding a special character with a dollar sign ($).As with positional parameters, it is not possible to modify the value of a special parameter by assignment.

$$: PID Number

The shell stores in the $$ parameter the PID number of the process that is executing it. In this example, echo displays the value of this variable and the ps utility confirms its value. Both commands show that the shell has a PID number of 5209:

```
$ echo $$
5209
$ ps
PID TTY TIME CMD
5209 pts/1 00:00:00 bash
6015 pts/1 00:00:00 ps
```

Because echo is built into the shell, the shell does not create another process when you give an echo command. However, the results are the same whether echo is a built in or not, because the shell substitutes the value of $$ before it forks a new process to run a command. Incorporating a PID number in a filename is useful for creating unique file names when the meanings of the names do not matter; this technique is often used in shell scripts for creating names of temporary files. When two people are running the same shell script, having unique filenames keeps the users from inadvertently sharing the same temporary file.

$?: Exit Status

When a process stops executing for any reason, it returns an exit status to its parent process. The exit status is also referred to as a condition code or a return code. The $? ($status under tcsh) variable stores the exit status of the last command. By convention a nonzero exit status represents a false value and means the command failed. A zero is true and indicates the command executed successfully. The first ls command succeeds and the second fails, as demonstrated by the exit status:

```
$ ls es

es

$ echo $?

0

$ ls xxx

ls: xxx: No such file or directory

$ echo $?

1
```

You can specify the exit status that a shell script returns by using the exit builtin, followed by a number, to terminate the script. If you do not use exit with a number to terminatea script, the exit status of the script is that of the last command the script ran.

```
$ cat es
echo This program returns an exit status of 7.
exit 7
$ es
This program returns an exit status of 7.
$ echo $?
7
$ echo $?

0
```

Positional Parameters

Positional parameters comprise the command name and command-line arguments. These parameters are called positional because within a shell script, you refer to them by their position on the command line. Only the **set** builtin allows you to change the values of positional parameters. However, you cannot change the value of the command name from within a script. The tcsh set builtin does not change the values of positional parameters.

$#: Number of Command-Line Arguments

The $# parameter holds the number of arguments on the command line (positional parameters), not counting the command itself:

```
$ cat num_args

echo "This script was called with $# arguments."

$ ./num_args sam max zach

This script was called with 3 arguments.
```

$0: Name of the Calling Program

The shell stores the name of the command you used to call a program in parameter $0. This parameter is numbered zero because it appears before the first argument on the command line:

```
$ cat abc
echo "The command used to run this script is $0"
$ ./abc
The command used to run this script is ./abc
$ ~sam/abc
The command used to run this script is /home/sam/abc
```

$1–$n: Command-Line Arguments

The first argument on the command line is represented by parameter $1, the second argument by $2, and so on up to $n. For values of n greater than 9, the number must be enclosed within braces. For example, the twelfth command-line argument is represented by ${12}. The following script displays positional parameters that hold command-line arguments:

```
$ cat display_5args
echo First 5 arguments are $1 $2 $3 $4 $5

$ ./display_5args zach max helen
First 5 arguments are zach max helen
```

shift: Promotes Command-Line Arguments

The shift builtin promotes each command-line argument. The first argument (which was $1) is discarded. The second argument (which was $2) becomes the first argument (now $1), the third becomes the second, and so on. Because no "unshift" command exists, you cannot bring back arguments that have been discarded. An optional argument to shift specifies the number of positions to shift (and the number of arguments to discard); the default is 1.

The following demo_shift script is called with three arguments. Double quotation marks around the arguments to echo preserve the spacing of the output. The program displays the arguments and shifts them repeatedly until no more arguments are left to shift:

```
$ cat demo_shift
echo "arg1= $1 arg2= $2 arg3= $3"
shift
echo "arg1= $1 arg2= $2 arg3= $3"
shift
echo "arg1= $1 arg2= $2 arg3= $3"
shift
echo "arg1= $1 arg2= $2 arg3= $3"
shift
$ ./demo_shift alice helen zach
arg1= alice arg2= helen arg3= zach
arg1= helen arg2= zach arg3=
arg1= zach arg2= arg3=
arg1= arg2= arg3=
```

set: Initializes Command-Line Arguments

When you call the set builtin with one or more arguments, it assigns the values of the arguments to the positional parameters, starting with $1 (not available in tcsh). The following script uses set to assign values to the positional parameters $1, $2, and $3:

```
$ cat set_it
set this is it
echo $3 $2 $1
$ ./set_it
it is this
```

$* and $@: Represent All Command-Line Arguments

The $* parameter represents all command-line arguments, as the display_all program demonstrates:

```
$ cat display_all
echo All arguments are $*

$ ./display_all a b c d e f g h i j k l m n o p
All arguments are a b c d e f g h i j k l m n o p
```

9.4 Builtin Commands

Builtin commandsdo not fork a new processwhen you execute them.

Builtins	Funcexittions
:	Returns 0 or *true*
.	Executes a shell script as part of the current process
bg	Puts a suspended job in the background
break	Exits from a looping control structure
cd	Changes to another working directory
continue	Starts with the next iteration of a looping control structure

echo	Displays its arguments
eval	Scans and evaluates the command line
exec	Executes a shell script or program in place of the current process
export	Exits from the current shell
fg	Brings a job from the background into the foreground
getopts	Parses arguments to a shell script
jobs	Displays a list of background jobs
kill	Sends a signal to a process or job
pwd	Displays the name of the working directory
read	Reads a line from standard input
readonly	Declares a variable to be readonly
set	Sets shell flags or command-line argument variables; with no argument, lists all variables
shift	Promotes each command-line argument
test	Compares arguments
times	Displays total times for the current shell and its children
trap	Traps a signal
type	Displays how each argument would be interpreted as a command

umask	Returns the value of the file-creation mask
unset	Removes a variable or function
wait	Waits for a background process to terminate

9.5 Expressions

An expression comprises constants, variables, and operators that the shell can process to return a value. It contains arithmetic, logical and conditional expressions and operators.

Arithmetic Evaluation

The Bourne Again Shell can perform arithmetic assignments and evaluate many different types of arithmetic expressions, all using integers. The shell performs arithmetic assignments in a number of ways.

One is with arguments to the let builtin:$ **let "VALUE=VALUE * 10 + NEW".** Within a let statement you do not need to use dollar signs ($) in front of variable names. Double quotation marks must enclose a single argument, or expression, that contains SPACEs. Because most expressions contain SPACEs and need to be quoted, bash accepts ((expression)) as a synonym for let "expression", obviating the need for both quotation marks and dollar signs: $ ((VALUE=VALUE * 10 + NEW))

Logical expressions

You can use the ((expression)) syntax for logical expressions, although that task is frequently left to [[expression]].

```
$ cat age2
#!/bin/bash
echo -n "How old are you? "
read age
if ((30 < age && age < 60)); then
echo "Wow, in $((60-age)) years, you'll be 60!"
else
echo "You are too young or too old to play."
fi
$ ./age2
How old are you? 25
You are too young or too old to play.
```

Logical Evaluation (Conditional expressions)

The syntax of a conditional expression is [[expression]]where expression is a Boolean (logical) expression. You must precede a variable name with a dollar sign ($) within expression. The result of executing this builtin, as with the test builtin, is a return status. The conditions allowed within the brackets are almost a superset of those accepted by test. Where the test builtin uses –a as a

Boolean AND operator, [[expression]] uses &&. Similarly, where test uses –o as a Boolean OR operator, [[expression]] uses | |.

String comparisons

The test builtin tests whether strings are equal. The [[expression]] syntax adds comparison tests for string operators. The > and < operators compare strings for order (for example, "aa" < "bbb"). The = operator tests for pattern match, not just equality: [[string = pattern]] is true if string matches pattern. This operator is not symmetrical; the pattern must appear on the right side of the equal sign. For example,

[[artist = a*]] is true (= 0), whereas [[a* = artist]] is false (= 1):

```
$ [[ artist = a* ]]

$ echo $?

0

$ [[ a* = artist ]]

$ echo $?

1
```

String Pattern Matching

The Bourne Again Shell provides string pattern-matching operators that can manipulate pathnames and other strings. These operators can delete from strings prefixes or suffixes that match patterns.

Operator	Function
#	Removes minimal matching prefixes
##	Removes maximal matching prefixes
%	Removes minimal matching suffixes
%%	Removes maximal matching suffixes

The syntax for these operators is: **${varname op pattern}.**

where op is one of the operators and pattern is a match pattern like that used for filename generation.

9.6 Operators

Arithmetic expansion and arithmetic evaluation in bash use the same syntax, precedence, and associativity of expressions as in the C language. Within an expression you can use parentheses to change the order of evaluation.

Type of Operator	Function
Post	
	Var++ Postincrement Var-- Postdecrement
Pre	
	++var Preincrement --var Predecrement
Unary	
	+ Unary Plus - Unary Minus
Negation	
	! Boolean NOT ~ Complement
Exponentiation	
	** Exponent
Multiplication, Division, Remainder	
	* Multiplication
	/ Division
	% Remainder
Addition, Subtraction	
	+ Addition
	- Subtraction

Bitwise Shifts	
	<< Left bitwise shift
	>> Right bitwise shift
Comparison	
	<= Less than or equal to
	>= Greater than or equal to
	< less than
	> Greater than
Equality, inequality	
	== Equality
	!= Inequality
Bitwise	
	* Bitwise AND
	^ Bitwise XOR
	| Bitwise OR
Boolean (Logical)	
	&& Boolean AND
	| | Boolean OR
Conditional Evaluation	
	?: Ternary Operator
Assignment	

	=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |= Assignment
Comma	
	, Comma

Pipe

The pipe token has higher precedence than operators. You can use pipes anywhere in a command that you can use simple commands. For example, the command line:

$ **cmd1 | cmd2 || cmd3 | cmd4 && cmd5 | cmd6**

is interpreted as if you had typed

$ **((cmd1 | cmd2) || (cmd3 | cmd4)) && (cmd5 | cmd6)**

9.7 Increment and Decrement

The post increment, post decrement, pre increment, and pre decrement operators work with variables. The pre- operators, which appear in front of the variable name(as in ++COUNT and –– VALUE), first change the value of the variable (++ adds 1;–– subtracts 1) and then provide the result for use in the expression. The post- operators appear after the variable name (as in COUNT++ and VALUE––); they first provide the unchanged value of the variable for use in the expression and then change the value of the variable.

Remainder

The remainder operator (%) yields the remainder when its first operand is divided by its second.

Boolean

The result of a Boolean operation is either 0 (false) or 1 (true). The && (AND) and || (OR) Boolean operators are called short-circuiting operators. If the result of using one of these operators can be decided by looking only at the left operand, the right operand is not evaluated. The && operator causes the shell to test the exit status of the command preceding it. If the command succeeded, bash executes the next command; otherwise, it skips the remaining commands on the command line. You can use this construct to execute commands conditionally.

Ternary

The ternary operator, ? :, decides which of two expressions should be evaluated, based on the value returned by a third expression:**expression1 ? expression2 : expression3**

Assignment: The assignment operators, such as +=, are shorthand notations. For example, N+=3 is the same as ((N=N+3)).

Summary

- The while loop is perfect for a situation where you need to execute a set of commands while some condition is true.
- You can interrupt a for, while, or until loop by using a break or continue statement.

- The break statement transfers control to the statement after the done statement, thereby terminating execution of the loop.
- The continue command transfers control to the done statement, continuing execution of the loop.
- A file descriptor is an index into the process's table of open files.
- A typical Linux process starts with three open files: standard input (file descriptor 0), standard output (file descriptor 1), and standard error (file descriptor 2).
- The <& token duplicates an input file descriptor; >& duplicates an output file descriptor.
- The Bourne Again Shell supports one-dimensional array variables.
- By default, variables are local to the process in which they are declared.
- Special parameters enable you to access useful values pertaining to command-line arguments and the execution of shell commands.
- Positional parameters comprise the command name and command-line arguments. These parameters are called positional because within a shell script, you refer to them by their position on the command line.
- An expression comprises constants, variables, and operators that the shell can process to return a value.

Keywords

- **Break:** The break statement transfers control to the statement after the done statement, thereby terminating execution of the loop.
- **Continue:** The continue command transfers control to the done statement, continuing execution of the loop.
- **File Descriptor**: A file descriptor is an index into the process's table of open files.
- **exec n> outfile:** It opens outfile for output and holds it open, associating it with file descriptor n.
- **exec m< infile:** It opens infile for input and holds it open, associating it with file descriptor m.
- **exec n<&–**: When you have finished using a file, you can close it using:exec n<&–
- **export**: Under bash, export makes a variable available to child processes.
- **Setenv**: Under tcsh, setenv assigns a value to a variable and makes it available to child processes.
- **Pipe**:The pipe token has higher precedence than operators. You can use pipes anywhere in a command that you can use simple commands.

Self Assessment

1. Continue statement

A. Breaks loop and goes to next statement after loop

B. does not break loop but starts new iteration

C. exits the program

D. Starts from beginning of program

2. >& duplicates

A. Input file descriptor

B. Output file descriptor

C. Error file descriptor

D. None of the above

3. << represents

A. Left bitwise shift

B. Right bitwise shift

C. Centre bitwise shift

D. None of the above

4. Which of these is assignment operator?

A. =

B. *=

C. /=

D. All of the above

5. Which of these builtin removes a variable or function?

A. set

B. unset

C. mask

D. umask

6. A typical Linux process has

A. File descriptor 0

B. File descriptor 1

C. File descriptor 2

D. All of the above mentioned

7. Before a file can read/write to a file, it must ______ the file.

A. Open

B. Close

C. Check

D. None of the above

8. We can reference a shell special parameter by preceding a special character with a _____

A. !

B. @

C. #

D. $

9. Which of these operators has the higher precedence?

A. Pipe

B. AND

C. OR

D. NOT

10. Instead of using if…..else multiple times, we can use one __________.

A. case….esac

B. if….fi

C. else……else

D. None of the above

11. Which of these control structures are available in Linux?

A. If…….then

B. For…….in

C. While

D. All of the above mentioned

12. The file descriptor is associated with _______

A. Opening of file

B. Reading from file

C. Writing from file

D. None of the above mentioned

13. The loops can be interrupted by using

A. break

B. continue

C. Both break and continue

D. None of the above

14. ^ represents

A. Bitwise AND

B. Bitwise OR

C. Bitwise XOR

D. None of the above

15. Which of these builtin removes a variable or function?

A. set

B. unset

C. mask

D. umask

Answers for Self Assessment

1.	B	2.	B	3.	A	4.	B	5.	B
6.	D	7.	D	8.	A	9.	A	10.	A
11.	D	12.	A	13.	C	14.	C	15.	B

Review Questions

1. Write a shell script to use a switch statement to process a menu selection, using both upper- and lower-case options.
2. Write a shell script that displays the names of all directory files, but no other types of files, in the working directory.
3. What is a control structure? Explain different types of control structures with examples.
4. Explain the syntax of while and for…in loop with examples.
5. What is a file descriptor? How can we open and duplicate a file descriptor?
6. What are special and positional parameters in Linux?
7. What is a builtin? Give ten examples of builtin commands.

Further Readings

Mark G. Sobell, A Practical Guide to Linux Commands, Editors and Shell Scripting, Second Edition, Prentice Hall.

Web Links

https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/13%3A_Working_with_Bash_Scripts/4.10%3A_Shell_Control_Statements

Unit 10: Linux System Administration

CONTENTS

Objectives

After studying this unit, you will be able to

- Know about the system administrator and superuser
- Understand the rescue mode
- Understand the SELinux
- Understand the system operations and system administration utilities
- Understand how to set up a server
- know important files and directories in Linux
- Understand the file types and file systems
- Know how to back up files and schedule tasks
- Understand how to configure user and group accounts, system reports and parted

Introduction

A well-maintained system:

- Runs quickly enough so users do not get too frustrated waiting for the system to respond or complete a task.
- Has enough storage to accommodate users' reasonable needs.
- Provides a working environment appropriate to each user's abilities and requirements.
- Is secure from malicious and accidental acts altering its performance or compromising the security of the data it holds and exchanges with other systems.
- Is backed up regularly, with recently backed-up files readily available to users.
- Has recent copies of the software that users need to get their jobs done.
- Is easier to administer than a poorly maintained system.

10.1 System Administrator and Superuser

A system administrator should be available to help users with all types of system-related problems from logging in to obtaining and installing software updates to tracking down and fixing obscure network issues. Much of what a system administrator does is work that ordinary user do not have permission to do. When performing one of these tasks, the system administrator logs in as root to have system wide powers that are beyond those of ordinary users: A user with root privileges is referred to as **Super user**. The username is root by default.

Superuser has the following powers:

- Some commands, such as those that add new users, partition hard drives, and change system configuration, can be executed only by root.
- Superuser can use certain tools, such as sudo, to give specific users permission to perform tasks that are normally reserved for Superuser.
- Read, write, and execute file access and directory access permissions do not affect root: Superuser can read from, write to, and execute all files, as well as examine and work in all directories.
- Some restrictions and safeguards that are built into some commands do not apply to root. For example, root can change any user's password without knowing the old password.

When you are running with root (Superuser) privileges, the shell by convention displays a special prompt to remind you of your status. By default, this prompt is or ends with a pound sign (#).

Gain Superuser Privileges

You can gain or grant Superuser privileges in several ways:

1) When you bring the system up in single-user mode, you are Superuser.

2) Once the system is up and running in multiuser mode, you can log in as root. When you supply the proper password, you will be Superuser.

3) You can give an su (substitute user) command while you are logged in as yourself and, with the proper password, you will have Superuser privileges.

4) You can use sudo selectively to give users Superuser privileges for a limited amount of time on a per-user and per-command basis. The sudo utility is controlled by the /etc/sudoers file, which must be set up by root.

5) Any user can create a setuid (set user ID) file. Setuid programs run on behalf of the owner of the file and have all the access privileges that the owner has. While you are running as Superuser, you can change the permissions of a file owned by root to setuid. When an ordinary user executes a file that is owned by root and has setuid permissions, the program has full root privileges.

6) Some programs ask you for a password (either your password or the root password, depending on the command and the configuration of the system) when they start. When you provide the root password, the program runs with root privileges.

10.2 System Administration Tools

Many tools can help you be an efficient and thorough system administrator.

- su: Gives You Another User's Privileges
- console helper: Runs Programs as root
- kill: Sends a Signal to a Process

su: Gives You Another User's Privileges

The su (substitute user) utility can create a shell or execute a program with the identity and permissions of a specified user. Follow su on the command line with the name of a user; if you are working with root privileges or if you know the user's password, you take on the identity of that user. When you give an su command without an argument, su defaults to Superuser so that you take on the identity of root (you have to know the root password). It is better to use /bin/su which is the official version of su to avoid the trouble. When you give an su command to become Superuser, you spawn a new shell, which displays the # prompt. You return to your normal status (and your former shell and prompt) by terminating this shell: Press CONTROL-D or give an exit command.

Console helper: Runs Programs as root

The console helper utility can make it easier for someone who is logged in on the system console but not logged in as root to run system programs that normally can be run only by root.

kill: Sends a Signal to a Process

The kill built in sends a signal to a process. This signal may or may not terminate (kill) the process, depending on which signal is sent and how the process is designed.

10.3 Rescue Mode

Rescue mode is an environment you can use to fix a system that does not boot normally. To bring a system up in rescue mode, boot the system from the first installation CD, the Net Boot CD, or the install DVD. From the install DVD, select Rescue installed system from the Welcome menu. From the first installation CD and the Net Boot CD, enter the rescue (FEDORA) or boot rescue (RHEL) boot parameter. In rescue mode, you can change or replace configuration files, check, and repair partitions using fsck, rewrite boot information, and more. The rescue screen first asks if you want to set up the network interface. This interface is required if you want to copy files from other systems on the LAN or download files from the Internet.

Avoiding a Trojan Horse

A Trojan horse is a program that does something destructive or disruptive to a system while appearing to be benign. As an example, you could store the following script in an executable file named mkfs:

```
while true

do

echo 'Good Morning Mr. Jones. How are you? Ha Ha Ha.' > /dev/console

done
```

If you are running as Superuser when you run this command, it would continuously write a message to the console. If the programmer were malicious, it could do worse. The only thing missing in this plot is access permissions. A malicious user could implement this Trojan horse by changing Superuser's PATH variable to include a publicly writable directory at the start of the PATH string. A good way to help prevent the execution of a Trojan horse is to make sure that your PATH variable does not contain a single colon (:) at the beginning or end of the PATH string or a period (.) or double colon (::) anywhere in the PATH string. This precaution ensures that you will not execute a file in the working directory by accident.

10.4 Security Enhanced Linux

Traditional Linux security, i.e., DAC is based on users and groups. Because a process run by a user has access to anything the user has access to, fine-grained access control is difficult to achieve. SELinux was developed by the U.S. NSA, implements MAC in the Linux kernel. MAC enforces security policies that limit what a user or program can do. It defines a security policy that controls some or all objects, such as files, devices, sockets, and ports, and some or all subjects, such as processes. Using SELinux, you can grant a process only those permissions it needs to be functional, following the principle of least privilege. MAC is an important tool for limiting security threats that come from user errors, software flaws, and malicious users. The kernel checks MAC rules after it checks DAC rules.

States/Modes of SELinux

SELinux can be in one of three states (modes):

- **Enforcing—** This is the default state.
- **Permissive—** This is the diagnostic state.
- **Disabled—** No policy.

Policies of SELinux

SELinux implements one of the following policies:

- **Targeted**—Applies SELinux MAC controls only to certain (targeted) processes (default).
- **MLS**—Multilevel Security protection.
- **Strict**—Applies SELinux MAC controls to all processes (RHEL).

Turning off SELinux

There are two ways to disable SELinux: You can modify the /etc/selinux/config file so that it includes the line SELINUX=disabled and reboot the system, or you can use system-config-selinux.

config: The SELinux Configuration File

The /etc/selinux/config file, which has a link at /etc/sysconfig/selinux, controls the state of SELinux on the local system. Although you can modify this file, it may be more straightforward to work with system-config-selinux. In the following example, the policy is set to targeted, but that setting is of no consequence because SELinux is disabled:

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

To put SELinux in enforcing mode, change the line containing the SELINUX assignment to SELINUX=enforcing. Similarly, you can change the policy by setting SELINUXTYPE.

getenforce, setenforce, and sestatus: Work with SELinux

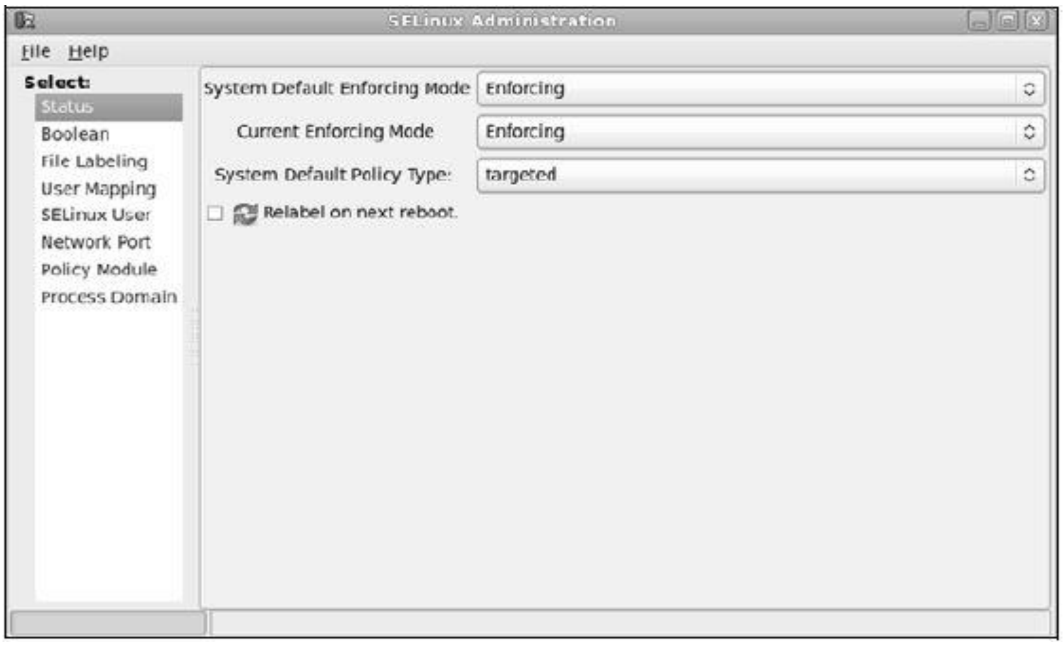
The getenforce and setenforce utilities report on and temporarily set the SELinux mode. The sestatus utility displays a summary of the state of SELinux:

```
# getenforce
Enforcing
# setenforce permissive
# sestatus
SELinux status: enabled
SELinuxfs mount: /selinux
Current mode: permissive
Mode from config file: enforcing
Policy version: 24
Policy from config file: targeted
```

Setting the Targeted Policy with system-config-selinux

The system-config-selinux utility displays the SELinux Administration window, which controls SELinux. To run this utility, enter system-config-selinux from a command line in a graphical environment or select Main menu: System➪Administration➪SELinux Management.

SELinux Administration window



System Default Enforcing Mode: Enforcing
Current Enforcing Mode: Enforcing
System Default Policy Type: targeted
☐ Relabel on next reboot.

With Status highlighted on the left side of the SELinux Administration window, choose Enforcing (default), Permissive, or Disabled from the drop-down list labeled System Default Enforcing Mode. The mode you choose becomes effective next time you reboot the system. You can use the drop-down list labeled Current Enforcing Mode to change between Enforcing and Permissive modes immediately. When you change the mode using this list, the system resumes the default mode when you reboot it. To modify the SELinux policy, highlight Boolean on the left side of the SELinux Administration window and scroll through the list of modules. To find modules that pertain to NFS, type nfs in the text box labeled Filter and then press RETURN. The SELinux Administration window displays all modules with the string nfs in their descriptions. The modules with tick marks in the Active column are in use.

10.5 Run levels

Number	Name	Login	Level	Filesystems
0	Halt			
1	Single user	Textual	Down	Mounted
2	Multiuser without NFS	Textual	Up	Mounted
3	Multiuser	Textual	Up	Mounted
4	User Defined			
5	Multiuser with X	Graphical	Up	Mounted
6	Reboot			

- **Default Run level**: By default, Fedora systems boot to graphical multiuser mode (runlevel 5).
- **runlevel utiity**: The runlevel utility displays the previous and current runlevels. This utility is a transitional tool; it provides compatibility with SysVinit. In the following example, the N indicates that the system does not know what the previous runlevel was and the 5 indicates that the system is in multiuser mode.

```
$ runlevel
N 5
```

- **telinit utility**: The telinit utility allows a user with root privileges to bring the system down, reboot the system, or change between recovery (single-user) and multiuser modes. The telinit utility is a transitional tool; it provides compatibility with SysVinit. The format of a telinit command is: telinit runlevel.
- **Recovery mode and the root password:** When the system enters recovery (single-user) mode, init requests the root password before displaying the root prompt. When the system enters graphical multiuser mode, it displays a graphical login screen.

10.6 <u>Booting the System</u>

Booting a system is the process of reading the Linux kernel into system memory and starting it running. As the last step of the boot procedure, Linux runs the init program as PID number 1. The init program is the first genuine process to run after booting and is the parent of all system processes. (That is why when you run as root and kill process 1, the system dies.)

- **initdefault**: The initdefault entry in the /etc/inittab file tells init which runlevel to bring the system to. Set initdefault to 3 to cause the system to present a text login message when it boots; set it to 5 to present a graphical login screen (default).
- **init daemon:** As the last step of the boot procedure, Linux starts the init daemon as PID number 1. The init daemon is the first genuine process to run after booting and is the parent of all system processes.

Init Scripts: Start and Stop System Services

The first script that runs is /etc/rc.d/rc.sysinit, which performs basic system configuration, including setting the system clock, hostname, and keyboard mapping; setting up swap partitions; checking the filesystems for errors; and turning on quota management.

service: Configures Services I

Fedora/RHEL provides service, a handy utility that can report on or change the status of any of the system services in **/etc/rc.d/init.d.**

system-config-services: Configures Services II

The system-config-services utility displays the Service Configuration window. This utility has two functions: It turns system services on and off immediately, and it controls which services are stopped and started when the system enters and leaves runlevels 2–5. The system-config-services utility works with many of the services listed in /etc/rc.d/init.d as well as with those controlled by xinetd and listed in /etc/xinetd.d (or specified in /etc/xinetd.conf). To run system-config-services, enter system-config-services from a command line in a graphical environment or select Main menu: System | Administration | Services.

chkconfig: Configures Services III

The chkconfig character-based utility duplicates much of what the system-config services utility does: It makes it easier for a system administrator to maintain the /etc/rc.d directory hierarchy.

This utility can add, remove, list startup information, and check the state of system services. It changes the configuration only—it does not change the current state of any service.

Single-User Mode

When the system is in single-user mode, only the system console is enabled. You can run programs from the console in single-user mode just as you would from any terminal in multiuser mode. The only difference is that few of the system daemons will be running. The scripts in /etc/rc.d/rc1.d are run as part of single-user initialization. With the system in single-user mode, you can perform system maintenance that requires file systems to be unmounted or that requires just a quiet system—no one except you using it, so that no user programs interfere with disk maintenance and backup programs.

Going to Multiuser Mode

After you have determined that all is well with the filesystems, you can bring the operating system up to multiuser mode. When you exit from the single-user shell, init brings the system to the default run level—usually 5. Alternatively, you can give the following command in response to the Superuser prompt to bring the system to textual multiuser mode (use 5 to go to graphical multiuser mode):   # /sbin/telinit 3. When it goes from single-user to textual multiuser mode, the system executes the K (kill or stop) scripts and then the S (start) scripts in /etc/rc.d/rc3.d.

Graphical Multiuser Mode

Graphical multiuser mode is the default state for a Fedora/RHEL system. In this mode all appropriate filesystems are mounted, and users can log in from all connected terminals, dial-up lines, and network connections. All support services and daemons are enabled and running. Once the system is in graphical multiuser mode, a login screen appears on the console. Most systems are set up to boot directly to graphical multiuser mode without stopping at single-user mode.

Logging In

- Textual login
- Graphical login

Textual login

With a textual login, the system uses init, mingetty, and login to allow a user to log in; login uses PAM modules to authenticate users. The system is in multiuser mode, the Upstart init daemon is responsible for spawning a mingetty process on each of the lines that a user can use to log in.

Graphical login

With a graphical login, the Upstart init daemon starts gdm (the GNOME display manager) by default on the first free virtual terminal, providing features similar to those offered by mingetty and login. The gdm utility starts an X server and presents a login window. The gdm display manager then uses PAM to authenticate the user and runs the scripts in the /etc/gdm/PreSession directory.

Logging Out

When the system displays a shell prompt, you can either execute a program or exit from the shell. If you exit from the shell, the process running the shell dies and the parent process wakes up. When the shell is a child of another shell, the parent shell wakes up and displays a prompt. Exiting from a login shell causes the operating system to send Upstart a signal that one of its children has died. Upon receiving this signal, Upstart takes action based on the contents of the appropriate tty job definition file. In the case of a process controlling a line for a terminal, Upstart informs mingetty that the line is free for another user. When you are at runlevel 5 and exit from a GUI, the GNOME display manager, gdm, initiates a new login display.

Bringing the System Down

The shutdown and halt utilities perform the tasks needed to bring the system down safely. These utilities can restart the system, prepare the system to be turned off, put the system in single-user mode, and, on some hardware, power down the system. The poweroff and reboot utilities are linked to halt. If you call halt when the system is not shutting down (runlevel 0) or rebooting (runlevel 6), halt calls shutdown. CONTROL-ALT-DEL: Reboots the System

Crash

A crash occurs when the system stops suddenly or fails unexpectedly. A crash may result from software or hardware problems or from a loss of power. As a running system loses power, it may behave in erratic or unpredictable ways. In a fraction of a second, some components are supplied with enough voltage; others are not. Buffers are not flushed, corrupt data may be written to the hard disk, and so on. IDE drives do not behave as predictably as SCSI drives under these circumstances. After a crash, you must bring the operating system up carefully to minimize possible damage to the filesystems. On many occasions, little or no damage will have occurred.

10.7 System Administration Utilities

These utilities can help you perform system administration tasks.

Fedora/RHEL configuration tools

Most of the Fedora/RHEL configuration tools are named system-config-*. These tools bring up a graphical display when called from a GUI; some display a textual interface when called from a non-GUI command line. Some, such as system-configfirewall-tui, use a name with a –tui extension for the textual interface.

Tool	Function
system-config-authentication	Displays the Authentication Configuration window with three tabs. **User Information tab**: It allows you to enable NIS, LDAP, Hesiod, and Winbind support. **Authentication tab**: It allows you to work with Kerberos, LDAP, Smart Card, Fingerprint Reader, and Windbind. **Options tab**: It allows you to use shadow and sha512 passwords as well as to enable other system options.
system-config-bind	Displays the Domain Name Service window.
system-config-boot	Allows you to specify a default kernel and timeout for the **grub.conf file**
system-config-display	Brings up the Display Settings window with three tabs: Settings, Hardware, and Dual Head.

system-config-date	Displays the Date/Time Properties window with two tabs: Date & Time and Time Zone. **Date & Time:** You can set the date and time or enable NTP (Network Time Protocol) from the first tab. **Time:** The Time Zone tab allows you to specify the time zone of the system clock or set the system clock to *UTC*
system-config-firewall[-tui] (FEDORA)	Displays the Firewall Configuration window
system-config-httpd	Displays the HTTP window with four tabs: Main, Virtual Hosts, Server, and Performance Tuning
system-config-keyboard	Displays the Keyboard window, which allows you to select the type of keyboard attached to the system. You use this utility to select the keyboard when you install the system.
system-config-language	Displays the Language Selection window, which allows you to specify the default system language from among those that are installed. You use this utility to select the system language when you install the system.
system-config-lvm	Displays the Logical Volume Management window, which allows you to modify existing logical volumes
system-config-network[-tui]	Displays the Network Configuration window
system-config-network-cmd	Displays the parameters that system-config-network uses.
system-config-nfs	Displays the NFS Server Configuration window
system-config-packages (RHEL)	Runs pirut
system-config-printer	Displays the Printer Configuration window, which allows you to set up printers and edit printer configurations
system-config-rootpassword	Displays the Root Password window, which allows you to change the root password. While logged in as root, you can also use passwd from a command line to change the root password.
system-config-samba	Displays the Samba Server Configuration window, which can help you configure Samba

system-config-selinux (FEDORA)	Displays the SELinux Administration window, which controls SELinux
system-config-services	Displays the Service Configuration window, which allows you to specify which daemons (services) run at each runlevel.
system-config-soundcard (RHEL)	Displays the Audio Devices window, which tells you which audio device the system detected and gives you the option of playing a sound to test the device
system-config-users	Displays the User Manager window, which allows you to work with users and groups

Command-Line Utilities

- **chsh:** Changes the login shell for a user. When you call chsh without an argument, you change your own login shell. Superuser can change the shell for any user by calling chsh with that user's username as an argument.
- **clear:** Clears the screen. You can also use CONTROL-L from the bash shell to clear the screen.
- **dmesg:** Displays the kernel ring buffer.
- **e2label:** Displays or creates a volume label on an ext2, ext3, or ext4 filesystem. An e2label command has the following format: e2label device [newlabel]

where device is the name of the device (e.g., /dev/hda2, /dev/sdb1, /dev/fd0) you want to work with. When you include the optional newlabel parameter, e2label changes the label on device to newlabel. Without this parameter, e2label displays the label. You can also create a volume label with the –L option of tune2fs.

- **mkfs:** Creates a new filesystem on a device. This utility is a front end for many utilities, each of which builds a different type of filesystem. By default, mkfs builds an ext2 filesystem and works on either a hard disk partition or a floppy diskette. Although it can take many options and arguments, you can use mkfs simply as # mkfs device

where device is the name of the device (e.g., /dev/hda2, /dev/sdb1, /dev/fd0) you want to make a file system on.

- **ping:** Sends packets to a remote system. This utility determines whether you can reach a remote system through the network and tells you how much time it takes to exchange messages with the remote system.
- **reset (link to tset):** Resets terminal characteristics. The value of the environment variable **TERM** determines how to reset the screen. The screen is cleared, the kill and interrupt characters are set to their default values, and character echo is turned on. When given from a graphical terminal emulator, this command also changes the size of the window to its default. The reset utility is useful to restore your screen to a sane state after it has been corrupted.
- **setserial:** Gets and sets serial port information. Superuser can use this utility to configure a serial port. The following command sets the input address of **/dev/ttys0 to 0x100,** the interrupt (IRQ) to 5, and the baud rate to 115,000 baud:

  # setserial /dev/ttys0 port 0x100 irq 5 spd_vhi
- **stat:** Displays information about a file or filesystem. Giving the –f (filesystem) option followed by the device name or mount point of a filesystem displays information about the filesystem including the maximum length of filenames.

$ **stat -f /dev/sda**

```
File: "/dev/sda"
ID: 0 Namelen: 255 Type: tmpfs
Block size: 4096 Fundamental block size: 4096
Blocks: Total: 121237 Free: 121206 Available: 121206
Inodes: Total: 121237 Free: 120932
```

- **umask:** shell builtin that specifies a mask the system uses to set up access permissions when you create a file. A umask command has the following format: umask **[mask].**

-where mask is a three-digit octal number or a symbolic value such as you would use with chmod. The mask specifies the permissions that are not allowed.

When mask is an octal number, the digits correspond to the permissions for the owner of the file, members of the group the file is associated with, and everyone else. Because mask specifies the permissions that are not allowed, the system subtracts each of the three digits from 7 when you create a file. A mask that you specify using symbolic values indicates the permissions that are allowed.

- **uname:** Displays information about the system. Without any arguments, this utility displays the name of the operating system (Linux). With a –a (all) option, it displays the operating system name, hostname, version number and release date of the operating system, and type of hardware you are using:

```
$ uname -a
Linux F12 2.6.31.6-145.fc12.i686.PAE #1 SMP Sat Nov 21 16:12:37 EST 2009 i686 athlon i386 GNU/Linux
```

10.8 Standard Rules in Configuration Files

Most configuration files, which are typically named *.conf, rely on the following conventions:

1) Blank lines are ignored.
2) A # anywhere on a line starts a comment that continues to the end of the line. Comments are ignored.
3) When a name contains a SPACE, you must quote the SPACE by preceding it with a backslash (\) or by enclosing the entire name within single or double quotation marks.
4) To make long lines easier to read and edit, you can break them into several shorter lines. Break a line by inserting a backslash (\) immediately followed by a NEWLINE (press RETURN in a text editor).

Specifying Clients

Some common ways to specify a host or a subnet:

Client name pattern	Matches
n.n.n.n	One IP address.
name	One hostname, either local or remote.
Name that starts with .	Matches a hostname that ends with the specified string. For example, .tcorp.com matches the systems kudos.tcorp.com and speedy.tcorp.com, among others.

IP address that ends with .	Matches a host address that starts with the specified numbers. For example, 192.168.0. matches 192.168.0.0 – 192.168.0.255. If you omit the trailing period, this format does not work.
Starts with @	Specifies a netgroup.
n.n.n.n/m.m.m.m or n.n.n.n/mm	An IP address and subnet mask specify a subnet.
Starts with /	An absolute pathname of a file containing one or more names or addresses as specified in this table.

Wildcard	Matches
* and ?	Matches one (?) or more (*) characters in a simple hostname or IP address. These wildcards do not match periods in a domain name.
ALL	Always matches.
LOCAL	Matches any hostname that does not contain a period.

Operator	Function
EXCEPT	Matches anything in the preceding list that is not in the following list. For example, a b c d EXCEPT c matches a, b, and d. Thus you could use 192.168. EXCEPT 192.168.0.1 to match all IP addresses that start with 192.168. except 192.168.0.1.

Specifying a Subnet

When you set up a server, you frequently need to specify which clients are allowed to connect to the server. Sometimes it is convenient to specify a range of IP addresses, called a subnet. Usually, you can specify a subnet as

n.n.n.n/m.m.m.m

or

n.n.n.n/maskbits

where n.n.n.n is the base IP address and the subnet is represented by m.m.m.m (the subnet mask) or maskbits (the number of bits used for the subnet mask). Example: 192.168.0.1/255.255.255.0 represents the same subnet as 192.168.0.1/24. In binary, decimal 255.255.255.0 is represented by 24 ones followed by 8 zeros. The /24 is shorthand for a subnet mask with 24 ones.

Different ways to represent a subnet

Bits	**Mask**	**Range**
10.0.0.0/8	10.0.0.0/255.0.0.0	10.0.0.0 – 10.255.255.255
172.16.0.0/12	172.16.0.0/255.240.0.0	172.16.0.0 – 172.31.255.255
192.168.0.0/16	192.168.0.0/255.255.0.0	192.168.0.0 – 192.168.255.255

rpcinfo: Displays Information About rpcbind

Fedora uses the rpcbind daemon while RHEL uses portmap for the same purpose. The rpcinfo utility displays information about programs registered with rpcbind and makes RPC calls to programs to see if they are alive. The rpcinfo utility takes the following options and arguments:

rpcinfo –p **[host]**

rpcinfo [–n **port] –u | –t host program [version]**

rpcinfo –b | –d **program version**

There are various options available, and the associated functions are given in the below table:

Option	**Function**
-b (broadcast)	Makes an RPC broadcast to version of program and lists hosts that respond.
-d (delete)	Removes local RPC registration for version of program. Available to Superuser only.
-n (port number)	With –t or –u, uses the port numbered port instead of the port number specified by rpcbind.
-p (probe)	Lists all RPC programs registered with rpcbind on host or on the local system if host is not specified.
-t (TCP)	Makes a TCP RPC call to version (if specified) of program on host and reports whether it received a response.
-u (UDP)	Makes a UDP RPC call to version (if specified) of program on host and reports whether it received a response.

Give the following command to see which RPC programs are registered with the rpcbind daemon (portmapper) on the system named peach:

```
$ /usr/sbin/rpcinfo -p peach
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
```

```
100024 1 udp 32768 status
100024 1 tcp 32768 status
100021 1 udp 32769 nlockmgr
100021 3 udp 32769 nlockmgr
```

Locking down rpcbind

Because the rpcbind daemon holds information about which servers are running on the local system and which port each server is running on, only trusted systems should have access to this information.

- One way to ensure only selected systems have access to rpcbind is to lock it down in the /etc/hosts.allow and /etc/hosts.deny files.
- Put the following line in hosts.deny preventing all systems from using rpcbind on the local (server) system: rpcbind: ALL

10.9 The xinetd Superserver

RHEL uses the xinetd daemon, a more secure replacement for the inetd superserver that was originally shipped with 4.3BSD. Fedora uses the Upstart init daemon for runlevel control and most servers. However, some Fedora servers still require xinetd to be installed and running. The xinetd superserver listens for network connections. When one is made, it launches a specified server daemon and forwards the data from the socket to the daemon's standard input. The version of xinetd distributed with Fedora/RHEL is linked against libwrap.so, so it can use the /etc/hosts.allow and /etc/hosts.deny files for access control. Using TCP wrappers can simplify configuration but hides some of the more advanced features of xinetd. The base configuration for xinetd is stored in the /etc/xinetd.conf file. If this file is not present, xinetd is probably not installed. Working as root, give the following command to install xinetd: # yum install xinetd. The default xinetd.conf file is well commented.

Securing a Server

You may secure a server either by using TCP wrappers or by setting up a chroot jail.

TCP Wrappers: Client/Server Security (hosts.allow and hosts.deny)

When you open a local system to access from remote systems, you must ensure that the following criteria are met: Open the local system only to systems you want to allow to access it. Allow each remote system to access only the data you want it to access. Allow each remote system to access data only in the appropriate manner (readonly, read/write, write only). As part of the client/server model, TCP wrappers, which can be used for any daemon that is linked against libwrap.so, rely on the /etc/hosts.allow and /etc/hosts.deny files as the basis of a simple access control language. This access control language defines rules that selectively allow clients to access server daemons on a local system based on the client's address and the daemon the client tries to access.

Each line in the hosts.allow and hosts.deny files has the following format:

```
daemon_list : client_list [: command]
```

where daemon_list is a comma-separated list of one or more server daemons (such as rpcbind, vsftpd, or sshd), client_list is a comma-separated list of one or more clients and the optional command is the command that is executed when a client from client_list tries to access a server daemon from daemon_list.

When a client requests a connection with a local server, the hosts.allow and hosts.deny files are consulted in the following manner until a match is found:

1. If the daemon/client pair matches a line in hosts.allow, access is granted.
2. If the daemon/client pair matches a line in hosts.deny, access is denied.

3. If there is no match in either the hosts.allow or hosts.deny files, access is granted.

Setting Up a chroot Jail

On early UNIX systems, the root directory was a fixed point in the filesystem. On modern UNIX variants, including Linux, you can define the root directory on a preprocess basis. The chroot utility allows you to run a process with a root directory other than **/.** The root directory appears at the top of the directory hierarchy and has no parent: A process cannot access any files above the root directory (because they do not exist). If, for example, you run a program (process) and specify its root directory as /home/sam/jail, the program would have no concept of any files in /home/sam or above: jail is the program's root directory and is labeled / (not jail). By creating an artificial root directory, frequently called a (chroot) jail, you prevent a program from accessing or modifying—possibly maliciously—files outside the directory hierarchy starting at its root. You must set up a chroot jail properly to increase security: If you do not set up the chroot jail correctly, you can make it easier for a malicious user to gain access to a system than if there were no chroot jail.

10.10 DHCP: Configures Hosts

Instead of storing network configuration information in local files on each system, DHCP (Dynamic Host Configuration Protocol) enables client systems to retrieve network configuration information each time they connect to the network. A DHCP server assigns IP addresses from a pool of addresses to clients as needed. Assigned addresses are typically temporary but need not be. This technique has several advantages over storing network configuration information in local files:

1) A new user can set up an Internet connection without having to deal with IP addresses, netmasks, DNS addresses, and other technical details. An experienced user can set up a connection more quickly.

2) DHCP facilitates assignment and management of IP addresses and related network information by centralizing the process on a server. A system administrator can configure new systems, including laptops that connect to the network from different locations, to use DHCP; DHCP then assigns IP addresses only when each system connects to the network. The pool of IP addresses is managed as a group on the DHCP server.

3) IP addresses can be used by more than one system, reducing the total number of IP addresses needed. This conservation of addresses is important because the Internet is quickly running out of IPv4 addresses. Although a particular IP address can be used by only one system at a time, many end-user systems require addresses only occasionally, when they connect to the Internet. By reusing IP addresses, DHCP lengthens the life of the IPv4 protocol.

DHCP is particularly useful for administrators who are responsible for maintaining many systems because individual systems no longer need to store unique configuration information.

How DHCP Works

The client daemon, dhclient (part of the dhcp package), contacts the server daemon, dhcpd, to obtain the IP address, netmask, broadcast address, nameserver address, and other networking parameters. The server provides a lease on the IP address to the client. The client can request the specific terms of the lease, including its duration; the server can, in turn, limit these terms. While connected to the network, a client typically requests extensions of its lease as necessary, so its IP address remains the same. The lease can expire once the client is disconnected from the network, with the server giving the client a new IP address when it requests a new lease. You can also set up a DHCP server to provide static IP addresses for specific clients

DHCP Client

A DHCP client requests network configuration parameter from the DHCP server and uses those parameters to configure its network interface. The prerequisites is to install the following package: dhclient. When a DHCP client system connects to the network, dhclient requests a lease from the DHCP server and configures the client's network interface(s). Once a DHCP client has requested and established a lease, it stores information about the lease in a file named dhclient.leases, which is stored in the /var/lib/dhclient directory. This information is used to reestablish a lease when either the server or the client needs to reboot. The DHCP client configuration file, /etc/dhclient.conf, is required only for custom configurations.

DHCP Server

The DHCP server maintains a list of IP addresses and other configuration parameters. When requested to do so, the DHCP server provides configuration parameters to a client. The prerequisites is to install the following package: **dhcp.** Run chkconfig to cause dhcpd to start when the system enters multiuser mode: # /sbin/chkconfig dhcpd on.

Start dhcpd: # /sbin/service dhcpd start. A simple DHCP server allows you to add clients to a network without maintaining a list of assigned IP addresses. A simple network, such as a home LAN sharing an Internet connection, can use DHCP to assign a dynamic IP address to almost all nodes. The exceptions are servers and routers, which must be at known network locations to be able to receive connections. If servers and routers are configured without DHCP, you can specify a simple DHCP server configuration in /etc/dhcp/dhcpd.conf (FEDORA) or /etc/dhcpd.conf (RHEL):

```
$ cat /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 86400;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.1;
option domain-name-servers 192.168.1.1;
subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.2 192.168.1.200;
}
```

Once you have configured a DHCP server, you can start (or restart) it by using the dhcpd init script: # /sbin/service dhcpd restart. Once the server is running, clients configured to obtain an IP address from the server using DHCP should be able to do so.

Static IP Addresses

Routers and servers typically require static IP addresses. While you can manually configure IP addresses for these systems, it may be more convenient to have the DHCP server provide them with static IP addresses. When a system that requires a specific static IP address connects to the network and contacts the DHCP server, the server needs a way to identify the system so the server can assign the proper IP address to the system. The DHCP server uses the MAC address of the system's Ethernet card (NIC) as an identifier. When you set up the server, you must know the MAC address of each system that requires a static IP address. You can use ifconfig to display the MAC addresses of the Ethernet cards (NICs) in a system. The MAC addresses are the colon-separated series of hexadecimal number pairs following HWaddr:

```
$ /sbin/ifconfig | grep -i hwaddr
eth0 Link encap:Ethernet HWaddr BA:DF:00:DF:C0:FF
eth1 Link encap:Ethernet HWaddr 00:02:B3:41:35:98
```

10.11 Important files and directories in Linux

Filesystems hold directories of files. These structures store user data and system data that are the basis of users' work on the system and the system's existence.

~/.bash_profile

It Contains an individual user's login shell initialization script. The shell executes the commands in this file in the same environment as the shell each time a user logs in. The file must be located in a user's home directory. The default Fedora/RHEL .bash_profile file executes the commands in ~/.bashrc. You can use .bash_profile to specify a terminal type (for vi, terminal emulators, and other programs), run stty to establish the terminal characteristics, set up aliases, and perform other housekeeping functions when a user logs in. A simple .bash_profile file specifying a vt100 terminal and CONTROL-H as the erase key follows:

```
$ cat .bash_profile

export TERM=vt100

stty erase '^h'
```

~/.bashrc

It Contains an individual user's interactive, nonlogin shell initialization script. The shell executes the commands in this file in the same environment as the (new) shell each time a user creates a new interactive shell. The .bashrc script differs from .bash_profile in that it is executed each time a new shell is spawned, not just when a user logs in. The default Fedora/RHEL .bash_profile file executes the commands in ~/.bashrc so that these commands are executed when a user logs in.

/dev

It contains files representing pseudo devices and physical devices that may be attached to the system.

- **/dev/fd0:** The first floppy disk. The second floppy disk is named /dev/fd1.
- **/dev/had:** The master disk on the primary IDE controller. The slave disk on the primary IDE controller is named /dev/hdb. This disk may be a CDROM drive.
- **/dev/hdc:** The master disk on the secondary IDE controller. The slave disk on the secondary IDE controller is named /dev/hdd. This disk may be a CD-ROM drive.
- **/dev/sda:** Traditionally the first SCSI disk; now the first non-IDE drive, including SATA and USB drives. Other, similar drives are named /dev/sdb, /dev/sdc, etc.

These names, such as /dev/sda, represent the order of the devices on the bus the devices are connected to, not the device itself. For example, if you swap the data cables on the disks referred to as /dev/sda and /dev/sdb, the drive's designations will change. Similarly, if you remove the device referred to as /dev/sda, the device that was referred to as /dev/sdb will now be referred to as /dev/sda.

/dev/disk/by-id

It holds symbolic links to local devices. The names of the devices in this directory identify the devices. Each entry points to the device in /dev that it refers to.

/dev/disk/by-uuid

It holds symbolic links to local devices. The names of the devices in this directory consist of the UUID numbers of the devices. Each entry points to the device in /dev that it refers to.

/dev/null

It is also called a bit bucket, output sent to this file disappears. The /dev/null file is a device file and must be created with mknod. Input that you redirect to come from this file appears as nulls, creating an empty file. You can create an empty file named nothing by giving the following command:

```
$ cat /dev/null > nothing
```

or

```
$ cp /dev/null nothing
```

or, without explicitly using /dev/null,

```
$ > nothing
```

/dev/pts

The /dev/pts pseudofilesystem is a hook into the Linux kernel; it is part of the pseudoterminal support. Pseudoterminals are used by remote login programs, such as ssh and telnet, and xterm as well as by other graphical terminal emulators. The following sequence of commands demonstrates that the user is logged in on /dev/pts/1. After using who am i to verify the line the user is logged in on and using ls to show that this line exists, the user redirects the output of an echo command to /dev/pts/1, whereupon the output appears on the user's screen:

```
$ who am i
alex pts/1 2006-02-16 12:30 (bravo.example.com)
$ ls /dev/pts
0 1 2 3 4
$ echo Hi there > /dev/pts/1
Hi there
```

/dev/random and /dev/urandom

These files are interfaces to the kernel's random number generator. You can use either one with dd to create a file filled with pseudorandom bytes.

```
$ dd if=/dev/urandom of=randfile2 bs=1 count=100
100+0 records in
100+0 records out
100 bytes (100 B) copied, 0.001241 seconds, 80.6 kB/s
```

The preceding command reads from /dev/urandom and writes to the file named randfile. The block size is 1 and the count is 100 so randfile is 100 bytes long. For bytes that are more random, you can read from /dev/random.

/dev/zero

Input you take from this file contains an infinite string of zeros (numerical zeros, not ASCII zeros). You can fill a file or overwrite a file with zeros with a command such as the following

```
$ dd if=/dev/zero of=zeros bs=1024 count=10
10+0 records in
10+0 records out
10240 bytes (10 kB) copied, 0.000195 seconds, 52.5 MB/s
$ od -c zeros
0000000 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
*
0024000
```

/etc/aliases

It is used by the mail delivery system (typically sendmail) to hold aliases for users. Edit this file to suit local needs.

/etc/at.allow, /etc/at.deny, /etc/cron.allow, and /etc/cron.deny

By default, users can use the at and cron tab utilities. The at.allow file lists the users who are allowed to use at. The cron.allow file works in the same manner for crontab. The at.deny and cron.deny files specify users who are not permitted to use the corresponding utilities. As Fedora/RHEL is configured, an empty at.deny file and the absence of an at.allow file allows anyone to use at; the absence of cron.allow and cron.deny files allows anyone to use crontab. To prevent anyone except Superuser from using at, remove the at.allow and at.deny files. To prevent anyone except Superuser from using crontab, create a cron.allow file with the single-entry root.

/etc/dumpdates

It contains information about the last execution of dump. For each filesystem, it stores the time of the last dump at a given dump level. The dump utility uses this information to determine which files to back up when executing at a particular dump level.

/etc/fstab

Filesystem (mount) table Contains a list of all mountable devices as specified by the system administrator. Programs do not write to this file but only read from it.

/etc/group

Groups allow users to share files or programs without giving all system users access to those files or programs. This scheme is useful when several users are working with files that are not public. The /etc/group file associates one or more usernames with each group (number). An entry in the /etc/group file has four fields arranged in the following format: **group-name:password:group-ID:login-name-list**

The group-name is the name of the group. The password is an optional encrypted password. This field frequently contains an x, indicating that group passwords are not used. The group-ID is a number, with 1–499 reserved for system accounts. The login-name-list is a comma-separated list of users who belong to the group. The login-name-list is a comma-separated list of users who belong to the group. If an entry is too long to fit on one line, end the line with a backslash (\), which quotes the following RETURN, and continue the entry on the next line.

/etc/hosts

The /etc/hosts file stores the name, IP address, and optional aliases of the other systems that the local system knows about. At the very least, this file must have the hostname and IP address that you have chosen for the local system and a special entry for localhost. This entry supports the loopback service, which allows the local system to talk to itself (for example, for RPC services). The IP address of the loopback service is always 127.0.0.1. Following is a simple /etc/hosts file for the system named rose with an IP address of 192.168.0.10:

```
$ cat /etc/hosts

# Do not remove the following line, or various programs

# that require network functionality will fail.

127.0.0.1 rose localhost.localdomain localhost

192.168.0.1 bravo.example.com bravo

192.168.0.4 mp3server

192.168.0.5 workstation

192.168.0.10 rose
```

...

/etc/inittab (RHEL)

Initialization table Under RHEL, this file controls how the System V init process behaves. Fedora has replace the System V init daemon with the Upstart init daemon. Each line in inittab contains four colon-separated fields: **id:runlevel:action:process.** The id uniquely identifies an entry in the inittab file. The runlevel is the system runlevel(s) at which process is executed. The runlevel consists of zero or more characters chosen from 0123456S. If more than one runlevel is listed, the associated process is executed at each of the specified runlevels. The action is one of the following keywords: respawn, wait, once, boot, bootwait, ondemand, powerfail, powerwait, powerokwait, powerfailnow, ctrlaltdel, kbrequest, off, ondemand, initdefault, or sysinit. The wait keyword instructs init to start the process and wait for it to terminate.

/etc/motd

It contains the message of the day, which can be displayed each time someone logs in using a textual login. This file typically contains site policy and legal information. Keep this file short because users tend to see the message many times.

/etc/mtab

When you call mount without any arguments, it consults this file and displays a list of mounted devices. Each time you (or an init script) call mount or umount, these utilities make the necessary changes to mtab. Although this is an ASCII text file, you should not edit it.

/etc/netgroup

It defines netgroups, which are used for checking permissions when performing remote logins and remote mounts and when starting remote shells.

/etc/nsswitch.conf

It specifies whether a system uses as the source of certain information NIS, DNS, local files, or a combination, and in what order it consults these services.

/etc/pam.d

Files in this directory specify the authentication methods used by PAM applications.

/etc/passwd

It describes users to the system. Do not edit this file directly. Each line in passwd has seven colon-separated fields that describe one user: **login-name:dummy-password:user-ID:group-ID:info: directory: program**

The login-name is the user's username—the name you enter in response to the login: prompt or GUI login screen. The value of the dummy-password is the character x. An encrypted/hashed password is stored in /etc/shadow. For security reasons, every account should have a password. By convention, disabled accounts have an asterisk (*) in this field. The user-ID is a number, with 0 indicating Superuser and 1–499 being reserved for system accounts. The group-ID identifies the user as a member of a group. It is a number, with 0–499 being reserved for system accounts; see /etc/group. You can change these values and set maximum values in /etc/login.defs. The info is information that various programs, such as accounting programs and email, use to identify the user further. Normally it contains at least the first and last names of the user. It is referred to as the GECOS field. The directory is the absolute pathname of the user's home directory. The program is the program that runs once the user logs in. If program is not present, a value of /bin/bash is assumed. You can put /bin/tcsh here to log in using the TC Shell or /bin/zsh to log in using the Z Shell, assuming the shell you specify is installed.

/etc/printcap

The printer capability database. This file describes system printers and is derived from 4.3BSD UNIX.

/etc/profile

It contains a systemwide interactive shell initialization script for environment and start-up programs. When you log in, the shell immediately executes the commands in this file in the same environment as the shell. Following is an example of a /etc/profile file that displays the message of the day (the /etc/motd file), sets the file-creation mask, and sets the interrupt character to CONTROL-C:

```
# cat /etc/profile
cat /etc/motd
umask 022
stty intr '^c'
```

/etc/protocols

It provides protocol numbers, aliases, and brief definitions for DARPA Internet TCP/IP protocols. Do not modify this file.

/etc/rc.d

It holds the system init scripts, also called run command (rc) scripts. The init program executes several init scripts each time the system changes state or runlevel.

/etc/resolv.conf

The resolver configuration file, used to provide access to DNS. The following example shows the resolv.conf file for the example.com domain. A resolv.conf file usually has at least two lines—a domain line and a nameserver line: **# cat /etc/resolv.conf**

```
domain example.com
nameserver 10.0.0.50
nameserver 10.0.0.51
```

/etc/rpc

It maps RPC services to RPC numbers. The three columns in this file show the name of the server for the RPC program, the RPC program number, and any aliases.

/etc/services

It lists system services. The three columns in this file show the informal name of the service, the port number/protocol the service frequently uses, and any aliases for the service. This file does not specify which services are running on the local system, nor does it map services to port numbers. The services file is used internally to map port numbers to services for display purposes.

/etc/shadow

It contains encrypted or MD5 hashed user passwords. Each entry occupies one line composed of nine fields, separated by colons: **login-name: password: last-mod: min: max: warn: inactive: expire: flag.** The login-name is the user's username—the name that the user enters in response to the login: prompt or GUI login screen. The password is an encrypted or hashed. The last-mod field indicates when the password was last modified. The min is the minimum number of days that must elapse before the password can be changed. The max is the maximum number of days before the password must be changed. The warn specifies how much advance warning (in days) to give the user before the password expires. The account will be closed if the number of days between login sessions exceeds the number of days specified in the inactive field. The account will also be closed as of the date in the expire field. The last field in an entry, flag, is reserved for future use. You can use the Password Info tab in system-config-users to modify these fields.

/etc/sysconfig

A directory containing a hierarchy of system configuration files.

/proc

The /proc pseudofilesystem provides a window into the Linux kernel. Through /proc you can obtain information on any process running on your computer, including its current state, memory usage, CPU usage, terminal, parent, and group. You can extract information directly from the files in /proc.

/sbin/shutdown

A utility that brings the system down.

swap

Even though swap is not a file, swap space can be added and deleted from the system dynamically. Swap space is used by the virtual memory subsystem. When it runs low on real memory (RAM), the system writes memory pages from RAM to the swap space on the disk. Which pages are written and when they are written are controlled by finely tuned algorithms in the Linux kernel. When needed by running programs, these pages are brought back into RAM—a technique called paging. When a system is running very short on memory, an entire process may be paged out to disk.

/sys

The /sys pseudofilesystem was added in the Linux 2.6 kernel to make it easy for programs running in kernelspace, such as device drivers, to exchange information with programs running in userspace.

/usr/share/magic

Most files begin with a unique identifier called a magic number. This file is a text database listing all known magic numbers on the system. When you use the file utility, it consults /usr/share/magic to determine the type of a file. Occasionally you may acquire a new tool that creates a new type of file that is unrecognized by the file utility. In this situation you need to update the /usr/share/magic file

/var/log

It holds system log files.

/var/log/messages

It contains messages from daemons, the Linux kernel, and security programs. For example, you will find filesystem full warning messages, error messages from system daemons (NFS, rsyslog, printer daemons), SCSI and IDE disk error messages, and more in messages. Check /var/log/messages periodically to keep informed about important system events. Much of the information displayed on the system console is also sent to messages. If the system experiences a problem and you cannot access the console, check this file for messages about the problem.

/var/log/secure

It holds messages from security-related programs such as su and the sshd daemon.

10.12 Types of files

Linux supports many types of files:

1) Ordinary files, directories, links, and inodes.

2) Symbolic links.

3) Special files.

4) FIFO special file.

5) Sockets.

6) Block and character devices.

7) Raw devices.

Ordinary Files, Directories, Links, and Inodes

Ordinary and directory files

An ordinary file stores user data, such as textual information, programs, or images, such as a jpeg or tiff file. A directory is a standard-format disk file that stores information, including names, about ordinary files, and other directory files.

Inodes

An inode is a data structure, stored on disk, that defines a file's existence and is identified by an inode number. An inode contains critical information, such as the name of the owner of the file, where it is physically located on the disk, and how many hard links point to it. In addition, SELinux stores extended information about files in inodes. A directory relates each of the filenames it stores to an inode. When you move (mv) a file within a filesystem, you change the filename portion of the directory entry associated with the inode that describes the file. You do not create a new inode.

If you move a file to another filesystem, mv first creates a new inode on the destination filesystem and then deletes the original inode. You can also use mv to move a directory recursively, in which case all files in the directory are copied and deleted. When you remove (rm) a file, you delete the directory entry that describes the file. When you remove the last hard link to a file, the operating system puts all blocks the inode pointed to back in the free list (the list of blocks that are available for use on the disk) and frees the inode to be used again.

The . and .. directory entries

Every directory contains at least two entries (. and ..). The . entry is a link to the directory itself. The .. entry is a link to the parent directory. In the case of the root directory, there is no parent and the .. entry is a link to the root directory itself. It is not possible to create hard links to directories.

Symbolic links

Because each filesystem has a separate set of inodes, you can create hard links to a file only from within the filesystem that holds that file. To get around this limitation, Linux provides symbolic links, which are files that point to other files. Files that are linked by a symbolic link do not share an inode. Therefore, you can create a symbolic link to a file from any filesystem. You can also create a symbolic link to a directory, device, or other special file.

Special Files

Special files represent Linux kernel routines that provide access to an operating system feature. FIFO (first in, first out) special files allow unrelated programs to exchange information. Sockets allow unrelated processes on the same or different computers to exchange information. One type of socket, the UNIX domain socket, is a special file. Symbolic links are another type of special file.

Device files

Device files, which include both block and character special files, represent device drivers that let you communicate with peripheral devices, such as terminals, printers, and hard disks. By convention, device files appear in the /dev directory and its subdirectories. Each device file represents a device: You read from and write to the file to read from and write to the device it represents. For example, using cat to send an audio file to /dev/dsp plays the file. The following example shows part of the output that an ls –l command produces for the /dev directory:

```
$ ls -l /dev
total 0
crw-rw---- 1 root root 14, 12 Jan 25 08:33 adsp
crw------- 1 root root 10, 175 Jan 25 08:33 agpgart
crw------- 1 zach root 14, 4 Jan 25 08:33 audio
drwxr-xr-x 3 root root 60 Jan 25 08:33 bus
lrwxrwxrwx 1 root root 3 Jan 25 08:33 cdrom -> hdb
lrwxrwxrwx 1 root root 3 Jan 25 08:33 cdwriter -> hdb
.........................
```

The first character of each line is always –, b, c, d, l, or p, representing ordinary (plain), block, character, directory, symbolic link, or named pipe (see the following section), respectively. The next nine characters identify the permissions for the file, followed by the number of hard links and the names of the owner and group. Where the number of bytes in a file would appear for an ordinary or directory file, a device file shows major and minor device numbers separated by a comma. The rest of the line is the same as any other ls –l listing.

udev

The udev utility manages device naming dynamically. It replaces the earlier devfs and moves the device naming functionality from the kernel to userspace. Because devices are added to and removed from a system infrequently, the performance penalty associated with this change is minimal. The benefit of the move is that a bug in udev cannot compromise or crash the kernel. The udev utility is part of the hotplug system. When a device is added to or removed from the system, the kernel creates a device name in the /sys pseudofilesystem and notifies hotplug of the event, which is received by udev. The udev utility then creates the device file, usually in the /dev directory, or removes the device file from the system. The udev utility can also rename network interfaces.

Hotplug

The hotplug system allows you to plug a device into the system and use it immediately. Although hotplug was available in the Linux 2.4 kernel, the 2.6 kernel integrates hotplug with the unified device driver model framework.

FIFO Special Files (Named Pipe)

A FIFO special file, also called a named pipe, represents a pipe: You read from and write to the file to read from and write to the pipe. The term FIFO stands for first in, first out—the way any pipe works. In other words, the first information that you put in one end is the first information that comes out the other end. When you use a pipe on a command line to send the output of a program to the printer, the printer outputs the information in the same order that the program produced it and sent it to the pipe. The UNIX and Linux systems have included pipes for many generations. Without named pipes, only processes that were children of the same ancestor could use pipes to exchange information. Using named pipes, any two processes on a single system can exchange information. When one program writes to a FIFO special file, another program can read from the same file. The programs do not have to run at the same time or be aware of each other's activity. The operating system handles all buffering and information storage. This type of communication is termed asynchronous (async) because programs on the ends of the pipe do not have to be synchronized.

Sockets

Like a FIFO special file, a socket allows asynchronous processes that are not children of the same ancestor to exchange information. Sockets are the central mechanism of the interprocess communication that forms the basis of the networking facility. When you use networking utilities, pairs of cooperating sockets manage the communication between the processes on the local computer and the remote computer. Sockets form the basis of such utilities as ssh and scp.

Major and Minor Device Numbers

A major device number points to a driver in the kernel that works with a class of hardware devices: terminal, printer, tape drive, hard disk, and so on. In the list of the /dev directory, all of the hard disk partitions have a major device number of 3. A minor device number represents a particular piece of hardware within a class. Although all hard disk partitions are grouped together by their major device number, each has a different minor device number (sda1 is 1, sda2 is 2, and so on). This setup allows one piece of software (the device driver) to service all similar hardware yet to be able to distinguish among different physical units.

Block and Character Devices

A block device is an I/O (input/output) device that is characterized by

- Being able to perform random access reads.
- Having a specific block size.
- Handling only single blocks of data at a time.
- Accepting only transactions that involve whole blocks of data.
- Being able to have a filesystem mounted on it.
- Having the Linux kernel buffer its input and output.

Appearing to the operating system as a series of blocks numbered from 0 through n – 1, where n is the number of blocks on the device. Block devices commonly found on a Linux system include hard disks, floppy diskettes, and CDs.

A character device is any device that is not a block device. Examples of character devices include printers, terminals, tape drives, and modems. The device driver for a character device determines how a program reads from and writes to that device. For example, the device driver for a terminal allows a program to read the information you type on the terminal in two ways:

- First, a program can read single characters from a terminal in raw mode—that is, without the driver doing any interpretation of the characters.
- Alternatively, a program can read one line at a time. When a program reads one line at a time, the driver handles the erase and kill characters, so the program never sees typing mistakes that have been corrected. In this case, the program reads everything from the beginning of a line to the RETURN that ends a line; the number of characters in a line can vary.

Raw Devices

Device driver programs for block devices usually have two entry points so they can be used in two ways: as block devices or as character devices. The character device form of a block device is called a raw device. A raw device is characterized by

- Direct I/O (no buffering through the Linux kernel).
- A one-to-one correspondence between system calls and hardware requests.
- Device-dependent restrictions on I/O.

An example of a utility that uses a raw device is fsck. It is more efficient for fsck to operate on the disk as a raw device, rather than being restricted by the fixed size of blocks in the block device interface. Because it has full knowledge of the underlying filesystem structure, fsck can operate on the raw device using the largest possible units. When a filesystem is mounted, processes normally access the disk through the block device interface, which explains why it is important to allow fsck to modify only an unmounted filesystem. On a mounted filesystem, there is the danger that, while fsck is rearranging the underlying structure through the raw device, another process could change a disk block using the block device, resulting in a corrupted filesystem.

10.13 Filesystems

Filesystem	Features
adfs	Advanced Disc Filing System. Used on Acorn computers. The word Advanced differentiated this filesystem from its predecessor, DFS, which did not support advanced features such as a hierarchical filesystem.
affs	Amiga Fast Filesystem (FFS).
autofs	Automounting filesystem
coda	CODA distributed filesystem
devpts	A pseudofilesystem for pseudoterminals
ext2	A standard filesystem for Fedora/RHEL systems, usually with the ext3 extension.
ext3	A journaling extension to the ext2 filesystem. It greatly improves recovery time from crashes (it takes a lot less time to run fsck), promoting increased availability. As with any filesystem, a journaling filesystem can lose data during a system crash or hardware failure.
ext4	An extension of the ext3 filesystem
GFS	Global Filesystem. GFS is a journaling, clustering filesystem. It enables a cluster of Linux servers to share a common storage pool.
hfs	Hierarchical Filesystem: used by older Macintosh systems. Newer Macintosh

	systems use hfs+.
hpfs	High-Performance Filesystem: the native filesystem for IBM's OS/2.
iso9660	The standard filesystem for CD-ROMs.
minix	Very similar to Linux, the filesystem of a small operating system that was written for educational purposes by Andrew S. Tanenbaum
msdos	The filesystem used by DOS and subsequent Microsoft operating systems. Do not use msdos for mounting Windows filesystems; it does not read VFAT attributes.
ncpfs	Novell NetWare NCP Protocol Filesystem: used to mount remote filesystems under NetWare.
nfs	Network Filesystem. Developed by Sun Microsystems, this protocol allows a computer to access remote files over a network as if they were local
ntfs	NT Filesystem: the native filesystem of Windows NT.
proc	An interface to several Linux kernel *data structures that behaves* like a filesystem
qnx4	QNX 4 operating system filesystem
reiserfs	A journaling filesystem, based on balanced-tree algorithms.
romfs	A dumb, readonly filesystem used mainly for *RAM disks during* installation
smbfs	Samba Filesystem
software RAID	RAID implemented in software.
sysv	System V UNIX filesystem.
ufs	Default filesystem under Sun's Solaris operating system and other UNIXs.
umsdos	A full-feature UNIX-like filesystem that runs on top of a DOS FAT filesystem.
vfat	Developed by Microsoft, a standard that allows long filenames on FAT partitions.
xfs	SGI's journaled filesystem (ported from Irix).

10.14 mount: Mounts a Filesystem

The mount utility connects directory hierarchies—typically filesystems—to the Linux directory hierarchy. These directory hierarchies can be on remote and local disks, CDs, and floppy diskettes. Linux also allows you to mount virtual filesystems that have been built inside ordinary files, filesystems built for other operating systems, and the special /proc filesystem, which maps useful Linux kernel information to a pseudodirectory.

Mount point:

The mount point for the filesystem/directory hierarchy that you are mounting is a directory in the local filesystem. This directory must exist before you can mount a filesystem; its contents disappear as long as a filesystem is mounted on it and reappear when you unmount the filesystem. Without any arguments, mount lists the currently mounted filesystems, showing the physical device holding each filesystem, the mount point, the type of filesystem, and any options set when each filesystem was mounted:

```
$ mount
proc on /proc type proc (rw)
/dev/hdb1 on / type ext2 (rw)
/dev/hdb4 on /tmp type ext2 (rw)
/dev/hda5 on /usr type ext3 (rw)
/dev/sda1 on /usr/X386 type ext2 (rw)
/dev/sda3 on /usr/local type ext2 (rw)
/dev/hdb3 on /home type ext3 (rw)
/dev/hda1 on /dos type msdos (rw,umask=000)
tuna:/p04 on /p04 type nfs (rw,addr=192.168.0.8)
/dev/scd0 on /mnt/cdrom type iso9660 (ro,noexec,nosuid,nodev)
```

The mount utility gets this information from the /etc/mtab file. The first entry in the preceding example shows the /proc pseudofilesystem. The next six entries identify disk partitions holding standard Linux ext2 and ext3 filesystems. These partitions are found on three disks: two IDE disks (hda, hdb) and one SCSI disk (sda). Disk partition /dev/hda1 has a DOS (msdos) filesystem mounted at the directory /dos in the Linux filesystem. You can access the DOS files and directories on this partition as if they were Linux files and directories, using Linux utilities and applications. The line starting with tuna shows a mounted, remote NFS filesystem. The last line shows a CD mounted on a SCSI CD drive (/dev/scd0).

Do not mount anything on root (/):

Always mount network directory hierarchies and removable devices at least one level below the root level of the filesystem. The root filesystem is mounted on /; you cannot mount two filesystems in the same place. If you were to try to mount something on /, all files, directories, and filesystems that were under the root directory would no longer be available, and the system would crash.

Mount Options

The mount utility takes many options, which you can specify on the command line or in the /etc/fstab file. For a complete list of mount options for local filesystems, see the mount man page; for remote directory hierarchies, see the nfs man page. The **noauto** option causes Linux not to mount the filesystem automatically. The **nosuid** option forces mounted setuid executables to run with regular permissions (no effective user ID change) on the local system (the system that mounted the filesystem). Unless you specify the user, users, or owner option, only Superuser can mount and unmount a filesystem. The user option means that any user can mount the filesystem, but the filesystem can be unmounted only by the same user who mounted it; users means that any user can mount and unmount the filesystem. These options are frequently specified for CD and floppy drives. The owner option, which is used only under special circumstances, is similar to the user option except that the user mounting the device must own the device.

Mounting a Linux Floppy Diskette

Mounting a Linux floppy diskette is similar to mounting a partition of a hard disk. Put an entry similar to the following in /etc/fstab for a diskette in the first floppy drive:

```
/dev/fd0 /mnt/floppy auto noauto,users 0 0
```

Specifying a filesystem type of auto causes the system to probe the filesystem to determine its type and allows users to mount a variety of diskettes. Create the /mnt/floppy directory if necessary. Insert a diskette and try to mount it. The diskette must be formatted (use fdformat). Use mkfs to create a filesystem—but be careful, because mkfs destroys all data on the diskette: **# mount /dev/fd0**

```
mount: you must specify the filesystem type
# mkfs /dev/fd0
mke2fs 1.41.9 (22-Aug-2009)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
184 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=1572864
1 block group
8192 blocks per group, 8192 fragments per group
184 inodes per group
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

Try the mount command again:

```
# mount /dev/fd0
# mount
...
/dev/fd0 on /mnt/floppy type ext2 (rw,noexec,nosuid,nodev)
# df -h /dev/fd0
Filesystem Size Used Avail Use% Mounted on
/dev/fd0 1.4M 19K 1.3M 2% /mnt/floppy
```

The mount command without any arguments and df –h /dev/fd0 show the floppy is mounted and ready for use.

umount: Unmounts a Filesystem

The **umount** utility unmounts a filesystem as long as it does not contain any files or directories that are in use (open). For example, a logged-in user's working directory must not be on the filesystem you want to unmount. The next command unmounts the CD mounted earlier: $ **umount /mnt/cdrom.** Unmount a floppy or a remote directory hierarchy the same way you would unmount a partition of a hard drive. The umount utility consults /etc/fstab to get the necessary information and then unmounts the appropriate filesystem from its server. When a process has a file open on the filesystem that you are trying to unmount, umount displays a message similar to the following:

```
umount: /home: device is busy
```

Use the –a option to umount to unmount all filesystems, except for the one mounted at /, which can never be unmounted. You can combine –a with the –t option to unmount filesystems of a given type (ext3, nfs, or others). For example, the following command unmounts all mounted nfs directory hierarchies that are not being used: # umount -at nfs

fstab: Keeps Track of Filesystems

The system administrator maintains the /etc/fstab file, which lists local and remote directory hierarchies, most of which the system mounts automatically when it boots. The fstab file has six columns, where a hyphen is a placeholder for a column that has no value:

1) Name—The name, label, or UUID number of a local block device or a pointer to a remote directory hierarchy. When you install the system, Fedora/RHEL uses UUID numbers for fixed devices. It prefaces each line in fstab that specifies a UUID with a comment that specifies the device name. Using UUID numbers in fstab during installation circumvents the need for consistent device naming.

2) Mount point—The name of the directory file that the filesystem/directory hierarchy is to be mounted on. If it does not already exist, create this directory using mkdir.

3) Type—The type of filesystem/directory hierarchy that is to be mounted. Local filesystems are generally of type ext2, ext3, or iso9660, and remote directory hierarchies are of type nfs or cifs.

4) Mount options—A comma-separated list of mount options, such as whether the filesystem is mounted for reading and writing (rw, the default) or readonly (ro).

5) Dump—Used by dump to determine when to back up the filesystem.

6) Fsck—Specifies the order in which fsck checks filesystems. Root (/) should have a 1 in this column. Filesystems that are mounted to a directory just below the root directory should have a 2. Filesystems that are mounted on another mounted filesystem (other than root) should have a 3.

The following example shows a typical **fstab file:**

```
        # cat /etc/fstab
    LABEL=/1 / ext3 defaults 1 1
    LABEL=/boot1 /boot ext3 defaults 1 2
    devpts /dev/pts devpts gid=5,mode=620 0 0
    tmpfs /dev/shm tmpfs defaults 0 0
    LABEL=/home1 /home ext3 defaults 1 2
    proc /proc proc defaults 0 0
    sysfs /sys sysfs defaults 0 0
    LABEL=SWAP-hda5 swap swap defaults 0 0
    /dev/hda3 /oldhome ext3 defaults 0 0
    tuna:/p04 /p04 nfs defaults 0 0
    /dev/fd0 /mnt/floppy auto noauto,users 0 0
```

fsck: Checks Filesystem Integrity

The **fsck (filesystem check)** utility verifies the integrity of filesystems and, if possible, repairs any problems it finds. Because many filesystem repairs can destroy data, particularly on a nonjournaling filesystem, such as ext2, by default fsck asks you for confirmation before making each repair. The following command checks all unmounted filesystems that are marked to be checked in /etc/fstab except for the root filesystem: **# fsck -AR**

The –A option causes fsck to check filesystems listed in fstab. The –R option checks the same filesystems as –A except it does not check the root filesystem. You can check a specific filesystem with a command similar to one of the following:

# fsck /home

or

# fsck /dev/sda6

Crash flag: The /etc/rc.d/rc.sysinit start-up script looks for two flags in the root directory of each partition to determine whether fsck needs to be run on that partition before it is mounted. The .autofsck flag (the crash flag) indicates that the partition should be checked.

tune2fs: Changes Filesystem Parameters

The tune2fs utility displays and modifies filesystem parameters on ext2, ext3, and ext4 filesystems. This utility can also set up journaling on an ext2 filesystem, turning it into an ext3 filesystem. With the introduction of increasingly more reliable hardware and software, systems are rebooted less frequently, so it is important to check filesystems regularly. By default, fsck is run on each partition while the system is brought up, before the partition is mounted. Depending on the flags, fsck may do nothing more than display a message saying that the filesystem is clean. The larger the partition, the more time it takes to check it, assuming a nonjournaling filesystem. These checks are often unnecessary. The tune2fs utility helps you to find a happy medium between checking filesystems each time you reboot the system and never checking them. It does so by scheduling when fsck checks a filesystem (these checks occur only when the system is booted). You can use two scheduling patterns: time elapsed since the last check and number of mounts since the last check. The following command causes /dev/sda6 to be checked when fsck runs after it has been mounted eight times or after 15 days have elapsed since its last check, whichever happens first:

```
# tune2fs -c 8 -i 15 /dev/sda6
tune2fs 1.41.9 (22-Aug-2009)
Setting maximal mount count to 8
Setting interval between checks to 1296000 seconds
```

The next tune2fs command is similar but works on a different partition and sets the current mount count to 4. When you do not specify a current mount count, it is set to zero:

```
# tune2fs -c 8 -i 15 -C 4 /dev/sda6
tune2fs 1.41.9 (22-Aug-2009)
Setting maximal mount count to 8
Setting current mount count to 4
Setting interval between checks to 1296000 seconds
```

RAID Filesystem

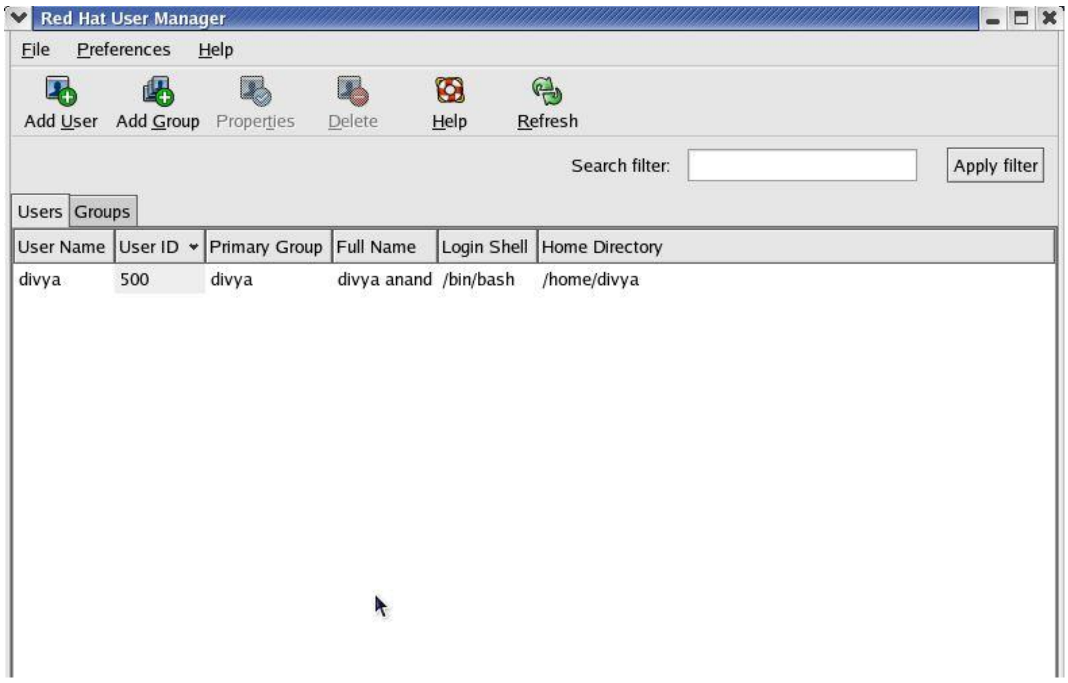
RAID (Redundant Arrays of Inexpensive/Independent Disks) spreads information across several disks to combine several physical disks into one larger virtual device. RAID improves performance and creates redundancy. More than six types of RAID configurations exist. Using Fedora/RHEL tools, you can set up software RAID. Hardware RAID requires hardware that is designed to implement RAID. RAID can be an effective addition to a backup. Fedora/RHEL offers RAID software that you can install either when you install a Fedora/RHEL system or as an afterthought. The Linux kernel can automatically detect RAID disk partitions at boot time if the partition ID is set to 0xfd, which fdisk recognizes as Linux **raid autodetect.** Software RAID, as implemented in the kernel, is much cheaper than hardware RAID. Not only does this software avoid specialized RAID disk controllers, but it also works with the less expensive IDE disks as well as SCSI disks.

10.15 Configuring User and Group Accounts

More than a username is required for a user to be able to log in and use a system. At a minimum a user must have an entry in the **/etc/passwd** and **/etc/shadow** files and a **home directory.**

system-config-users: Manages User Accounts

The **system-config-users** utility displays the User Manager window and enables you to add, delete, and modify system users and groups. To display the User Manager window, enter **system-config-users** on a command line or select **Main menu: System | Administration | Users and Groups**.



User Name	User ID	Primary Group	Full Name	Login Shell	Home Directory
divya	500	divya	divya anand	/bin/bash	/home/divya

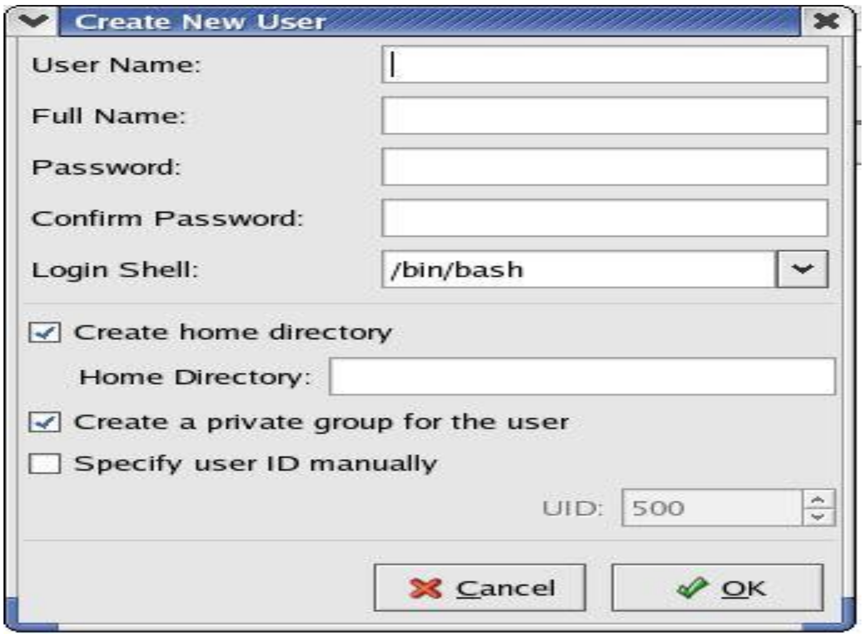
This window has two tabs: **Users and Groups**, where each tab displays information appropriate to its name.

Search filter

The **Search filter**, located just below the toolbar, selects users or groups whose names match the string, which can include wildcards, that you enter in the Search filter text box. The string matches the beginning of a name. For example, *nob matches nobody and nfsnobody, whereas nob matches only nobody.

Adding a user

To create a new user, click the **Add User button** on the toolbar. The User Manager displays the Create New User window. Enter the information for the new user and click OK.


Create New User

User Name:

Full Name:

Password:

Confirm Password:

Login Shell: /bin/bash

[✓] Create home directory

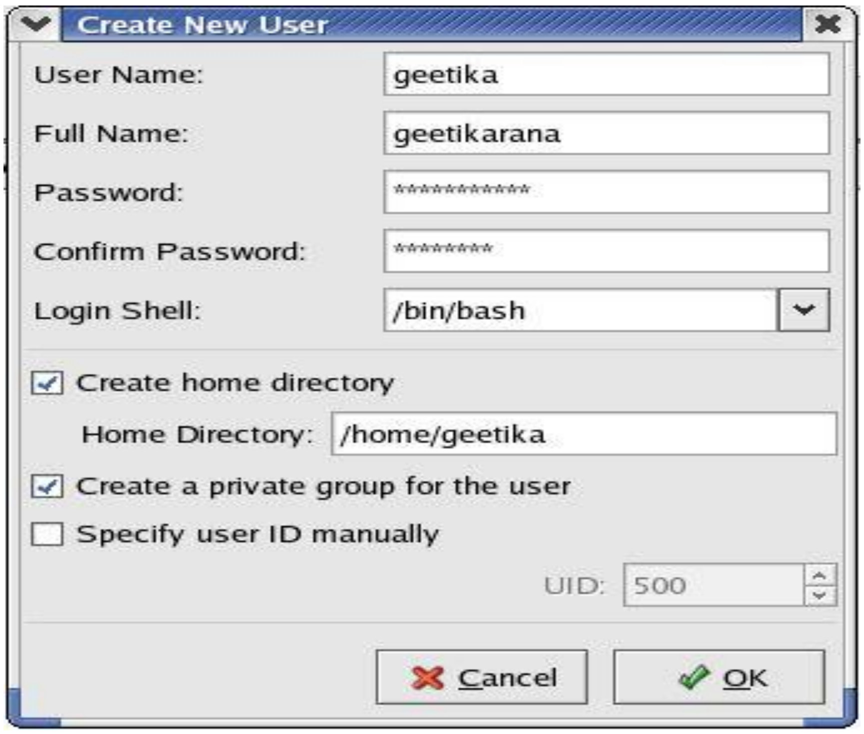
Home Directory:

[✓] Create a private group for the user

[ ] Specify user ID manually

UID: 500

Cancel OK


Create New User

User Name: geetika

Full Name: geetikarana

Password: **************

Confirm Password: **********

Login Shell: /bin/bash

[✓] Create home directory

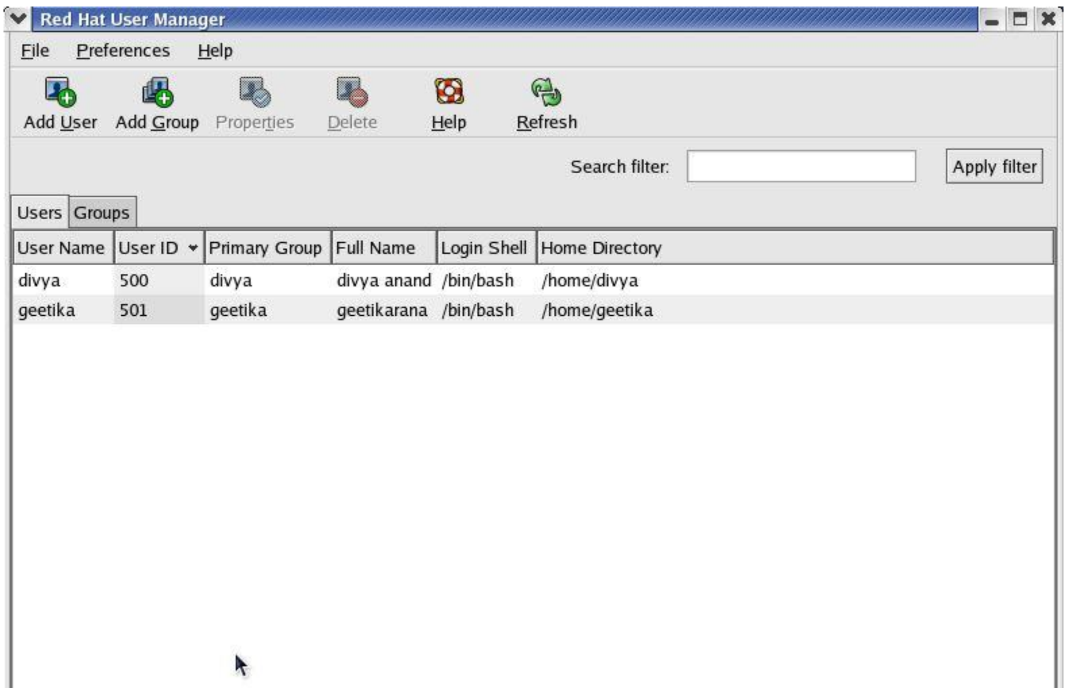
Home Directory: /home/geetika

[✓] Create a private group for the user

[ ] Specify user ID manually

UID: 500

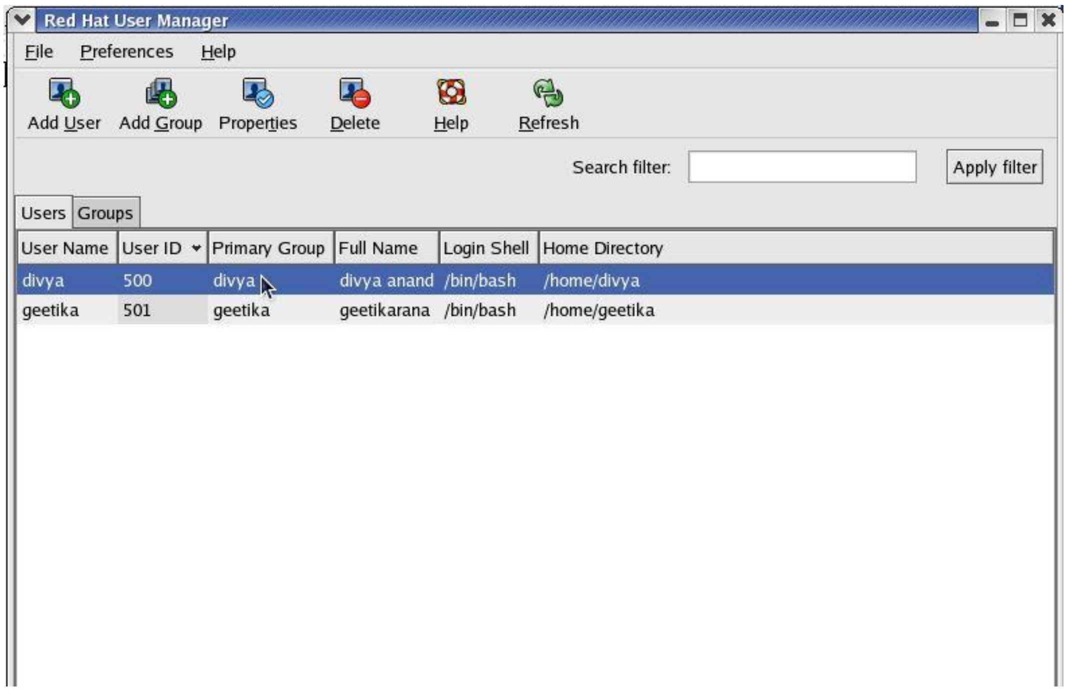
Cancel OK



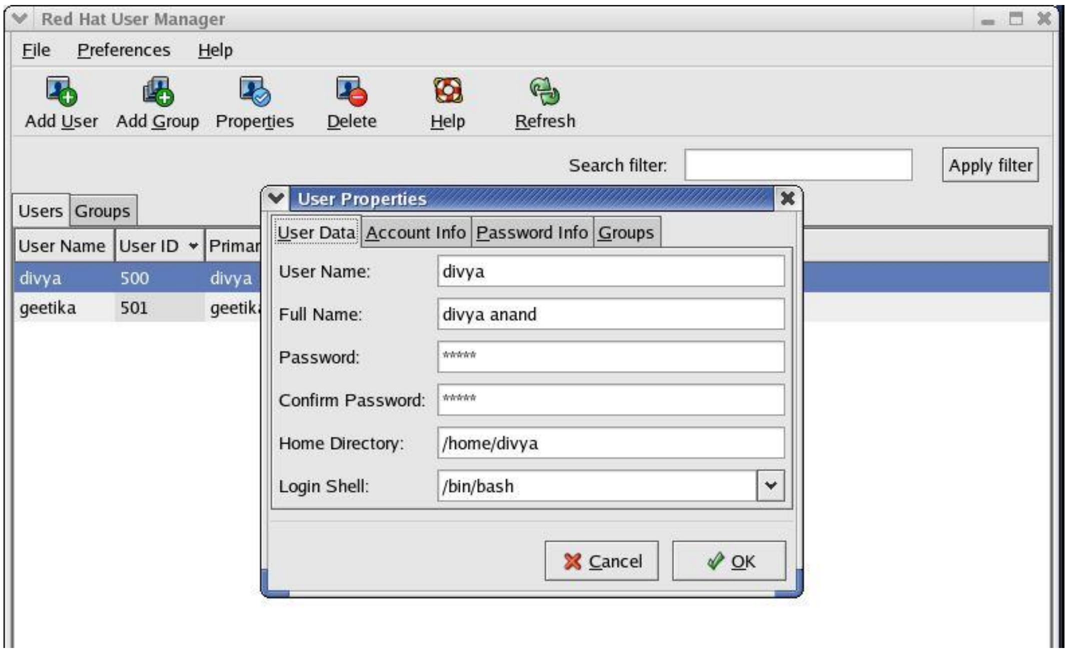
User Name	User ID	Primary Group	Full Name	Login Shell	Home Directory
divya	500	divya	divya anand	/bin/bash	/home/divya
geetika	501	geetika	geetikarana	/bin/bash	/home/geetika

Modifying a user

Once you create a user, you can **modify the user** to add/change/remove information. To modify a user, highlight the user in the User Manager window and click Properties on the toolbar; the utility displays the User Properties window.



User Name	User ID	Primary Group	Full Name	Login Shell	Home Directory
divya	500	divya	divya anand	/bin/bash	/home/divya
geetika	501	geetika	geetikarana	/bin/bash	/home/geetika



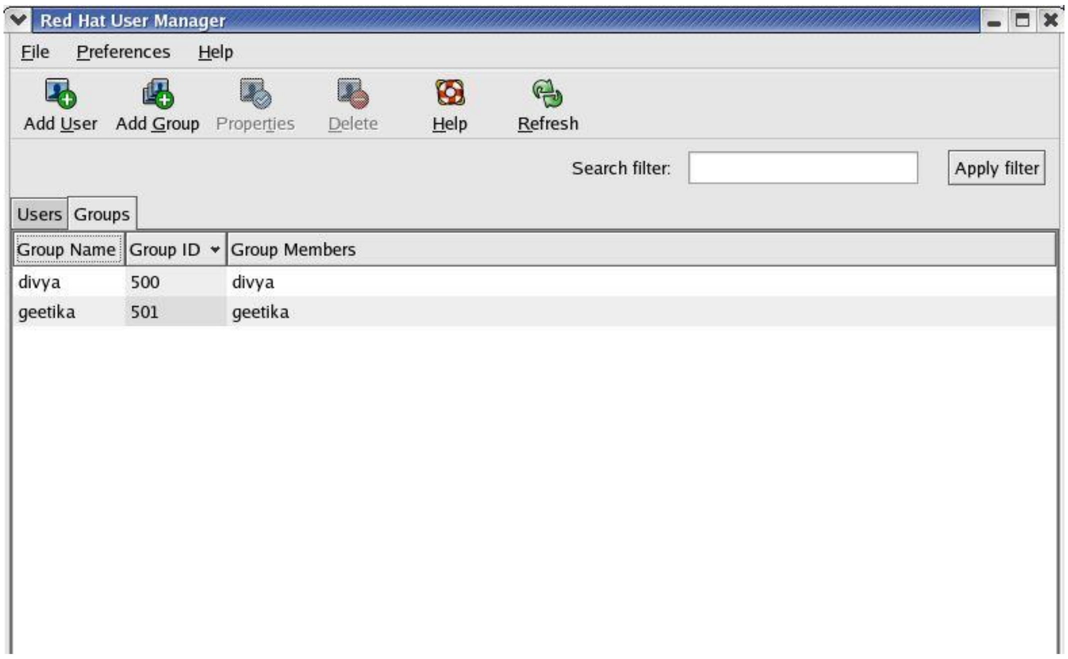
User Name: divya
Full Name: divya anand
Password: *****
Confirm Password: *****
Home Directory: /home/divya
Login Shell: /bin/bash

The User Properties window has four tabs: User Data, Account Info, Password Info, and Groups.

- **User Data tab**: It holds basic user information such as name and password.
- **Account Info tab**: It allows you to specify an expiration date for the account and to lock the account so the user cannot log in.
- **Password Info tab**: It allows you to turn on password expiration and specify various related parameters.
- **Groups tab**: You can specify the groups that the user is a member of.

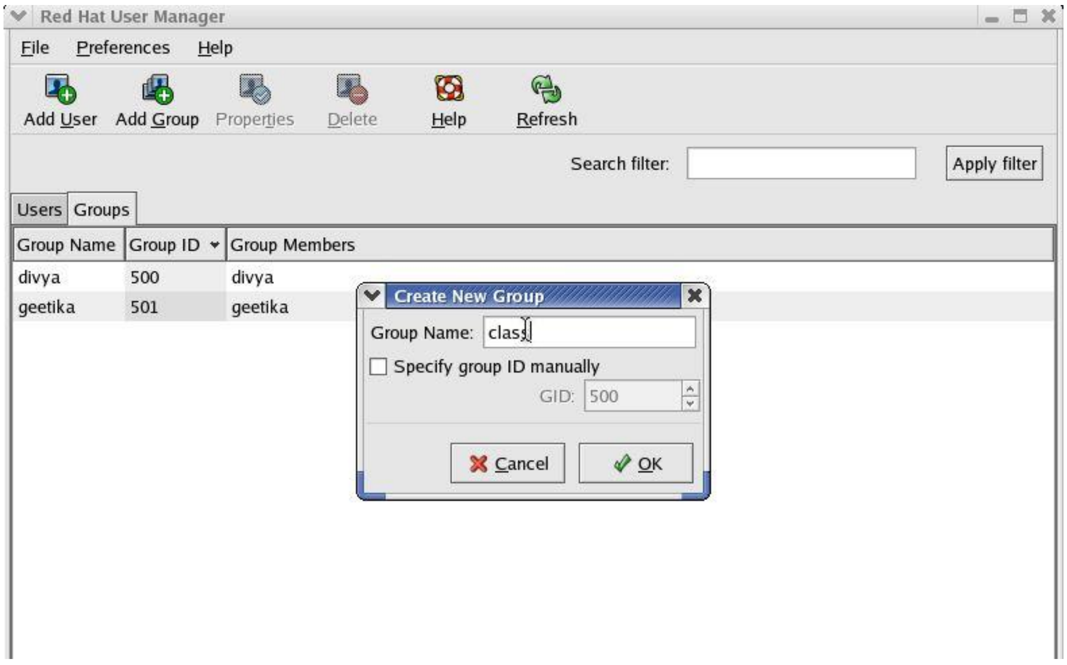
Working with groups

- Click the **Groups tab** in the User Manager window to work with groups.



Group Name	Group ID	Group Members
divya	500	divya
geetika	501	geetika

- To create a group, click **Add Group** on the toolbar and specify the name of the group.

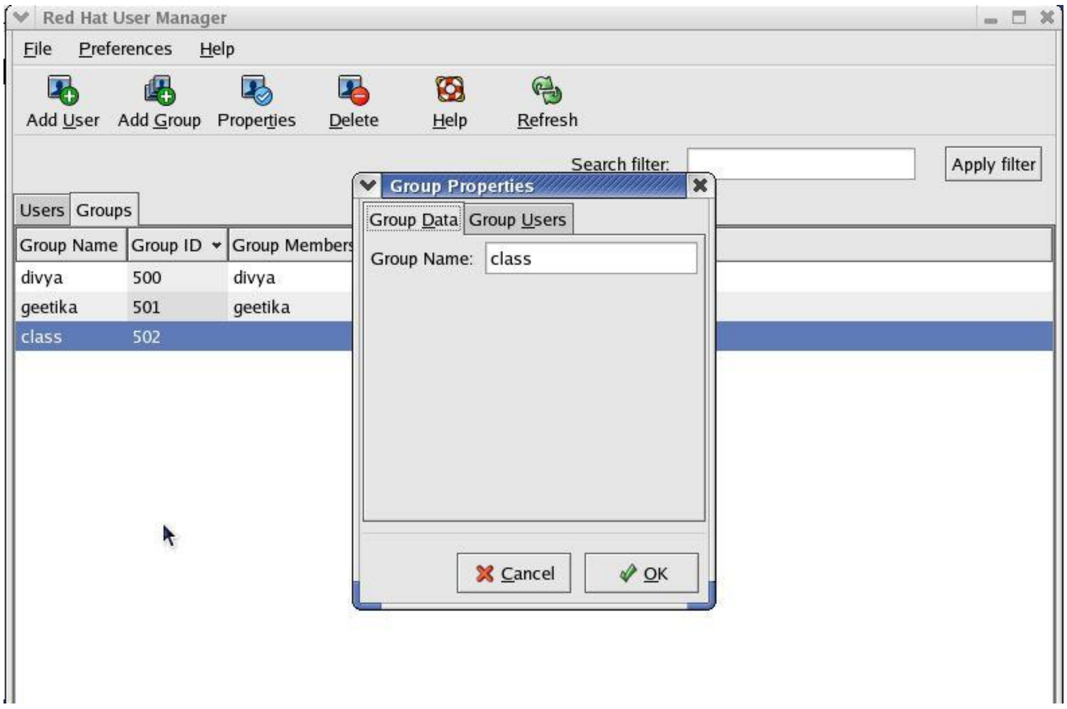


Group Name	Group ID	Group Members
divya	500	divya
geetika	501	geetika

Group Name: class

GID: 500

- To **change the name of a group or to add or remove users** from a group, highlight the group and click Properties on the toolbar.

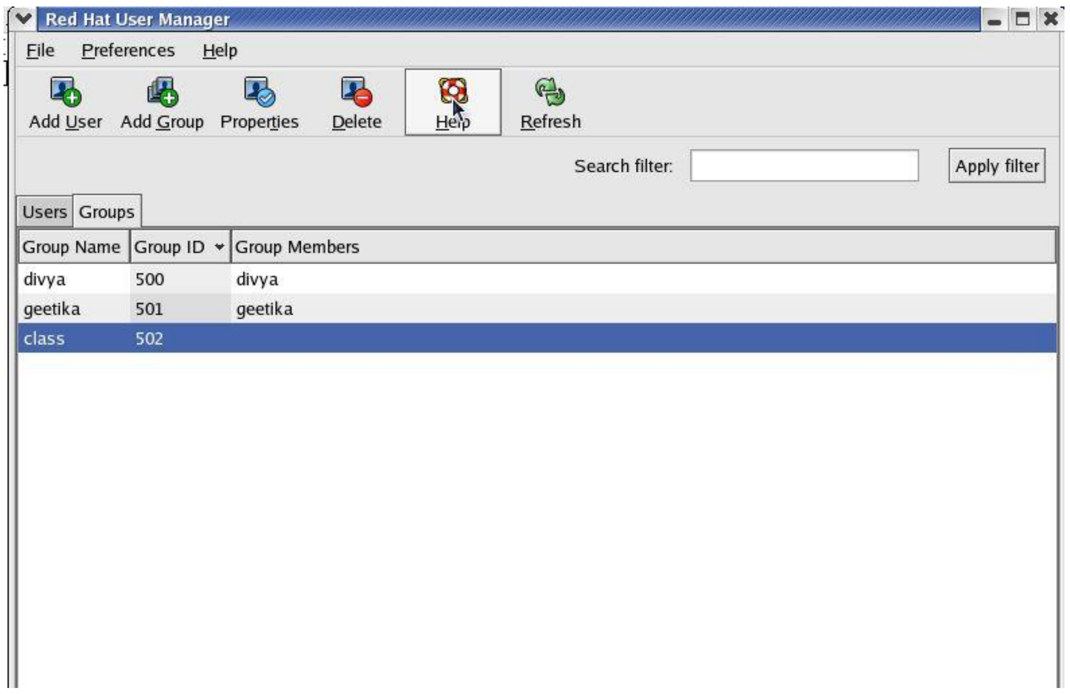


Group Name	Group ID	Group Members
divya	500	divya
geetika	501	geetika
class	502	

Group Name: class

Help:

- The User Manager provides extensive help. To access it, click Help on the toolbar.
- When you are done working with users and groups, close the window.



Group Name	Group ID	Group Members
divya	500	divya
geetika	501	geetika
class	502	

useradd: Adds a User Account

The **useradd utility** adds a new user account to the system. By default, useradd assigns the next highest unused user ID to a new account and specifies bash as the user's login shell. This command creates the user's home directory (in /home), specifies the user's group ID, and puts the user's full name in the comment field: **# useradd -g 500 -c "Alex Watson" alex**

userdel: Removes a User Account

The **userdel utility** deletes user accounts. This command removes alex's account and his home directory hierarchy: **# userdel -r alex.** To turn off a user's account temporarily, you can use **usermod** to change the expiration date for the account. Because it specifies that his account expired in the past (December 31, 2009), the following command line prevents alex from logging in:

# usermod -e "12/31/09" alex

groupadd: Adds a Group

Just as useradd adds a new user to the system, **groupadd** adds a new group by adding an entry for it in **/etc/group.** creates a new group named rtfm: **# groupadd -g 1024 rtfm**

Unless you use the **–g** option to assign a group ID, the system picks the next available sequential number greater than 500. The **–o** option allows the group ID to be nonunique if you want to have multiple names for the same group ID.

10.16 Backing Up Files

The backup copies are vital in three instances:

1) When the system malfunctions and files are lost,
2) When a catastrophic disaster (fire, earthquake, and so on) occurs,
3) When a user or the system administrator deletes or corrupts a file by accident.

Even when you set up RAID, you still need to back up files. Although **RAID provides fault tolerance** (helpful in the event of disk failure), it does not help when a catastrophic disaster occurs or when a file is corrupted or accidentally removed. You must back up filesystems on a regular basis. Backup files are usually kept on magnetic tape or some other removable media. A **full backup** makes copies of all files, regardless of when they were created or accessed. An **incremental backup** makes copies of those files that have been created or modified since the last (usually full) backup. The more people using the system, the more often you should back up the filesystems. One popular schedule is to perform an incremental backup one or two times a day and a full backup one or two times a week.

Choosing a Backup Medium

If the local system is connected to a network, you can write your backups to a tape drive on another system. This technique is often used with networked computers to avoid the cost of having a tape drive on each computer in the network and to simplify management of backing up many computers in a network. Most likely you want to use a tape system for backups. Other options for holding backups are writable CDs, DVDs, and removable hard disks. These devices, although not as cost-effective or able to store as much information as tape systems, offer convenience and improved performance over using tapes.

Backup Utilities

A number of utilities help you back up the system, and most work with any media. Most Linux backup utilities are based on one of the archive programs—**tar or cpio**— and augment these basic programs with bookkeeping support for managing backups conveniently. You can use any of the **tar, cpio, or dump/restore** utilities to construct full or partial backups of the system. Each utility constructs a large file that contains, or archives, other files. In addition to file contents, an archive includes header information for each file it holds.

The **amanda utility (Advanced Maryland Automatic Network Disk Archiver)**, one of the more popular backup systems, uses dump or tar and takes advantage of Samba to back up Windows systems. The amanda utility backs up a LAN of heterogeneous hosts to a single tape drive. You can use yum to install amanda.

tar: Archives Files

The **tar (tape archive)** utility stores and retrieves files from an archive and can compress the archive to conserve space. If you do not specify an archive device, tar uses standard output and standard input. For displaying the options: **# tar --help | less**

It combines single-letter options into a single command-line argument: **# tar –ztvf /dev/st0**. This command uses descriptive words for the same options: **# tar --gzip --list --verbose --file /dev/st0**. Both commands tell tar to generate a (v, verbose) table of contents (t, list) from the tape on /dev/st0 (f, file), using gzip (z, gzip) to decompress the files.

Option	Effect
--append (–r)	Appends files to an archive
--catenate (–A)	Adds one or more archives to the end of an existing archive
--create (–c)	Creates a new archive
--delete	Deletes files in an archive (not on tapes)
--diff (–d)	Compares files in an archive with disk files **--extract**

--extract (–x)	Extracts files from an archive
--help	Displays a help list of tar options
--list (–t)	Lists the files in an archive
--update (–u)	Like the –r option, but the file is not appended if a newer version is already in the archive

cpio: Archives Files

The **cpio (copy in/out)** program is similar to tar but can use archive files in a variety of formats, including the one used by tar. Normally cpio reads the names of the files to insert into the archive from standard input and produces the archive file as standard output. When extracting files from an archive, cpio reads the archive as standard input.

Performing a Simple Backup

When you prepare to make a major change to a system, such as replacing a disk drive or updating the Linux kernel, it is a good idea to archive some or all of the files so you can restore any that become damaged if something goes wrong. For this type of backup, tar or cpio works well. For example, if you have a SCSI tape drive as device **/dev/st0** that is capable of holding all the files on a single tape, you can use the following commands to construct a backup tape of the entire system:

```
# cd /
# tar –cf /dev/st0 .
```

The tar command then creates an archive (**c**) on the device **/dev/st0 (f). # tar –cjf /dev/st0 .** You can back up the system with a combination of find and cpio. These commands create an output file and set the I/O block size to 5120 bytes (the default is 512 bytes):

```
# cd /
# find . –depth | cpio –oB > /dev/st0
```

This command restores the files in the /home directory from the preceding backup. The options extract files from an archive (–i) in verbose mode, keeping the modification times and creating directories as needed.

```
# cd /
# cpio –ivmd /home/\* < /dev/st0
```

dump, restore: Back Up and Restore Filesystems

The **dump utility**, which first appeared in UNIX version 6, backs up either an entire filesystem or only those files that have changed since the last dump. The **restore utility** restores an entire filesystem, an individual file, or a directory hierarchy. This command backs up all files (including directories and special files) on the root (/) partition to SCSI tape 0. Frequently there is a link to the active tape drive, named /dev/tape, which you can use in place of the actual entry in the /dev directory.

```
        # dump -0uf /dev/st0 /
```

The option specifies that the entire filesystem is to be backed up (a full backup). There are ten dump levels: 0–9. Zero is the highest (most complete) level and always backs up the entire filesystem. Each additional level is incremental with respect to the level above it. For example, 1 is incremental to 0 and backs up only files that have changed since the last level 0 dump; 2 is incremental to 1 and backs up only files that have changed since the last level 1 dump; and so on. The **u option** updates the **/etc/dumpdates** file with filesystem, date, and dump level information for use by the next incremental dump.

- The **f option** and its argument write the backup to the device named **/dev/st0.**
- The **u option** updates the **/etc/dumpdates** file with filesystem, date, and dump level information for use by the next incremental dump.
- The **f option** and its argument write the backup to the device named **/dev/st0.** This command makes a partial backup containing all files that have changed since the last level 0 dump. The first argument is a 1, specifying a level 1 dump:

  # dump -1uf /dev/st0 /
- To restore an entire filesystem from a tape, first restore the most recent complete (level 0) backup. Perform this operation carefully because restore can overwrite the existing filesystem. When you are logged in as Superuser, cd to the directory the filesystem is mounted on and give this command: **# restore -if /dev/st0**

  i: Interactive mode

  f: specifies the name of the device that the backup medium is mounted on.

10.17 Scheduling Tasks

It is a good practice to schedule certain routine tasks to run automatically. For example, you may want to remove old core files once a week, summarize accounting data daily, and rotate system log files monthly.

crond and crontab: Schedule Routine Tasks

Using **crontab**, you can submit a list of commands in a format that can be read and executed by **crond**. Working as **Superuser**, you can put commands in one of the **/etc/cron.*** directories to be run at intervals specified by the directory name, such as cron.daily.

at: Runs Occasional Tasks

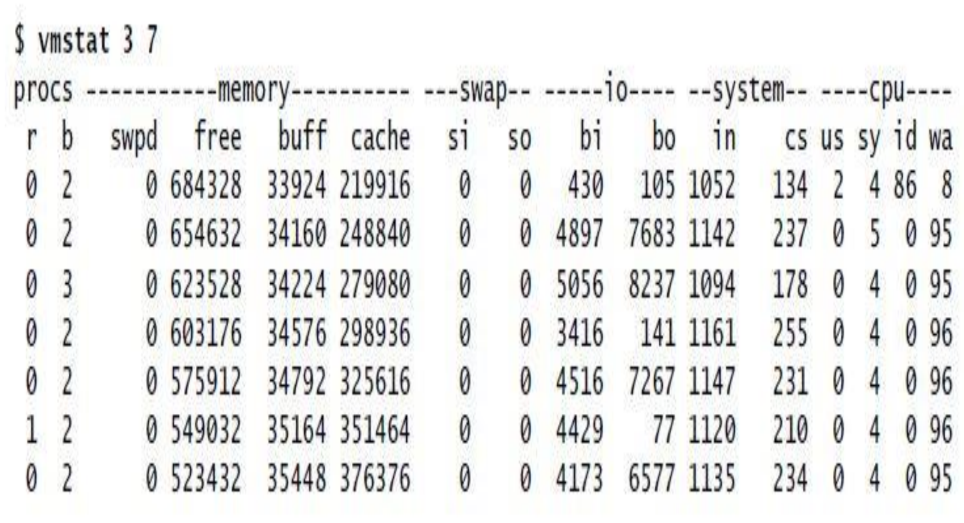
Like the cron utility, at allows you to run a job sometime in the future. Unlike cron, at runs a job only once.

System Reports

Many utilities report on one thing or another. The **who, finger, ls, ps**, and other utilities generate simple end-user reports. In some cases, these reports can help with system administration.

vmstat: Reports Virtual Memory Statistics

The **vmstat utility** (procps package) generates virtual memory information along with (limited) disk and CPU activity data.



```
$ vmstat 3 7
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 0  2      0 684328  33924 219916    0    0   430   105 1052   134  2  4 86  8
 0  2      0 654632  34160 248840    0    0  4897  7683 1142   237  0  5  0 95
 0  3      0 623528  34224 279080    0    0  5056  8237 1094   178  0  4  0 95
 0  2      0 603176  34576 298936    0    0  3416   141 1161   255  0  4  0 96
 0  2      0 575912  34792 325616    0    0  4516  7267 1147   231  0  4  0 96
 1  2      0 549032  35164 351464    0    0  4429    77 1120   210  0  4  0 96
 0  2      0 523432  35448 376376    0    0  4173  6577 1135   234  0  4  0 95
```

vmstat column heads

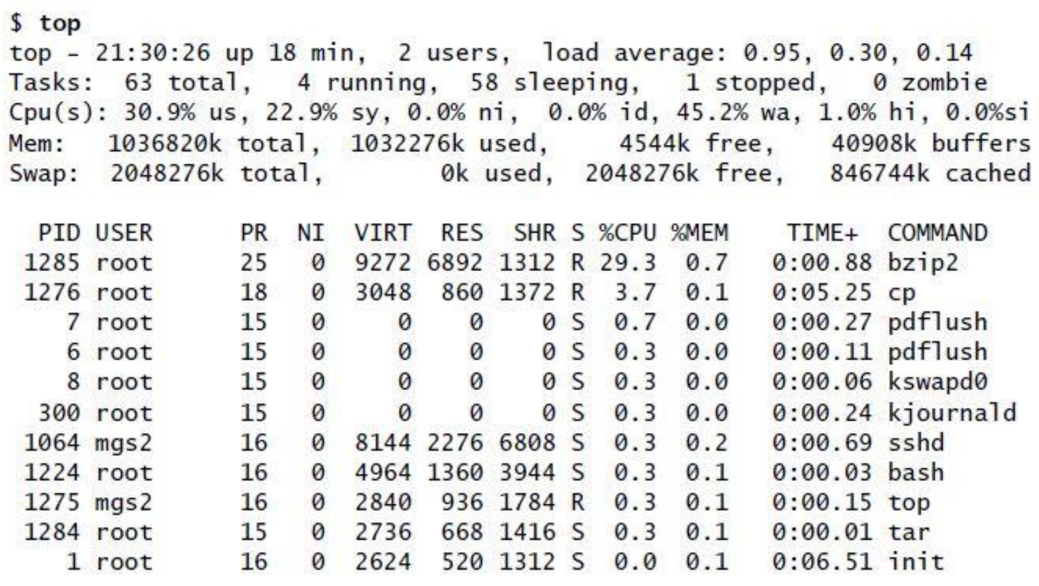
This shows virtual memory statistics in 3-second intervals for seven iterations. The first line covers the time since the system was last booted; the rest of the lines cover the period since the previous line.

procs	Process information
	r Number of waiting, runnable processes
	b Number of blocked processes (in uninterruptable sleep)
Memory	Memory Information in Kilobytes
	swpd Used virtual memory
	free Idle memory
	buff Memory used as buffers
	cache Memory used as cache
swap	System paging activity in kilobytes per second
	si Memory swapped in from disk
	so Memory swapped out to disk
io	System I/O activity in blocks per second
	bi Blocks received from a block device

	bo Blocks sent to a block device
system	Values are per second
	in Interrupts (including the clock)
	cs Context switches
cpu	Percentage of total CPU time spent in each of these states
	us User (nonkernel)
	sy System (kernel)
	id Idle
	wa Waiting for I/O

top: Lists Processes Using the Most Resources

The **top utility** is a useful supplement to ps. At its simplest, top displays system information at the top and the most CPU-intensive processes below the system information. The top utility updates itself periodically; type **q to quit.**



```
$ top
top - 21:30:26 up 18 min,  2 users,  load average: 0.95, 0.30, 0.14
Tasks:  63 total,   4 running,  58 sleeping,   1 stopped,   0 zombie
Cpu(s): 30.9% us, 22.9% sy, 0.0% ni,  0.0% id, 45.2% wa, 1.0% hi, 0.0%si
Mem:   1036820k total,  1032276k used,     4544k free,    40908k buffers
Swap:  2048276k total,        0k used,  2048276k free,   846744k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1285 root      25   0  9272 6892 1312 R 29.3  0.7   0:00.88 bzip2
 1276 root      18   0  3048  860 1372 R  3.7  0.1   0:05.25 cp
    7 root      15   0     0    0    0 S  0.7  0.0   0:00.27 pdflush
    6 root      15   0     0    0    0 S  0.3  0.0   0:00.11 pdflush
    8 root      15   0     0    0    0 S  0.3  0.0   0:00.06 kswapd0
  300 root      15   0     0    0    0 S  0.3  0.0   0:00.24 kjournald
 1064 mgs2      16   0  8144 2276 6808 S  0.3  0.2   0:00.69 sshd
 1224 root      16   0  4964 1360 3944 S  0.3  0.1   0:00.03 bash
 1275 mgs2      16   0  2840  936 1784 R  0.3  0.1   0:00.15 top
 1284 root      15   0  2736  668 1416 S  0.3  0.1   0:00.01 tar
    1 root      16   0  2624  520 1312 S  0.0  0.1   0:06.51 init
```

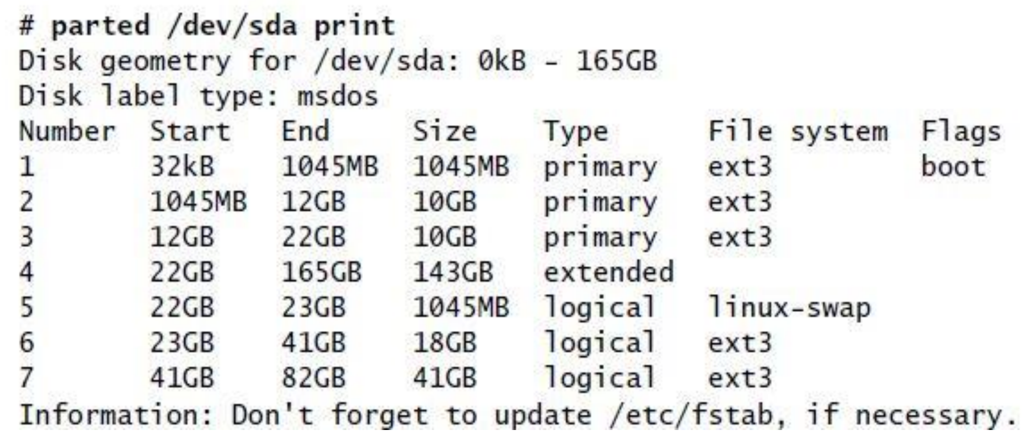
top: interactive commands

Command	**Function**
F	Specify a sort field.
h or ?	Displays a Help screen.
k	Prompts for a PID number and type of signal and sends the process that signal. Defaults to signal 15 (SIGTERM); specify 9 (SIGKILL) only when 15 does not work.
M	Sorts processes by memory usage.
O	Specify a sort field.
P	Sorts processes by CPU usage (default).
q	Quits.
S	Prompts for time between updates in seconds. Use 0 for continuous updates.
SPACE	Updates the display immediately.
T	Sorts tasks by time.
W	Writes a startup file named ~/.toprc so that next time you start top, it uses the same parameters it is currently using.

parted: Reports on and Partitions a Hard Disk

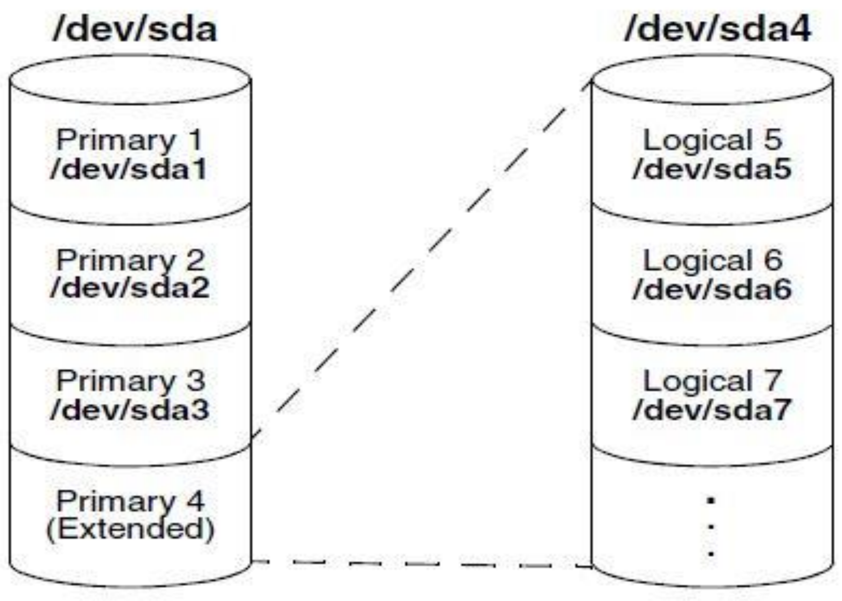
- The **parted (partition editor) utility** reports on and manipulates hard disk partitions.

The following example shows how to use parted from the command line



```
# parted /dev/sda print
Disk geometry for /dev/sda: 0kB - 165GB
Disk label type: msdos
Number  Start   End     Size    Type      File system  Flags
1       32kB    1045MB  1045MB  primary   ext3         boot
2       1045MB  12GB    10GB    primary   ext3
3       12GB    22GB    10GB    primary   ext3
4       22GB    165GB   143GB   extended
5       22GB    23GB    1045MB  logical   linux-swap
6       23GB    41GB    18GB    logical   ext3
7       41GB    82GB    41GB    logical   ext3
Information: Don't forget to update /etc/fstab, if necessary.
```

Partitions



parted: Reports on and Partitions a Hard Disk

The print command displays the following columns:

- **Number—**The minor device number of the device holding the partition. This number is the same as the last number in the device name. In the example, 5 corresponds to **/dev/sda5**.
- **Start—**The location on the disk where the partition starts. The parted utility specifies a location on the disk as the distance (in bytes) from the beginning of the disk. Thus partition 3 starts 12 gigabytes from the beginning of the disk.
- **End—**The location on the disk where the partition stops. Although partition 2 ends 12 gigabytes from the beginning of the disk and partition 3 starts at the same location, parted takes care that the partitions do not overlap at this single byte.
- **Size—**The size of the partition in kilobytes (kB), megabytes (MB), or gigabytes (GB).
- **Type—**The partition type: primary, extended, or logical.
- **File system—**The filesystem type: ext2, ext3, fat32, linux-swap, and so on.
- **Flags—**The flags that are turned on for the partition, including boot, raid, and lvm. In the example, partition 1 is bootable

Summary:

- Superuser can use certain tools, such as sudo, to give specific users permission to perform tasks that are normally reserved for Superuser.
- To bring a system up in rescue mode, boot the system from the first installation CD, the Net Boot CD, or the install DVD.
- SELinux can be in one of three states: enforcing, permissive and disabled.
- The system-config-selinux utility displays the SELinux Administration window, which controls SELinux.
- By default, Fedora systems boot to graphical multiuser mode (runlevel 5).
- The system-config-services utility displays the Service Configuration window.
- The shutdown and halt utilities perform the tasks needed to bring the system down safely.
- The key combination CONTROL-ALT-DEL Reboots the System

- Most of the Fedora/RHEL configuration tools are named system-config-*.
- When you set up a server, you frequently need to specify which clients are allowed to connect to the server. Sometimes it is convenient to specify a range of IP addresses, called a subnet.
- RHEL uses the xinetd daemon, a more secure replacement for the inetd superserver that was originally shipped with 4.3BSD.
- You may secure a server either by using TCP wrappers or by setting up a chroot jail.
- An ordinary file stores user data, such as textual information, programs, or images, such as a jpeg or tiff file.
- A character device is any device that is not a block device. Examples of character devices include printers, terminals, tape drives, and modems.

Keywords

- **System Administrator**: A system administrator should be available to help users with all types of system-related problems—from logging in to obtaining and installing software updates to tracking down and fixing obscure network issues.
- **su**: The su (substitute user) utility can create a shell or execute a program with the identity and permissions of a specified user.
- **Consolehelper:** The consolehelper utility can make it easier for someone who is logged in on the system console but not logged in as root to run system programs that normally can be run only by root.
- **Trojan House:** A Trojan horse is a program that does something destructive or disruptive to a system while appearing to be benign.
- **runlevel utiity**: The runlevel utility displays the previous and current runlevels. This utility is a transitional tool; it provides compatibility with SysVinit.
- **Booting:** Booting a system is the process of reading the Linux kernel into system memory and starting it running.
- **rpcinfo:** The rpcinfo utility displays information about programs registered with rpcbind and makes RPC calls to programs to see if they are alive.
- **Inode:** An inode is a data structure, stored on disk, that defines a file's existence and is identified by an inode number.
- **Sockets:** Sockets allow unrelated processes on the same or different computers to exchange information.
- **Hotplug:** The hotplug system allows you to plug a device into the system and use it immediately.
- **Mount point:** The mount point for the filesystem/directory hierarchy that you are mounting is a directory in the local filesystem.
- **umount:** The **umount** utility unmounts a filesystem as long as it does not contain any files or directories that are in use (open).

Self Assessment

1. su stands for

A. substitute user

B. switch user

C. substandard user

D. None of the above

2. Which of these tools gives you another user's privileges?

A. kill

B. su

C. consolehelper

D. None of the above

3. Which of these tools runs programs as a root?

A. kill

B. su

C. consolehelper

D. None of the above

4. Who is a superuser in Linux environment?

A. Root

B. Normal user

C. Machine

D. None of the above

5. What are the modes of SELinux?

A. Enforcing

B. Permissive

C. Disabled

D. All of the above

6. The policies of SELinux are

A. Targeted

B. MLS

C. Strict

D. All of the above

7. Which of these key combinations reboots the system?

A. CTRL-ALT-HOME

B. CTRL-DEL-END

C. CTRL-ALT-DEL

D. CTRL-TAB-DEL

8. Which of these utility displays the kernel ring buffer?

A. chsh

B. clear

C. dmesg

D. None of the above

9. Which of these utility creates a new filesystem on device?

A. mkfs

B. chsh

C. dmesg

D. clear

10. Which superserver listens for network connection?

A. xinted

B. Machine

C. Fedora

D. None of the above

11. How can we secure a server?

A. By using TCP wrappers

B. By setting up a chroot jail

C. By using both of the above

D. None of the above

12. The user's interactive non-login shell initialization script is located in _______ file.

A. ~/.bash_profile

B. ~/.bashrc

C. /dev

D. None of the above

13. Which of these is known as a bit bucket?

A. /dev/empty

B. /dev/bucket

C. /dev/null

D. None of the above

14. The __________ option causes Linux not to mount the filesystem automatically.

A. noauto

B. nosuid

C. nomount

D. None of the above

15. The _________ backup makes copies of all the files.

A. Full

B. Incremental

C. Decremental

D. Half

Answer for Self Assessment

1.	A	2.	B	3.	C	4.	A	5.	D
6.	D	7.	C	8.	C	9.	A	10.	A
11.	C	12.	B	13.	C	14.	A	15.	A

Review Questions

1. What is a well-maintained system? What kind of powers a superuser has?
2. How can we gain/grant the superuser privileges?
3. What are system administration tools? Explain.
4. What is a rescue mode? How can we avoid trojan horse?
5. What is security enhanced linux? What are states and policies of SELinux?
6. What is a run-level? Explain various utilities associated with this.
7. What is Fedora/RHEL configuration tools? Explain.
8. What are command line utilities?
9. What are standard rules in configuration files? How can we specify cilents?
10. How can we secure a server?
11. What are important files and directories?
12. What kind of files are supported by Linux?
13. What is fstab? How can we keep track of filesystems? Explain the columns of fstab file.
14. What is a special file in Linux?
15. Explain various types of file systems in Linux?
16. What is backing up of files? How can we choose a backup medium? Explain various backup utilities.
17. How can we schedule tasks? What are vmstat column heads?

Further Readings

Mark G. Sobell, A Practical Guide to Fedora and RedHat Enterprise Linux, Fifth Edition, Prentice Hall

Web Links

https://www.beyondtrust.com/resources/glossary/superuser-superuser-accounts#:~:text=In%20Linux%20and%20Unix%2Dlike,any%20permissions%20for%20other%20users.

https://searchsecurity.techtarget.com/definition/sudo-superuser-do

Unit 11: Web Server Configuration

CONTENTS

Objectives

After studying this unit, you will be able to

- Know what a web server is
- Understand how to set up a web server
- Know how to install a web server on Linux machine
- Understand Apache web server
- Understand the important packages of Apache web server

Introduction

A web server is nothing but software and hardware that uses the Hypertext Transfer Protocol, commonly known as HTTP, and some other protocols that respond to request from clients made on the World Wide Web. The main job that the webserver performs is to display the content of the website, which it does by storing, then processing, and eventually delivering the webpages to the user who has requested it. The web server also supports Simple Mail Transfer Protocol or SMTP and File Transfer Protocol or FTP and HTTP. These are used to transfer files for emailing and even for storage.

The web server hardware gets connected to the internet, allowing the exchange of data with the other devices related to it. The web server software controls how the user assesses the files that have been hosted. The web server process is basically a client and server model example. All the computers that host websites should have the webserver software. Web serves find use in web hosting or the hosting of data for websites as well as for all kinds of web-based applications. The web server software gets accessed using the domain name of a website. This then ensures that the content of the site gets delivered to the user who has requested it. The software part of the webserver is also made up of various components and has at least a single HTTP server. The HTTP server understands the URLs and the HTTP. The web server hardware is basically a computer that will store the webserver software as well as the files that are related to the website. These include documents, HTML, JavaScript files, and images.

Uses of Web Server

The web server is basically a part of a large internet package. It also offers many programs that are related to the intranet. The web server is used to:

- Send and receive emails.
- Download the file transfer protocol or FTP request
- Build and publish webpages.

The basic kinds of web servers can support the scripting on the server-side, which is used to employ the scripts on the webserver. This can be customized as per the request of the client. The serve side scripting works on the server machine, and this comes with a broad feature set that offers access to the database. The server-side scripting makes use of Active Server Pages or ASP, Hypertext Pre-processor or PHP, and many other scripting languages. The process also lets the HTML documents to get created.

Dynamic and Static Web Servers

A web server may be used as static or dynamic content. The static content is the one that is fixed. The static web server contains HTTP software and computer. This is static because the server sends hosted files as is present to the browser.

On the other hand, the dynamic web browser will have the webserver and the software like the database and the applications server. This is dynamic because the application server is used to update the files hosted before these are sent to the browser. The web server generates content when the database requests it. The process is flexible but complicated too.

A web server can host a single website or many websites with the help of the same software and hardware resource. This is called virtual hosting. The answer to what is the role of a web server is here**.** Web servers are also capable of limiting the speed of the response to various clients, which in turn does not allow a single client to dominate the resources. This is used to satisfy the requests of many clients. The web servers will typically host the websites that are internet accessible. These can also be used to communicate between the web clients and the servers in the local network area. This could be like through the intranet of a company. The web server could be embedded in a device like a digital camera. This lets the users communicate with the device using a commonly available web browser.

Basic Common Features

HTTP: The support for one or more versions of HTTP protocol in order to send versions of HTTP responses compatible with versions of client HTTP requests, e.g., HTTP/1.0, HTTP/1.1 plus, if available, HTTP/2, HTTP/3.

Logging: Usually web servers have also the capability of logging some information, about client requests and server responses, to log files for security and statistical purposes.

Setting up a web server

When we want to publish web pages on the Internet or on an intranet, we use a web server. In essence, a web server is an application that does two things:

1) It listens for page requests.

2) When it receives a page request, it examines the request and responds with the page that was requested.

Of course, there are many different web browsers in existence (including Mozilla, Opera, Internet Explorer, and others), and there are also a great many types of web server software. To enable a browser to request pages from a web server, they communicate using Hypertext Transfer Protocol (HTTP) − this is the standard protocol for the Internet. The request and response messages are composed using HTTP, and this is what allows any browser to request web pages from any type of web server. By default, all web servers listen for HTTP requests on port 80. Web servers also use port 443 to listen for requests made through secure HTTP connections, over SSL (secure sockets layer), through a protocol called HTTPS. So, if you want to publish your own web site, you'll need a machine with some web server software.

Why to install a web server on Red Hat Linux Machine

Well, here are two scenarios:

- First, if you're building a web site, then you'll need a web server so that you can test your site as you're developing it·
- Second, although you might not host an Internet site from your own machine, you might host an intranet site − a private web site available only to other machines inside your private network.

Commercial and open-source web servers

Commercial

- Netscape
- Iplanet
- SunONE
- Microsoft
- Zeus

Open Source

- Apache
- Thttpd
- Redhat TUX

11.1 The Apache Web Server

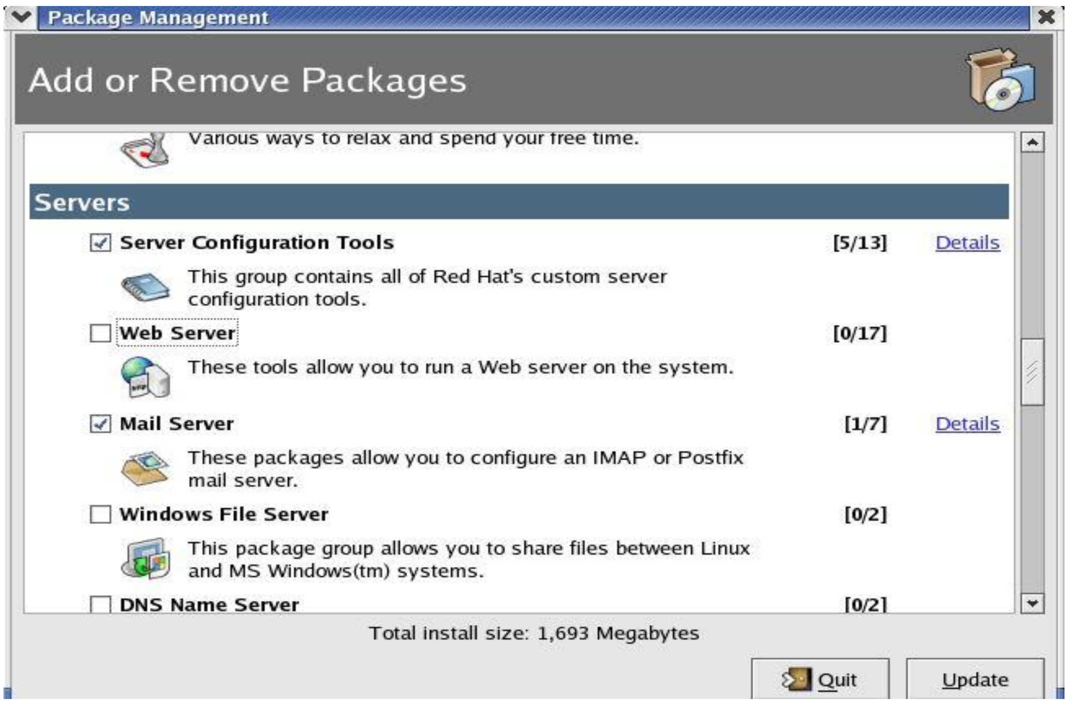
At the time of writing, 66% of all web sites are hosted on Apache web servers, most of them running on Linux or Unix operating systems. Apache's popularity is due not only of its open-source pedigree, but also to its highly competitive levels of performance, functionality, stability, flexibility, and security.

1) **Flexibility:** Apache's flexibility comes from the fact that it is a modular web server. That means that you can meet your requirements by plugging any number of external modules into the core httpd daemon. Of course, being open-source software, you also have access to Apache's source code, which you can customize to fit your needs.
2) **Scalable & Portable:** Apache is also very scalable. You can run Apache on high−end hardware, and it's possible to increase the capacity of Apache web servers by sharing the load across any number of servers. It's also very portable, being available for several operating systems.
3) **Security:** Apache's security is very good in comparison to other web servers. Moreover, the Apache Foundation is extremely active in the continued defense of Apache from security problems − particularly in the form of announcements and patches.
4) **Functionality:** Apache performs very well − it boasts a highly optimized daemon for serving static content which dramatically outperforms its nearest rivals. Moreover, it rarely crashes and achieves extremely long up−times.
5) **Documentation:** Apache comes with detailed documentation, which helps to make the setup and configuration easy.
6) **Support:** There's a wide network of support for Apache, in the form of mailing lists, newsgroups, and commercial vendors like Red Hat.
7) **Stability:** Apache development is active. The Apache Foundation is actively involved in development of new modules; new versions of Apache to make it reliable stable and secure.

11.2 Installing Apache

Apache is a modular server − the core server provides the basic functionality, with extended features available in various modules. This makes it very flexible and easy to configure, because you need to configure only the modules you need. So, it's worth looking at how to control the installation and removal of these modules. There are two ways to start RPM’s graphical interface:

One is to go to Main Menu| then System Settings | then Add/Remove Applications. Other is to open $ red hat-config-packages.



Package Management

Add or Remove Packages

Various ways to relax and spend your free time.

Servers

☑ Server Configuration Tools [5/13] Details
This group contains all of Red Hat's custom server configuration tools.

☐ Web Server [0/17]
These tools allow you to run a Web server on the system.

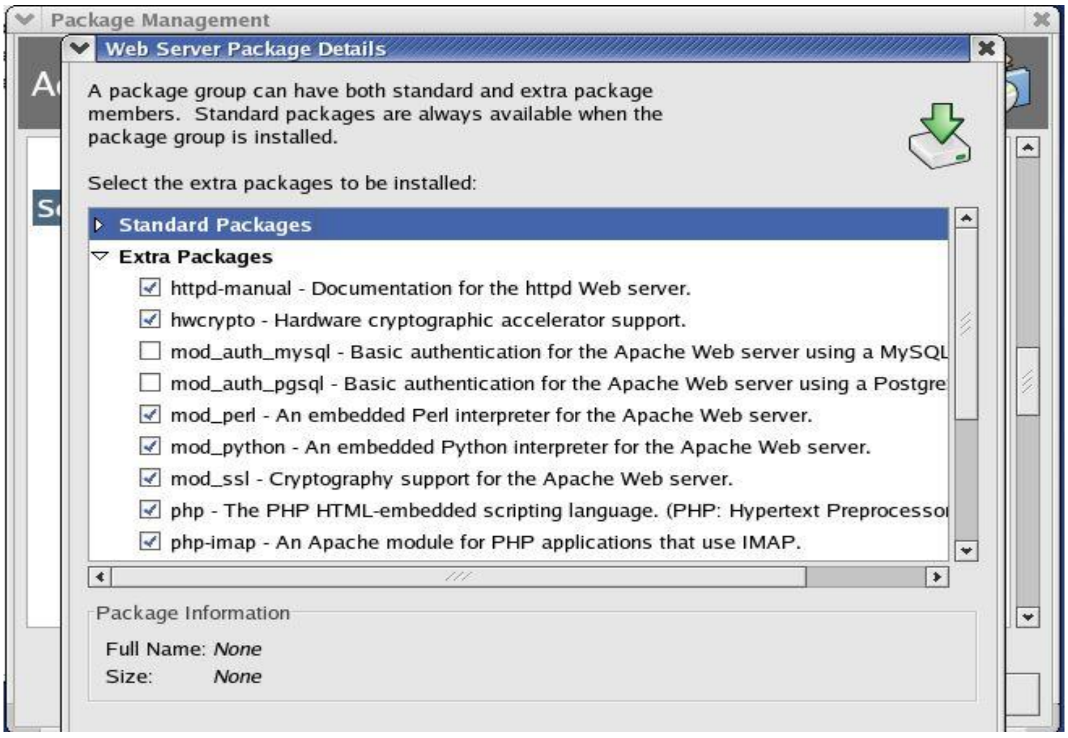
☑ Mail Server [1/7] Details
These packages allow you to configure an IMAP or Postfix mail server.

☐ Windows File Server [0/2]
This package group allows you to share files between Linux and MS Windows(tm) systems.

☐ DNS Name Server [0/2]

Total install size: 1,693 Megabytes

In this we need to search web server package.



Package Management

Web Server Package Details

A package group can have both standard and extra package members. Standard packages are always available when the package group is installed.

Select the extra packages to be installed:

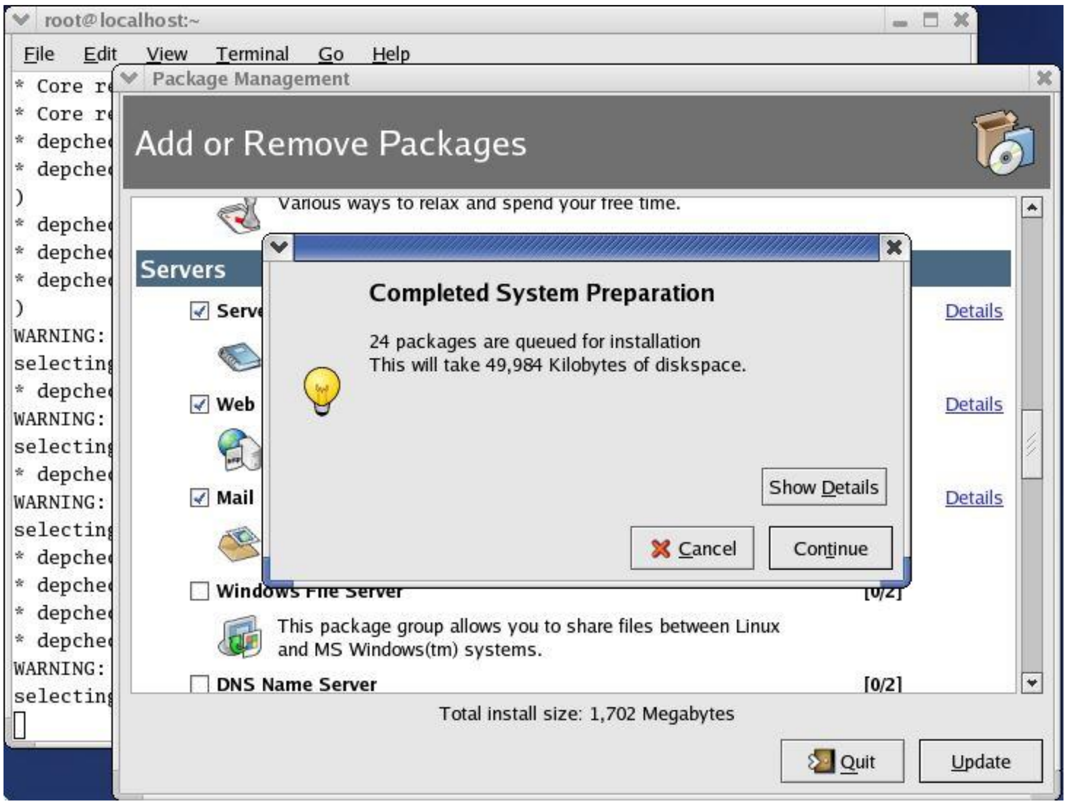
Standard Packages

Extra Packages

☑ httpd-manual - Documentation for the httpd Web server.
☑ hwcrypto - Hardware cryptographic accelerator support.
☐ mod_auth_mysql - Basic authentication for the Apache Web server using a MySQL
☐ mod_auth_pgsql - Basic authentication for the Apache Web server using a Postgre
☑ mod_perl - An embedded Perl interpreter for the Apache Web server.
☑ mod_python - An embedded Python interpreter for the Apache Web server.
☑ mod_ssl - Cryptography support for the Apache Web server.
☑ php - The PHP HTML-embedded scripting language. (PHP: Hypertext Preprocessor
☑ php-imap - An Apache module for PHP applications that use IMAP.

Package Information

Full Name: *None*
Size: *None*



```
* Core r
* Core r
* depche
* depche
)
* depche
* depche
* depche
)
WARNING:
selecting
* depche
WARNING:
selecting
* depche
WARNING:
selecting
* depche
* depche
* depche
* depche
WARNING:
selecting
```

Various ways to relax and spend your free time.

Completed System Preparation

24 packages are queued for installation
This will take 49,984 Kilobytes of diskspace.

Windows File Server [0/2]

This package group allows you to share files between Linux and MS Windows(tm) systems.

DNS Name Server [0/2]

Total install size: 1,702 Megabytes

Important packages

There are some important packages which needs to be installed for web server. These are:

Package	Description
httpd−manual	Contains the documentation for the Apache web server. After installation, you can access this documentation from the command line by typing man httpd.
hwcrypto	Provides support for hardware SSL acceleration cards. This package should be installed if you have hardware SSL acceleration cards like Ncipher Nforce on your server.
mod_ssl	Provides an SSL interface to the HTTPS web server, and hence enables the Apache web server to support SSL. This package should be installed if you want to provide secure connections to your clients.
Php	Provides the PHP module for Apache, which enables the web server to serve PHP web pages. This package is required if you if you want to host web sites which contain pages written with the PHP scripting language.
webalizer	Provides programs for web server log file analysis. This package enables you to generate HTML usage reports for your website.

Apache Configuration Files

There are various configuration files for configuring the Apache web server in the computer system.

1) The /etc/httpd/httpd.conf file is Apache's main configuration file.
2) The /etc/httpd/conf.d directory contains configuration files for any installed modules (such as PHP, SSL, and so on).
3) The /etc/httpd/logs directory is a symbolic link to /var/log/httpd directory, which contains all the Apache log files.
4) The /etc/httpd/modules directory is a symbolic link to /usr/lib/httpd/modules directory, which contains all the Apache modules configured as dynamic shared objects. Dynamic shared objects (or DSOs) are modules that are compiled separately from the Apache httpd binary. They are so−called because they can be loaded on demand.
5) The /etc/httpd/run directory is a symbolic link to /var/run, which contains the process ID file (httpd.pid) of the httpd process.
6) · /etc/rc.d/init.d/httpd is a shell script, used for starting and stopping the Apache web server.

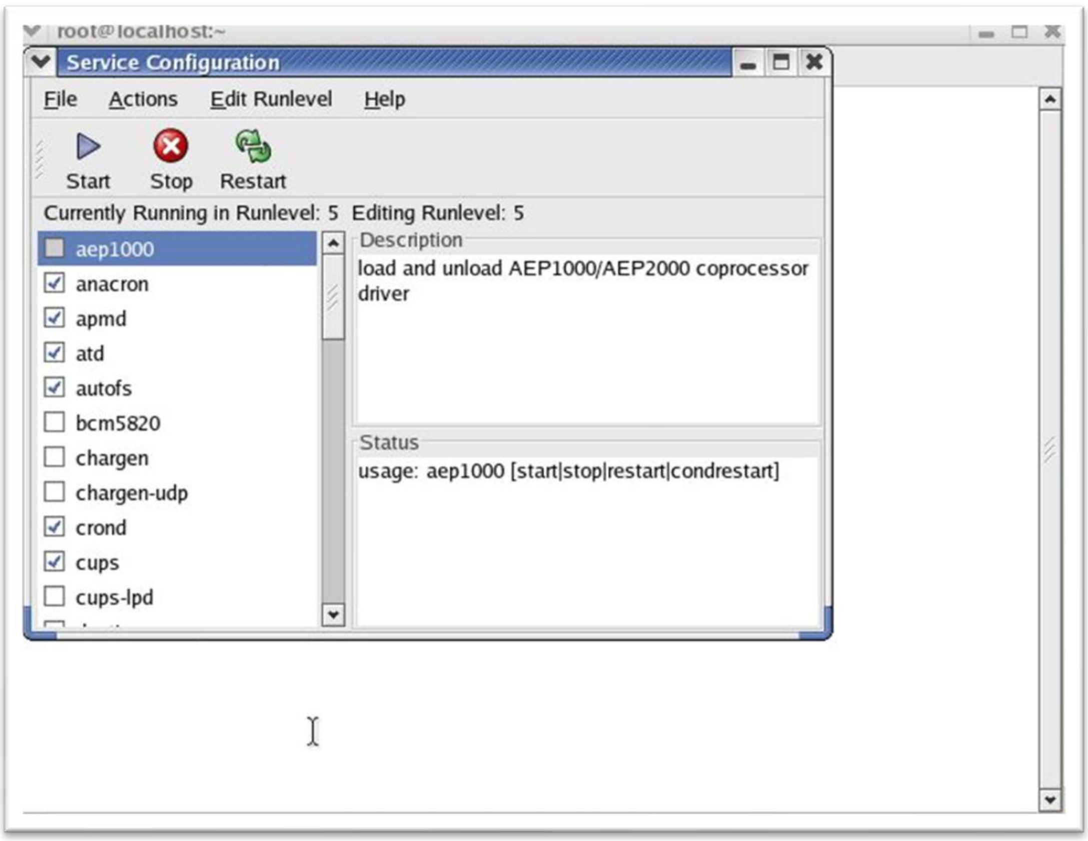
Starting Apache for the First Time

There are two ways by which Apache can be started. These are:

1) Main Menu | System Settings | Server Settings |Services
2) $ redhat−config−services

We can opt any of the ways. It would ask for root password if you started as a normal user.

Service Configuration Dialog: Here we need to start the service.



Currently Running in Runlevel: 5 Editing Runlevel: 5

- [ ] aep1000
- [x] anacron
- [x] apmd
- [x] atd
- [x] autofs
- [ ] bcm5820
- [ ] chargen
- [ ] chargen-udp
- [x] crond
- [x] cups
- [ ] cups-lpd

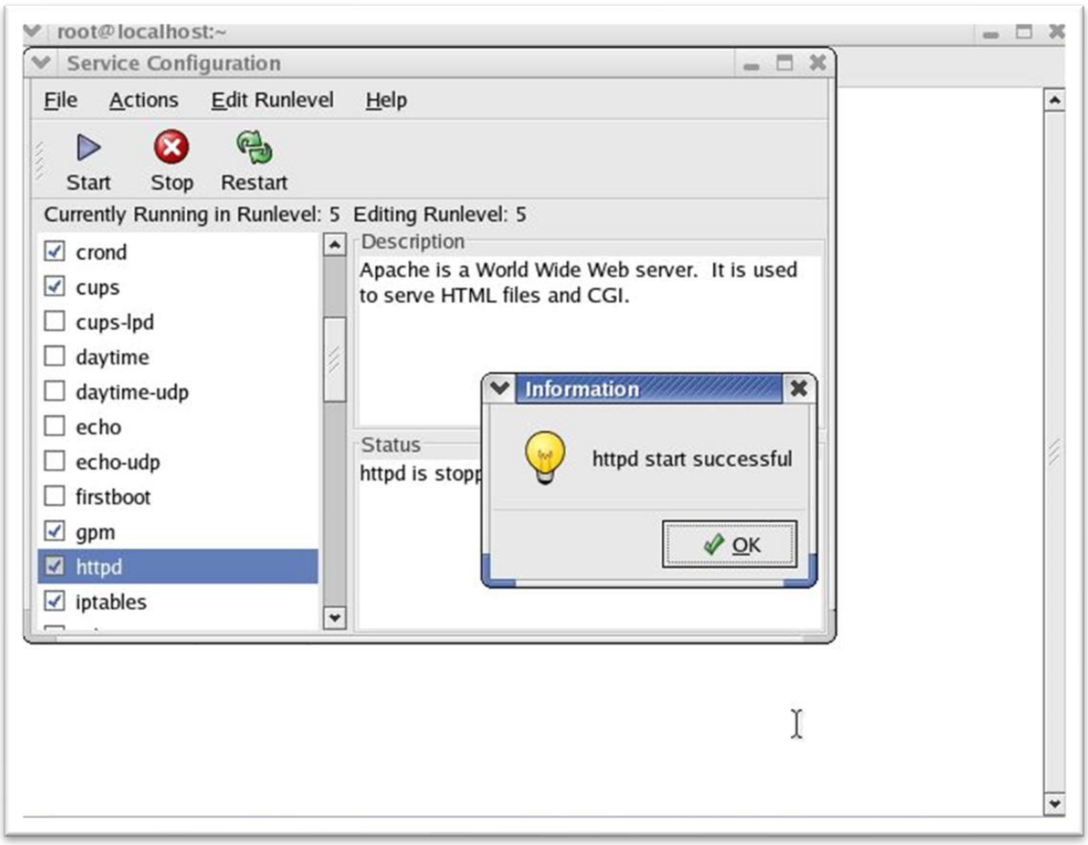
Description

load and unload AEP1000/AEP2000 coprocessor driver

Status

usage: aep1000 [start|stop|restart|condrestart]

Httpd need to be checked here.



Currently Running in Runlevel: 5 Editing Runlevel: 5

Description

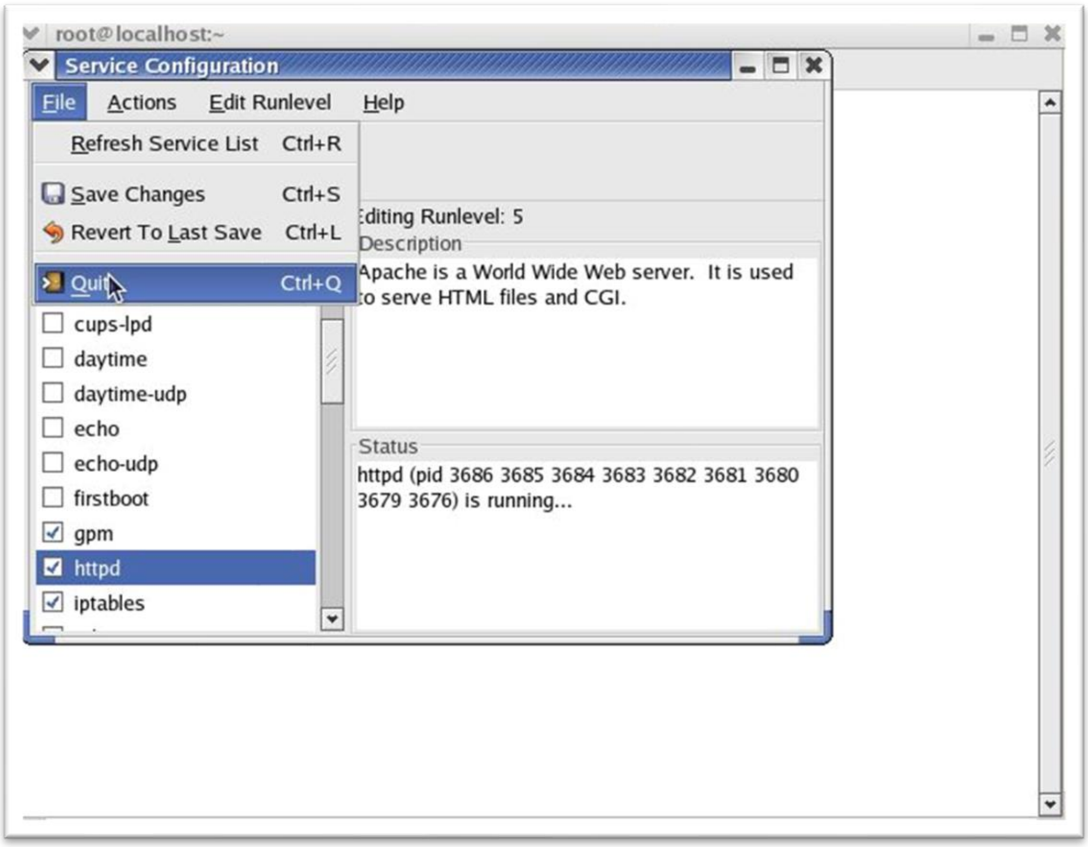
Apache is a World Wide Web server. It is used to serve HTML files and CGI.

Status

httpd is stopp

httpd start successful

When this is done. We need to save the changes and quit.



Editing Runlevel: 5

Description

Apache is a World Wide Web server. It is used to serve HTML files and CGI.

Status

httpd (pid 3686 3685 3684 3683 3682 3681 3680 3679 3676) is running...

To control the Apache web server from the command line, we can use the service command to fire the httpdscript. Here's how we use it to start the web server:

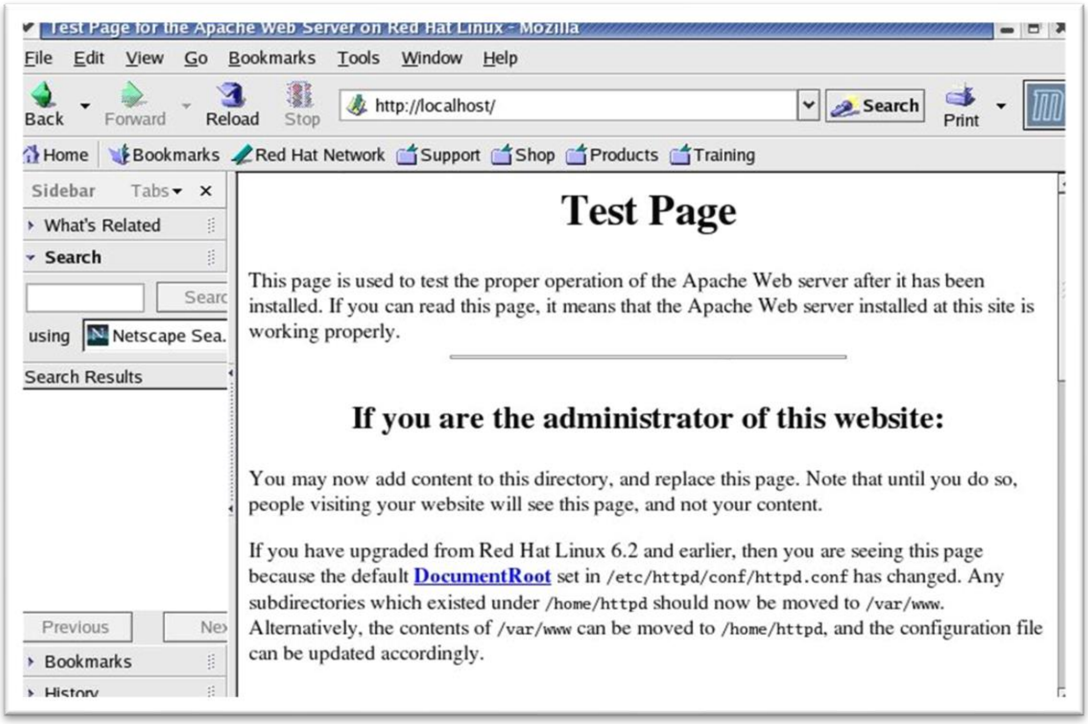
```
# service httpd start
Starting httpd:                                            [ OK ]
```

If there are difficulties in starting the web server, then you'll find out about it here. For example, if youattempt to do this without root privileges, then you'll get a message telling you that permission is denied. Andhere's another example:

```
# service httpd start
Starting httpd: httpd: Could not determine the server's fully qualified
domain name, using 192.168.0.99 for ServerName        [ OK ]
```

Testing the Apache Web Server

Once you've started the Apache web server, you should test it to see if it's working properly. To do that, we'll use a web browser to request a web page from our server!. There's a page provided by default for this purpose, and you can request it via the URL http://localhost. So, launch a web browser (Main Menu | Internet | Mozilla Web Browser), and type this URL into the address box:



Test Page

This page is used to test the proper operation of the Apache Web server after it has been installed. If you can read this page, it means that the Apache Web server installed at this site is working properly.

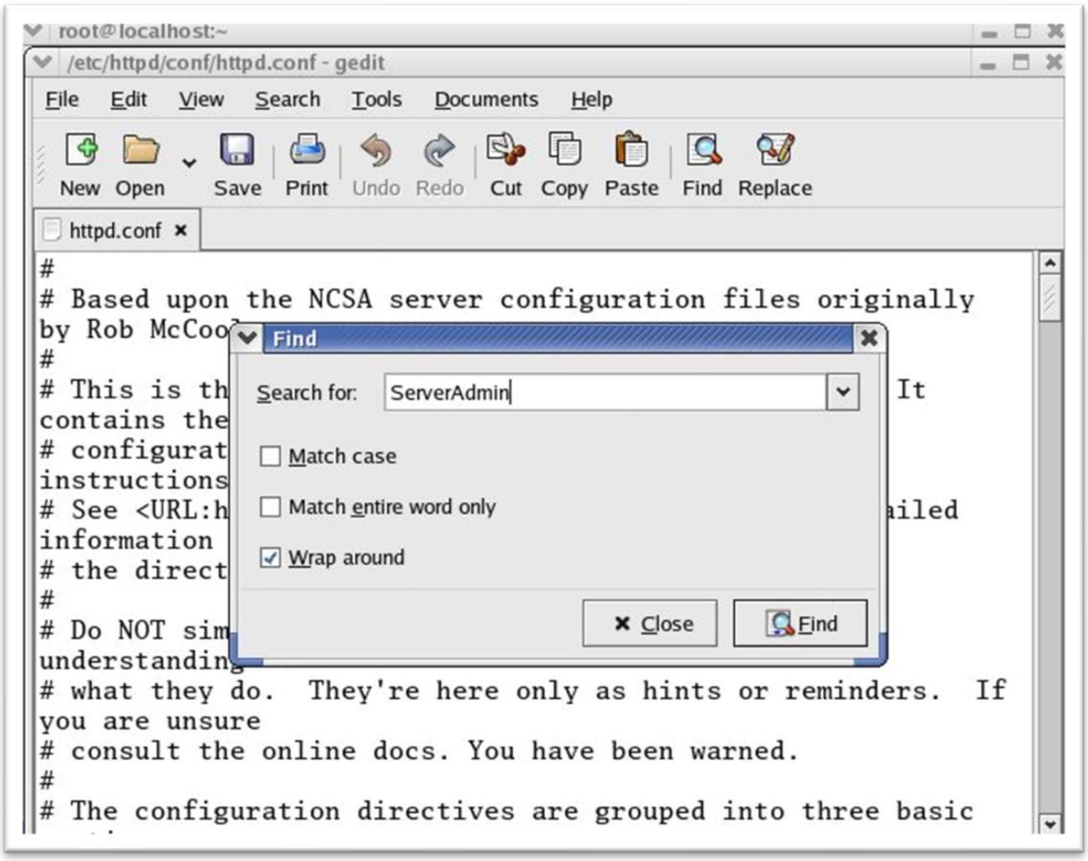
If you are the administrator of this website:

You may now add content to this directory, and replace this page. Note that until you do so, people visiting your website will see this page, and not your content.

If you have upgraded from Red Hat Linux 6.2 and earlier, then you are seeing this page because the default DocumentRoot set in /etc/httpd/conf/httpd.conf has changed. Any subdirectories which existed under /home/httpd should now be moved to /var/www. Alternatively, the contents of /var/www can be moved to /home/httpd, and the configuration file can be updated accordingly.

Configuring your Web Server

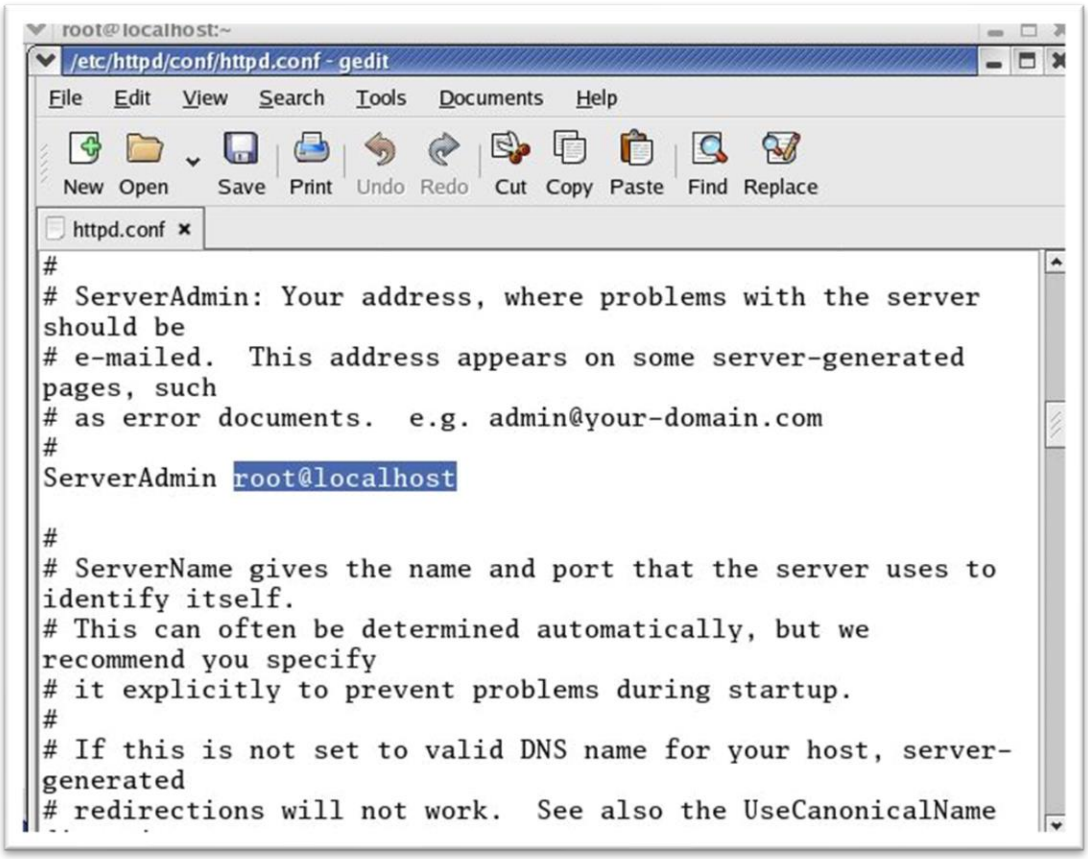
Launch the gedit text editor (by selecting Main Menu | Accessories | Text Editor).Use it to open the file /etc/httpd/conf/httpd.conf.Select Search | Find and use the resulting dialog to find the word Server Admin in the file. The first occurrence should be the Server Admin directive. Write your own email id here in the place of root@localhost.


root@localhost:~
/etc/httpd/conf/httpd.conf - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
httpd.conf
Find
Close
Find

Search for: ServerAdmin

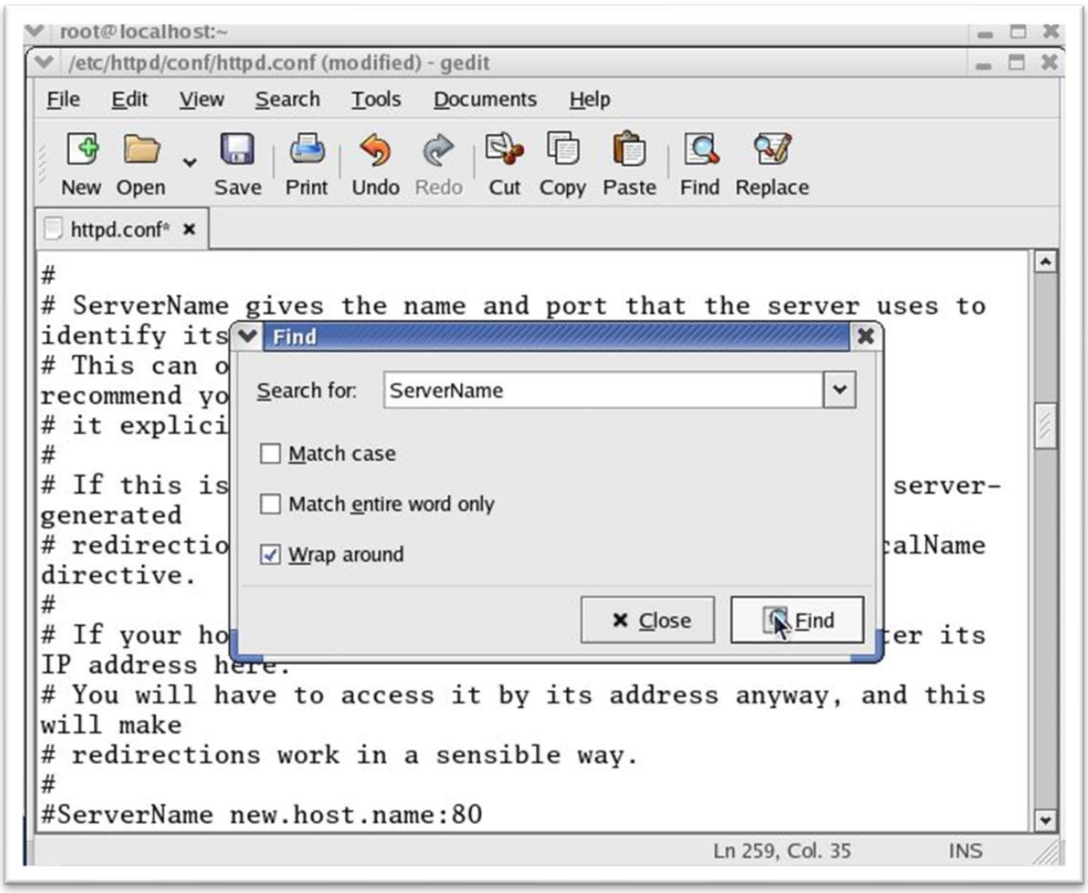
- [ ] Match case
- [ ] Match entire word only
- [x] Wrap around

```
#
# Based upon the NCSA server configuration files originally
by Rob McCoo
#
# This is th                                              It
contains the
# configurat
instructions
# See <URL:h                                              iled
information
# the direct
#
# Do NOT sim
understandin
# what they do.  They're here only as hints or reminders.  If
you are unsure
# consult the online docs. You have been warned.
#
# The configuration directives are grouped into three basic
```


root@localhost:~
/etc/httpd/conf/httpd.conf - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
httpd.conf

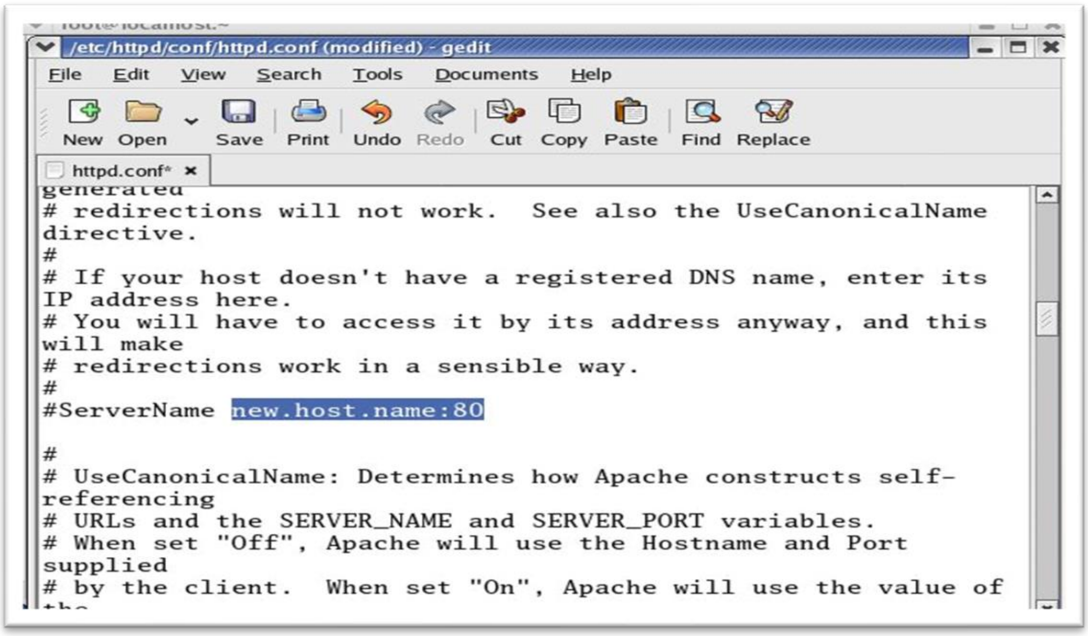
```
#
# ServerAdmin: Your address, where problems with the server
should be
# e-mailed.  This address appears on some server-generated
pages, such
# as error documents.  e.g. admin@your-domain.com
#
ServerAdmin root@localhost

#
# ServerName gives the name and port that the server uses to
identify itself.
# This can often be determined automatically, but we
recommend you specify
# it explicitly to prevent problems during startup.
#
# If this is not set to valid DNS name for your host, server-
generated
# redirections will not work.  See also the UseCanonicalName
```

Next thing we are going to search is ServerName. So in front of ServerName, new.host.name:80 is written. Here, we need to write the IP address of our machine.


root@localhost:~
/etc/httpd/conf/httpd.conf (modified) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
httpd.conf*
Find
Close
Find
Ln 259, Col. 35
INS

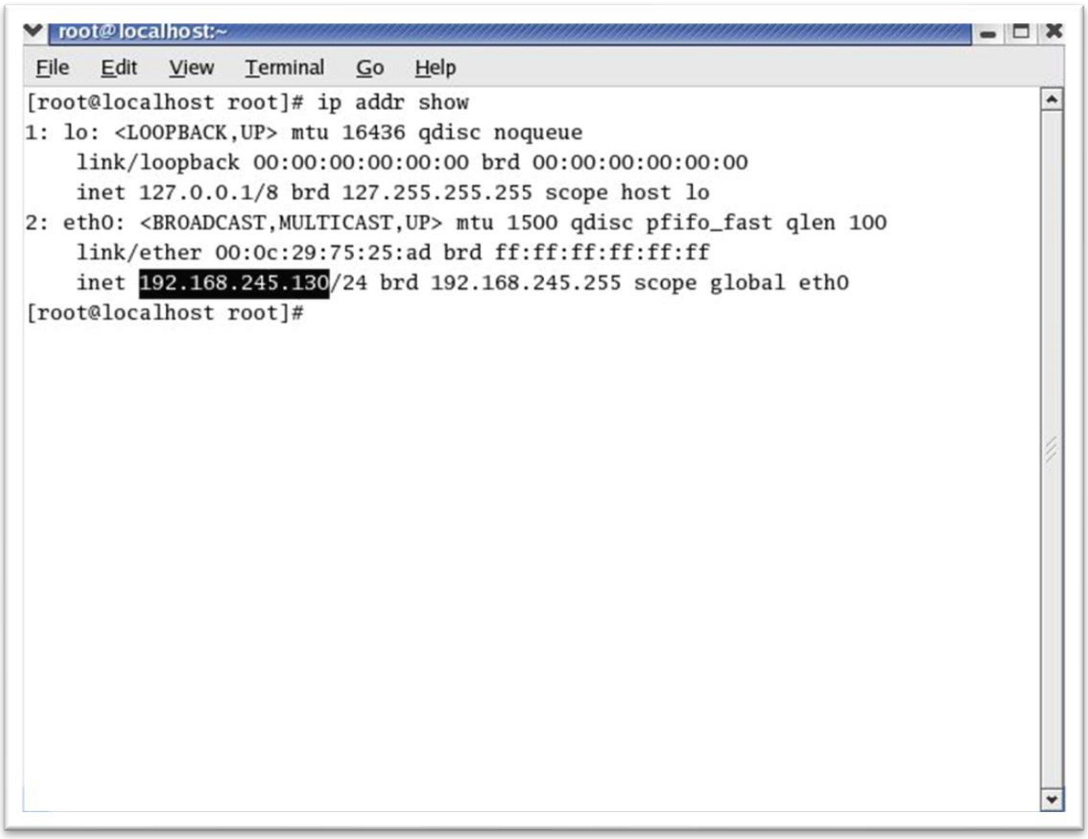
Search for: ServerName
Match case [ ]
Match entire word only [ ]
Wrap around [✓]

```
#
# ServerName gives the name and port that the server uses to
identify its
# This can o
recommend yo
# it explici
#
# If this is                                          server-
generated
# redirectio                                          alName
directive.
#
# If your ho                                          er its
IP address here.
# You will have to access it by its address anyway, and this
will make
# redirections work in a sensible way.
#
#ServerName new.host.name:80
```


root@localhost:~
/etc/httpd/conf/httpd.conf (modified) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
httpd.conf*

```
generated
# redirections will not work.  See also the UseCanonicalName
directive.
#
# If your host doesn't have a registered DNS name, enter its
IP address here.
# You will have to access it by its address anyway, and this
will make
# redirections work in a sensible way.
#
#ServerName new.host.name:80

#
# UseCanonicalName: Determines how Apache constructs self-
referencing
# URLs and the SERVER_NAME and SERVER_PORT variables.
# When set "Off", Apache will use the Hostname and Port
supplied
# by the client.  When set "On", Apache will use the value of
```

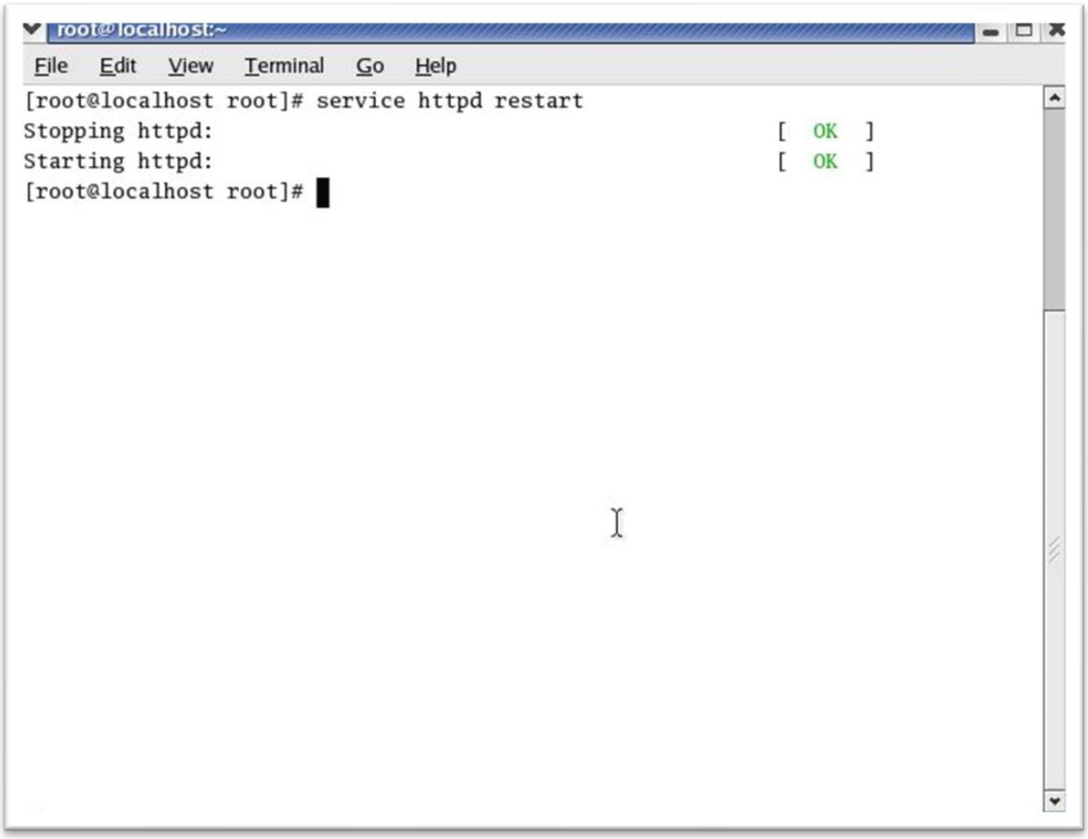
By giving the command ip addr show, it will show us the ip address of the machine. So, in front of ServerName write the ip address of the machine.



```
[root@localhost root]# ip addr show
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:0c:29:75:25:ad brd ff:ff:ff:ff:ff:ff
    inet 192.168.245.130/24 brd 192.168.245.255 scope global eth0
[root@localhost root]#
```

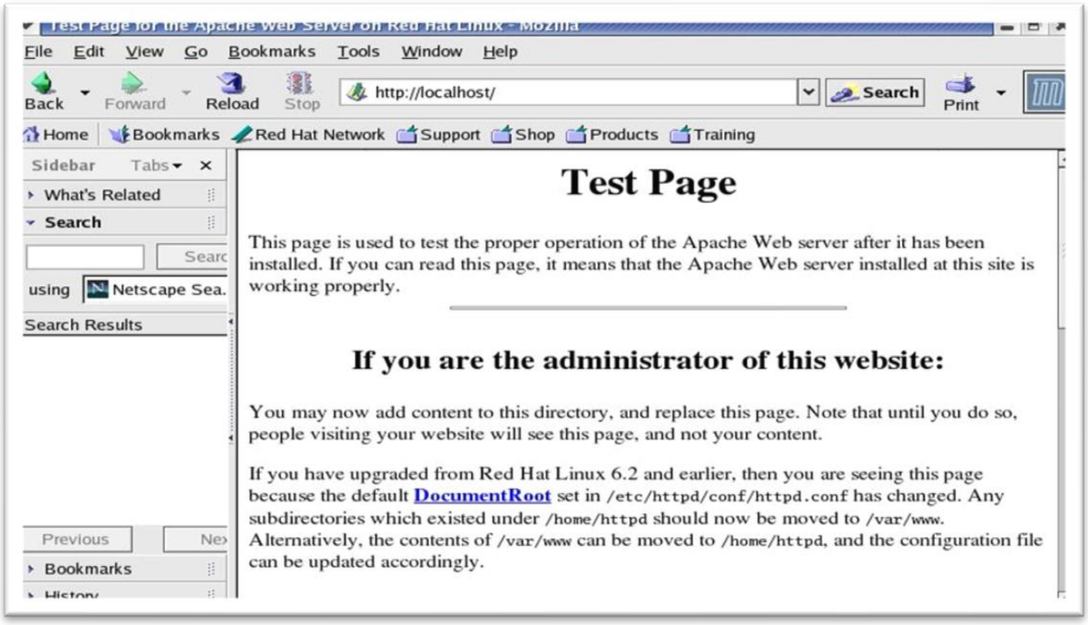
Configuring your Web Server

Again we need to restart the httpd service by writing service httpd restart.



```
[root@localhost root]# service httpd restart
Stopping httpd:                                           [  OK  ]
Starting httpd:                                           [  OK  ]
[root@localhost root]#
```

Once this is done. Again open httpd://localhost


Test Page for the Apache Web Server on Red Hat Linux - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost/
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
Sidebar Tabs
What's Related
Search
using Netscape Sea.
Search Results
Previous
Bookmarks
History

Test Page

This page is used to test the proper operation of the Apache Web server after it has been installed. If you can read this page, it means that the Apache Web server installed at this site is working properly.

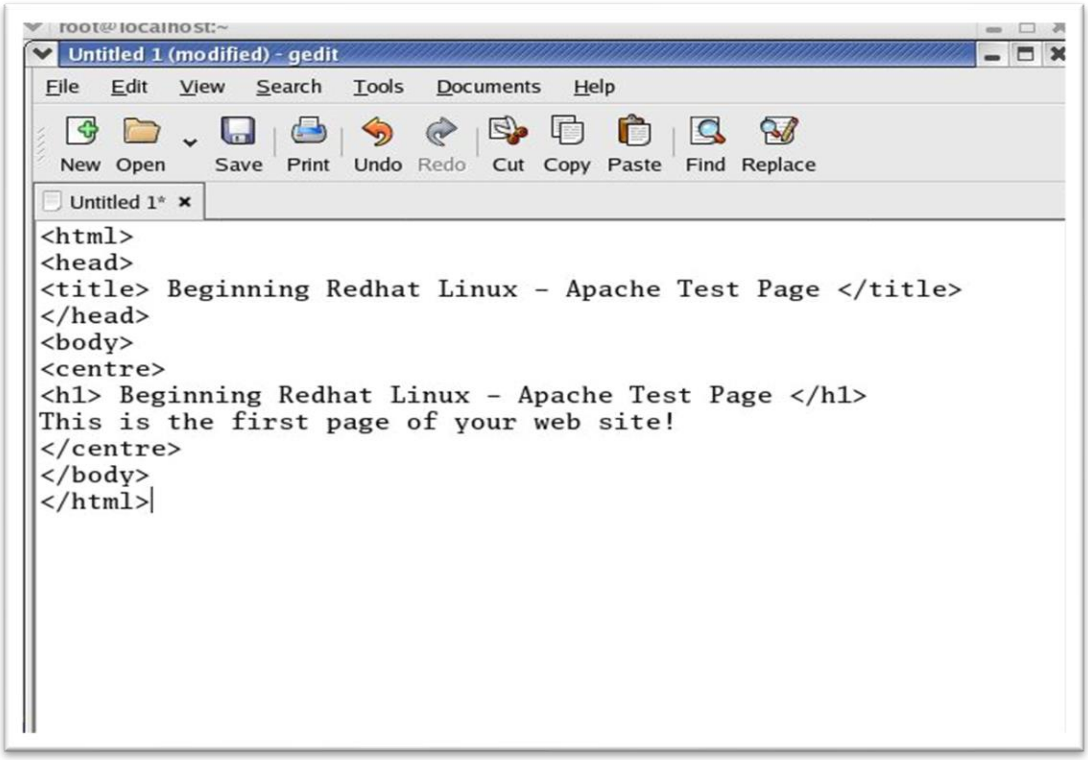
If you are the administrator of this website:

You may now add content to this directory, and replace this page. Note that until you do so, people visiting your website will see this page, and not your content.

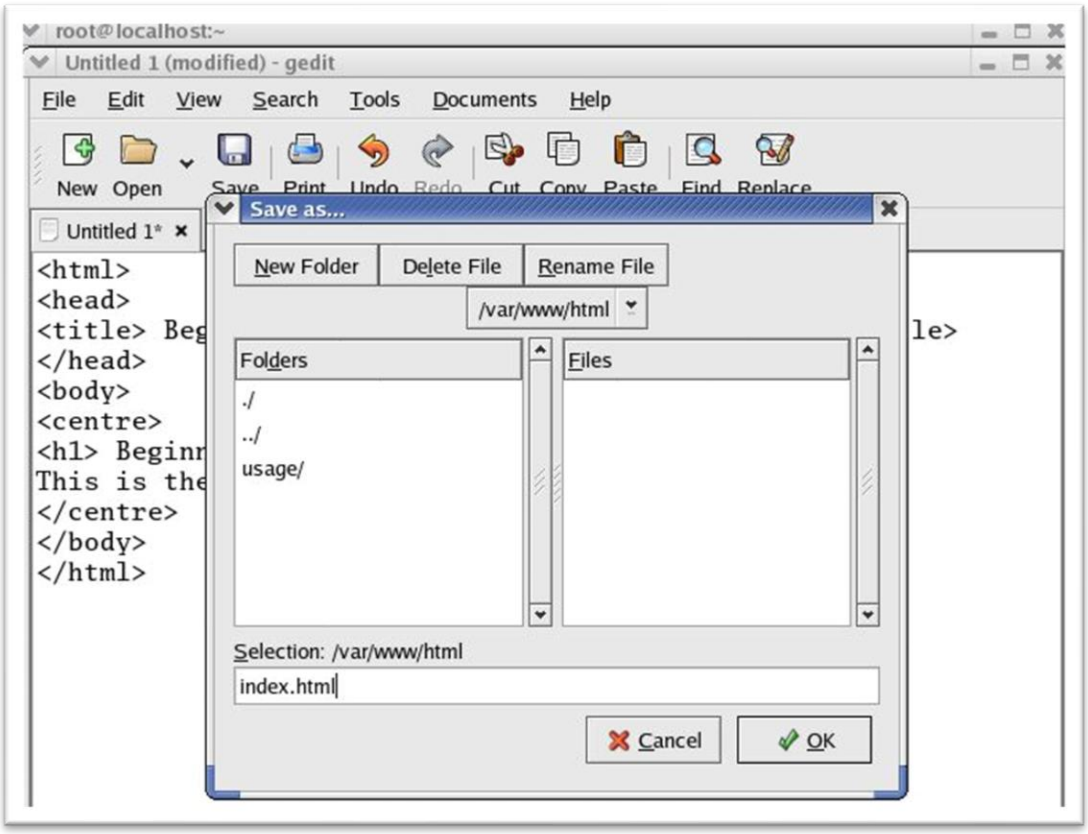
If you have upgraded from Red Hat Linux 6.2 and earlier, then you are seeing this page because the default DocumentRoot set in /etc/httpd/conf/httpd.conf has changed. Any subdirectories which existed under /home/httpd should now be moved to /var/www. Alternatively, the contents of /var/www can be moved to /home/httpd, and the configuration file can be updated accordingly.

Setting up Your First Web Page

- This involves creating a simple HTML web page, and saving it to a location on the hard disk that is used by the web server to store published web pages.
- Then, when a user requests the page, the web server will be able to respond by retrieving it from this location and sending it to the requestor.
- Launch an editor (gedit).


root@localhost:~
Untitled 1 (modified) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
Untitled 1*

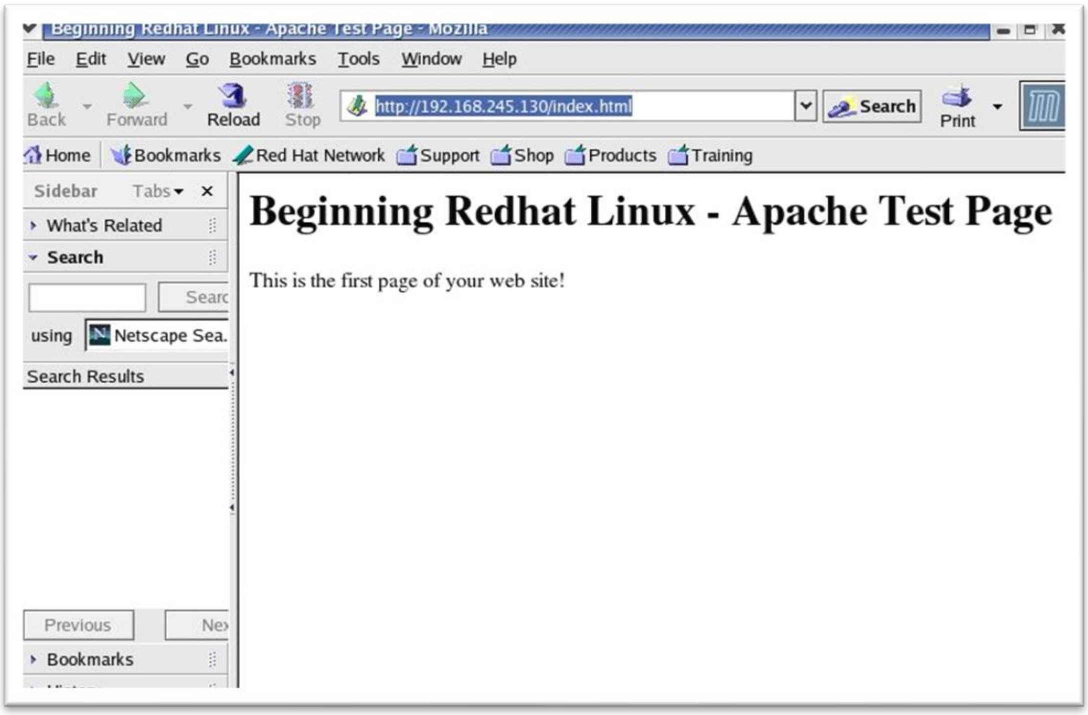
```
<html>
<head>
<title> Beginning Redhat Linux - Apache Test Page </title>
</head>
<body>
<centre>
<h1> Beginning Redhat Linux - Apache Test Page </h1>
This is the first page of your web site!
</centre>
</body>
</html>
```


root@localhost:~
Untitled 1 (modified) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
Untitled 1*
Save as...
New Folder Delete File Rename File
/var/www/html
Folders
./
../
usage/
Files
Cancel
OK

```
<html>
<head>
<title> Beg                                    le>
</head>
<body>
<centre>
<h1> Beginr
This is the
</centre>
</body>
</html>
```

Selection: /var/www/html

index.html


Beginning Redhat Linux - Apache Test Page - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://192.168.245.130/index.html
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
Sidebar Tabs
What's Related
Search
Searc
using Netscape Sea.
Search Results
Previous Ne
Bookmarks

Beginning Redhat Linux - Apache Test Page

This is the first page of your web site!

Summary

- A web server is required When we want to publish web pages on the Internet or on an intranet.

- The standard protocol for internet is Hypertext Transfer Protocol (HTTP).
- All web servers listen for HTTP requests on port 80.
- Apache's popularity is due not only of its open source pedigree, but also to its highly competitive levels of performance, functionality, stability, flexibility, and security.
- There are two ways to start RPM's graphical interface: One is: Main Menu | System Settings | Add/Remove Applications and another is $ redhat-config-packages.
- The important package httpd_manual contains the documentation for the Apache web server. After installation, you can access this documentation from the command line by typing man httpd.
- The /etc/httpd/httpd.conf file is Apache's main configuration file.
- A web server is an application that does two things:It listens for page requests and when it receives a page request, it examines the request and responds with the page that was requested.

Keywords

- **Web server:** web server performs is to display the content of the website, which it does by storing, then processing, and eventually delivering the webpages to the user who has requested it.
- **Static Web Server:**The static content is the one that is fixed. The static web server contains HTTP software and computer. This is static because the server sends hosted files as is present to the browser.
- **Dynamic Web Server:** The dynamic web browser will have the webserver and the software like the database and the applications server. This is dynamic because the application server is used to update the files hosted before these are sent to the browser.
- **Apache Web Server:** Apache is a modular server − the core server provides the basic functionality, with extended features available in various modules. This makes it very flexible and easy to configure, because you need to configure only the modules you need.
- **HTTP**: Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes.

Self Assessment

1. When we are talking about the web servers, how many packages are available?

A. 15
B. 16
C. 17
D. 18

2. Which package contains the documentation of httpd web server?

A. httpd-manual
B. hwcrypto
C. php
D. php-image

3. We can install the web server package if we have logged in as
A. Root
B. Normal user
C. Either root or normal user
D. None of the above

4. When we are installing web server, which service should be started from service configuration dialog?
A. echo
B. httpd
C. cups
D. None of the above

5. The configuration file is modified by searching
A. ServerAdmin
B. ServerName
C. Both of the above
D. None of the above

6. Where do we provide the ip address of machine when the modification of configuration file is done?
A. ServerAdmin
B. ServerName
C. Both of the above
D. None of the above

7. The important packages for Apache web server are
A. httpd-manual
B. mod_ssl
C. php
D. All of the above mentioned

8. Which command is used to check the ip address of the Linux machine?
A. ip addr show
B. internet address show
C. ip address show
D. None of the above mentioned

9. Why do we require web server on a Linux machine?
A. To test the site that is under development
B. To test a private website only available in private network
C. Both of the above
D. None of the above

10. With which way we can open RPM's graphical interface?
A. Main Menu | System Settings | Add/Remove Applications
B. $ redhat-config-packages
C. Either of these mentioned
D. None of the above

11. HTTP stands for
A. Host text transfer protocol
B. Hypertext transfer protocol
C. High text transfer protocol
D. None of the above

12. The commercial web servers are
A. Zeus
B. Microsoft
C. SunOne
D. All of the mentioned

13. Which of these features belongs to Apache web server?
A. Portability
B. Scalability
C. Security
D. All of the above mentioned

14. What is the task of a web server?
A. Listens for a page request.
B. Examines and responds with the page requested
C. Both of the above
D. None of the above

15. Which one is the standard protocol of internet?
A. HTTP
B. PTTH
C. PTHT
D. None of the above

Answers for Self Assessment

1.	C	2.	A	3.	A	4.	B	5.	C
6.	B	7.	D	8.	A	9.	C	10.	C
11.	B	12.	D	13.	D	14.	C	15.	A

Review Questions:

1. What is a web server? What is the need of installing web server on RedHat Linux machine? Give some examples of commercial and open source web servers.
2. What is Apache web server? Write some features of it in detail.
3. Write in detail how can we install Apache in RedHat Linux machine.
4. What are the important packages and configuration files of Apache web service. Give details.
5. How can we start and test Apache web server? Write in detail about its configuration.

Further Readings

Sandip Bhattacharya,Pancrazio De Mauro,Shishir Gundavaram,Mark Mamone,Kalip Sharma,Deepak Thomas, and Simon Whiting, Beginning Red Hat Linux 9, Wiley Publishing, Inc.

Web Links

https://www.sumologic.com/blog/apache-web-server-introduction/

Unit 12: File Server Configuration

CONTENTS

Objectives

After studying this unit, you will be able to

- Understand the FTP protocol
- Understand the relevance and use of FTP
- How to start FTP server
- How to test FTP server
- How to use FTP client to test anonymous read access

Introduction

FTP stands for File Transfer Protocol. At its core, the file transfer protocol is a way to connect two computers to one another in the safest possible way to help transfer files between two or more points. To put it simply, it's the means by which files are securely shared between parties. FTP servers are the solutions used to facilitate file transfers across the internet. If you send files using FTP, files are either uploaded or downloaded to the FTP server. When you're uploading files, the files are transferred from a personal computer to the server. When you're downloaded files, the files are transferred from the server to your personal computer. TCP/IP (Transmission Control Protocol/Internet Protocol), or the language the internet uses to execute commands, is used to transfer files via FTP. So, the main points in consideration are:

- If you want to enable other users to download files from a location on your server's hard disk, and/or to upload files to that location, then one solution is to install an FTP server.
- You can think of an FTP server essentially as an area of disk space that is used for storing files, plus the software and configuration required to allow other users to upload and download files.

- When users want to upload or download from your FTP server, they use a program called an FTP client.

These communications between FTP server and FTP client take place using the File Transfer Protocol (FTP). FTP is a TCP protocol that is designed specifically for the transfer of files over a network, and it's one of the oldest Internet protocols still in widespread use.

Relevance of FTP

The availability of so many different FTP client programs, and the fact that many operating systems come with FTP software pre−installed, are indications of how relevant FTP still is today. FTP also is more efficient at large file transfers than HTTP and requires a password for access. FTP keeps a log of data transmission, where HTTP does not. All these components combined create a system that the common user might not notice, but which is incredibly important for professionals working with IT.FTP and its derivatives are the best choices when dealing with the manipulation of larger quantities of data. Rather than simply giving one-way access, FTP services allow users an enormous amount of control, and this can be extremely beneficial to individuals and business which update regularly.

Security of FTP

FTP is generally considered to be an insecure protocol because it relies on clear-text usernames and passwords for authentication and does not use encryption. Data sent via FTP is vulnerable to sniffing, spoofing, and brute force attacks, among other basic attack methods. It is not considered a secure protocol, because communication between the FTP client and server are unencrypted. Consequently, Secure FTP (SFTP) is also becoming popular (and, indeed, is part of the openssh package that comes with Red Hat Linux 9).It's also possible to configure your FTP server in other ways, for example by forcing users to log in, or by using access control lists (ACLs) to allow different rights to different groups of users.

Did you know?

The FTP was not built to be secure. The communication between the FTP client and server are unencrypted.

Anonymous FTP Access

- Anonymous FTP is called *anonymous* because you don’t need to identify yourself before accessing files. In fact, many FTP servers still allow anonymous FTP access, which means that the FTP server allows any user to access its disk space and download its files.
- Anonymous FTP access is used mostly to enable users to access freely available documents and files via the Internet without access control.

Despite the security issues, FTP remains popular − it's fast and easy to use, and it is the Internet standard protocol for file transfer.

FTP Servers in the Red Hat Linux Distribution

There are few FTP servers available in Red Hat Linux distribution. These are:

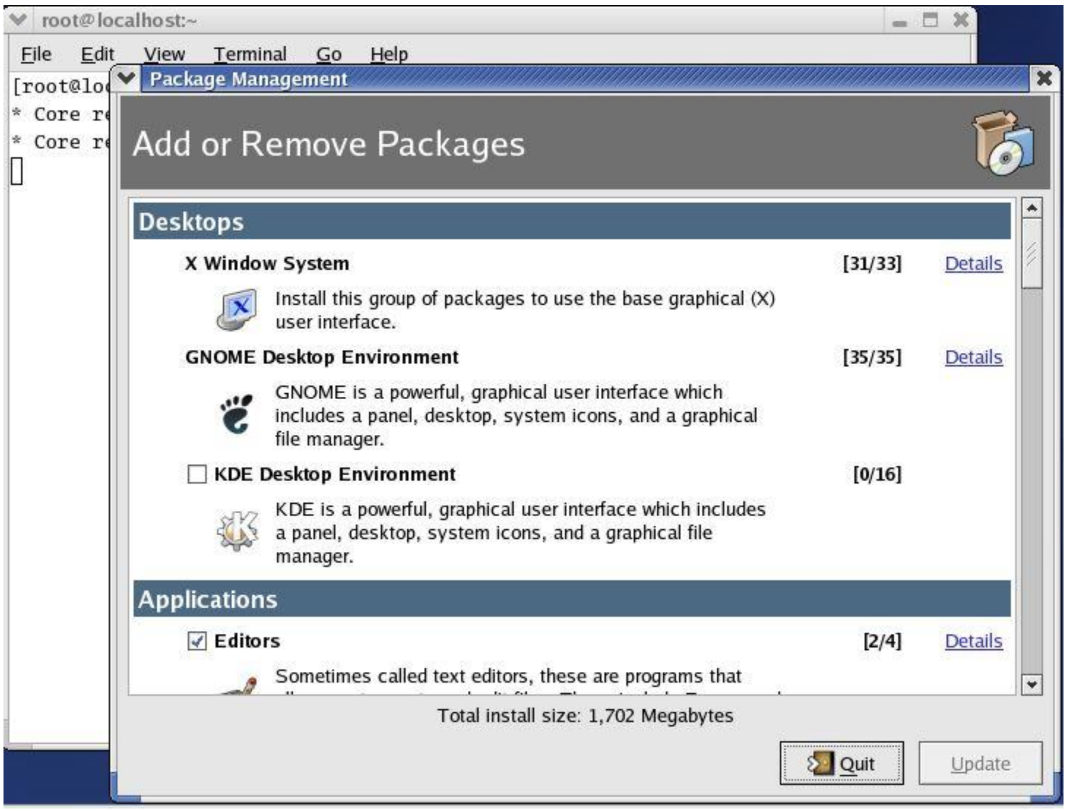
FTP Server	Remarks
vsftpd	It is a simplified FTP server implementation. It is designed to be a very secure FTP server and can also be configured to allow anonymous access.
TUX	It is a kernel−based, threaded, extremely high-performance HTTP server, which also has FTP capabilities. TUX is perhaps the best in terms of performance but offers less functionality than other FTP server software. TUX is installed by default with Red Hat Linux 9.

wu-ftpd	It is a highly configurable and full−featured FTP daemon, which was popular in earlier versions of Red Hat Linux but has since given way to the more security−conscious vsftpd.
gssftpd	It is a kerberized FTP daemon, which means that it is suitable for use with the Kerberos authentication system.

12.1 Installing the vsftpd FTP Server

The easiest way to install the vsftpd FTP Server package is via the RPM GUI tool. There are two ways by which we can open the RPM GUI tool. These are:

- Main Menu | System Settings | Add/Remove Applications
- $ redhat-config-packages



```
[root@lo
* Core re
* Core re
```

Add or Remove Packages

Desktops

X Window System [31/33] Details
Install this group of packages to use the base graphical (X) user interface.

GNOME Desktop Environment [35/35] Details
GNOME is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

☐ KDE Desktop Environment [0/16]
KDE is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

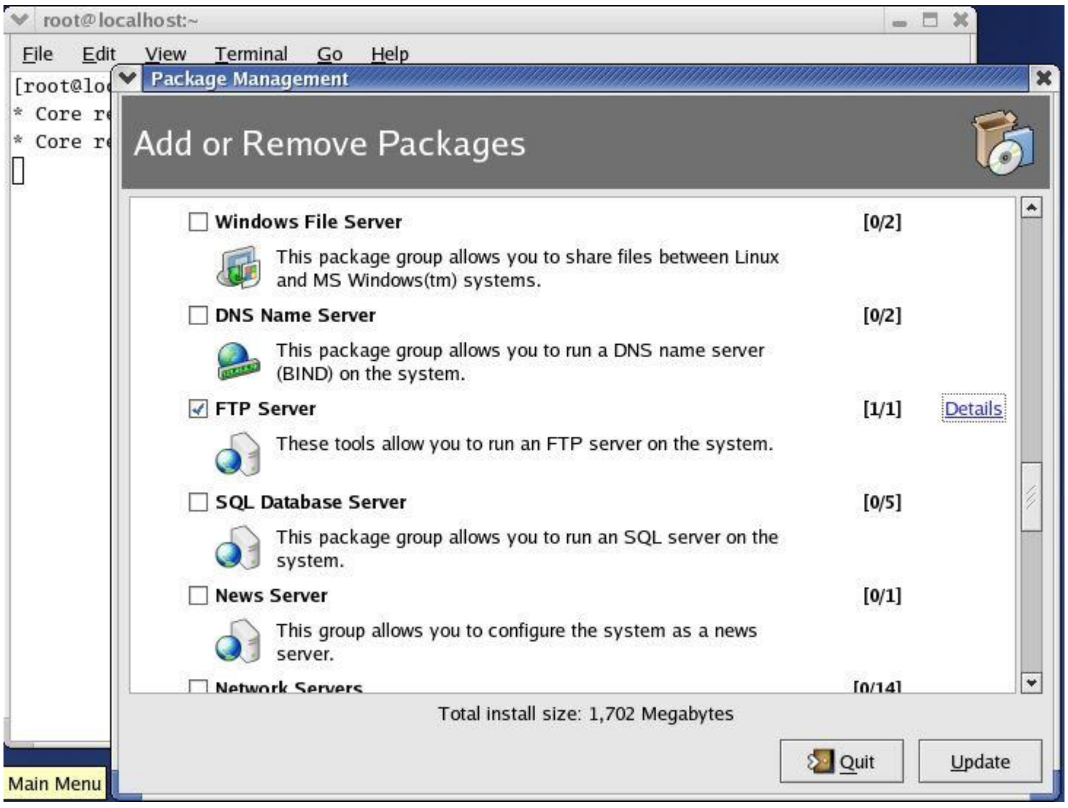
Applications

☑ Editors [2/4] Details
Sometimes called text editors, these are programs that

Total install size: 1,702 Megabytes

12.2 Installing the vsftpd FTP Server

For installation of FTP server, we will see the category ‘Servers’. In this we will click on FTP server. By clicking on details link, we can see the packages available in that. As in FTP server we have just one package. After clicking on update button, we can



```
[root@lo
* Core r
* Core r
```

Package Management

Add or Remove Packages

☐ Windows File Server [0/2]
This package group allows you to share files between Linux and MS Windows(tm) systems.

☐ DNS Name Server [0/2]
This package group allows you to run a DNS name server (BIND) on the system.

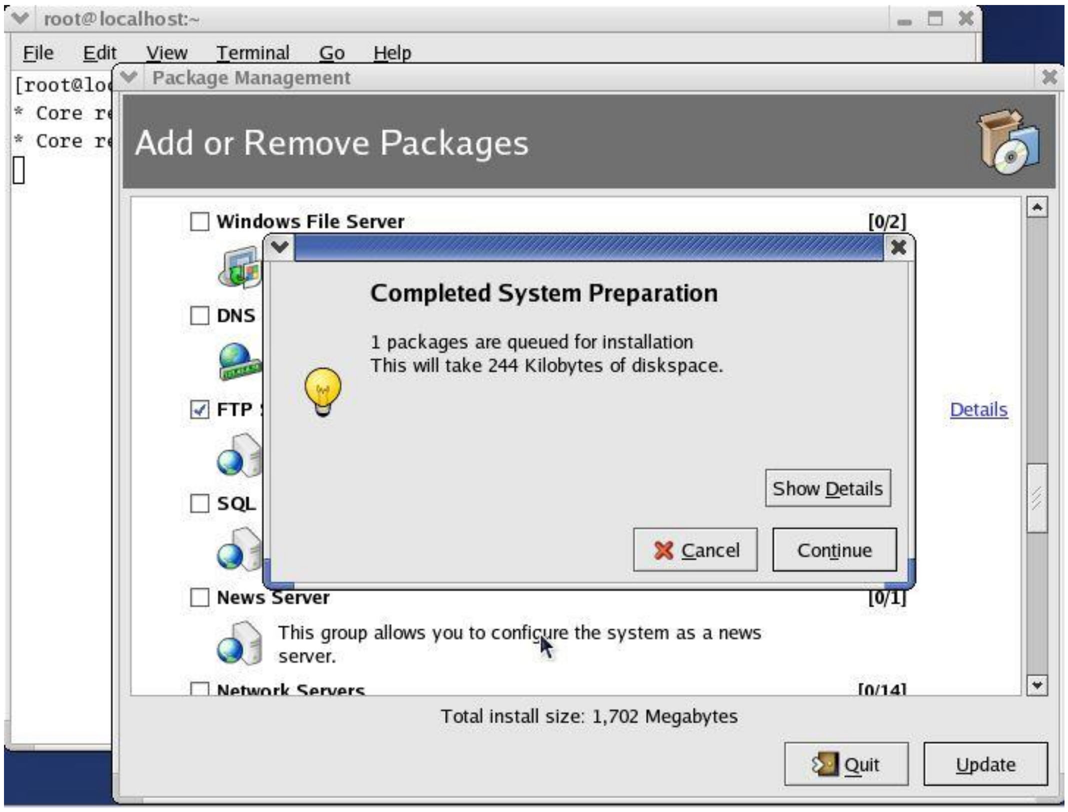
☑ FTP Server [1/1] Details
These tools allow you to run an FTP server on the system.

☐ SQL Database Server [0/5]
This package group allows you to run an SQL server on the system.

☐ News Server [0/1]
This group allows you to configure the system as a news server.

☐ Network Servers [0/14]

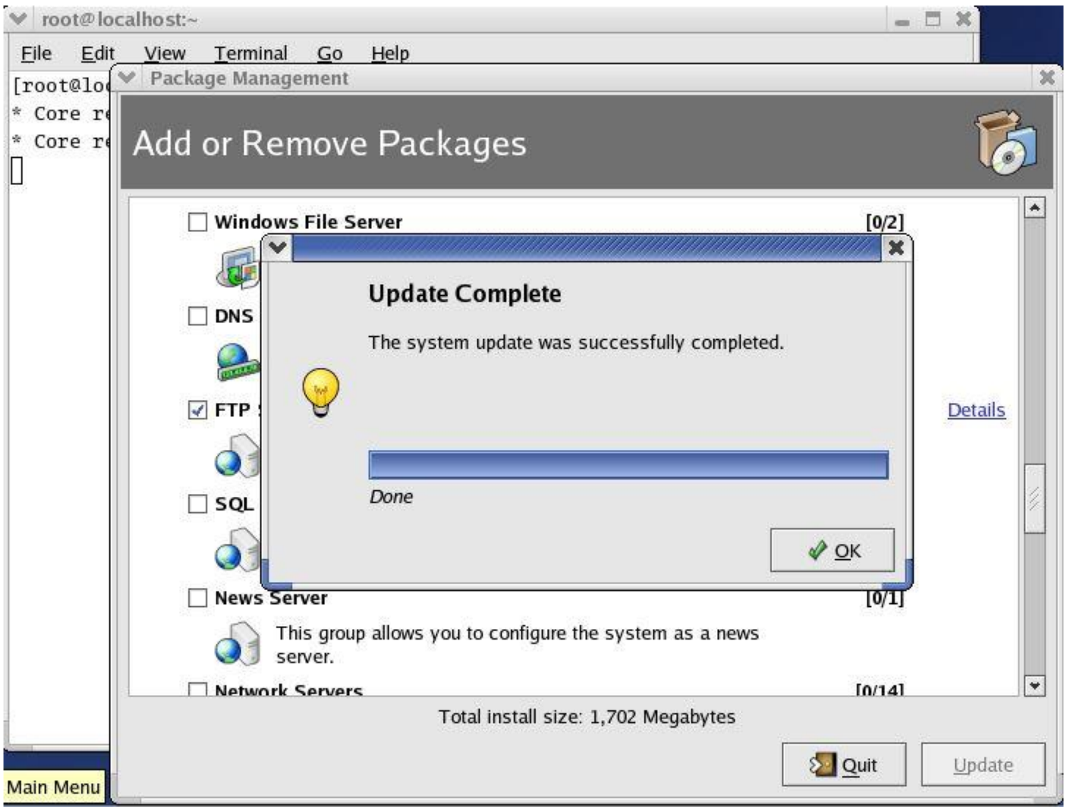
Total install size: 1,702 Megabytes



Completed System Preparation

1 packages are queued for installation
This will take 244 Kilobytes of diskspace.

Total install size: 1,702 Megabytes



Update Complete

The system update was successfully completed.

Done

OK

This group allows you to configure the system as a news server.

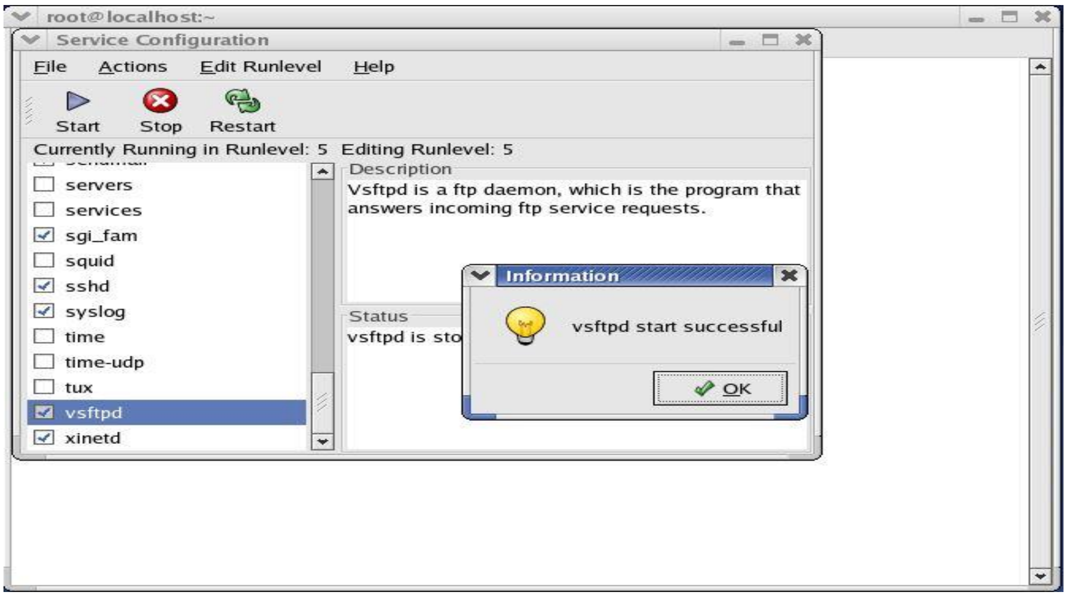
Total install size: 1,702 Megabytes

12.3 Starting your FTP Server

When the installation is done. We need to start the FTP Service, we can use the Service Configuration tool. There are again two ways to start FTP server. These are:

- Main Menu | System Settings | Server Settings | Services
- $ redhat−config−services

Again, you'll be prompted for the root password, unless you're already logged on as root. After logging in as a root, it will open service configuration dialog.



Currently Running in Runlevel: 5 Editing Runlevel: 5

Description
Vsftpd is a ftp daemon, which is the program that answers incoming ftp service requests.

Status
vsftpd is sto

vsftpd start successful

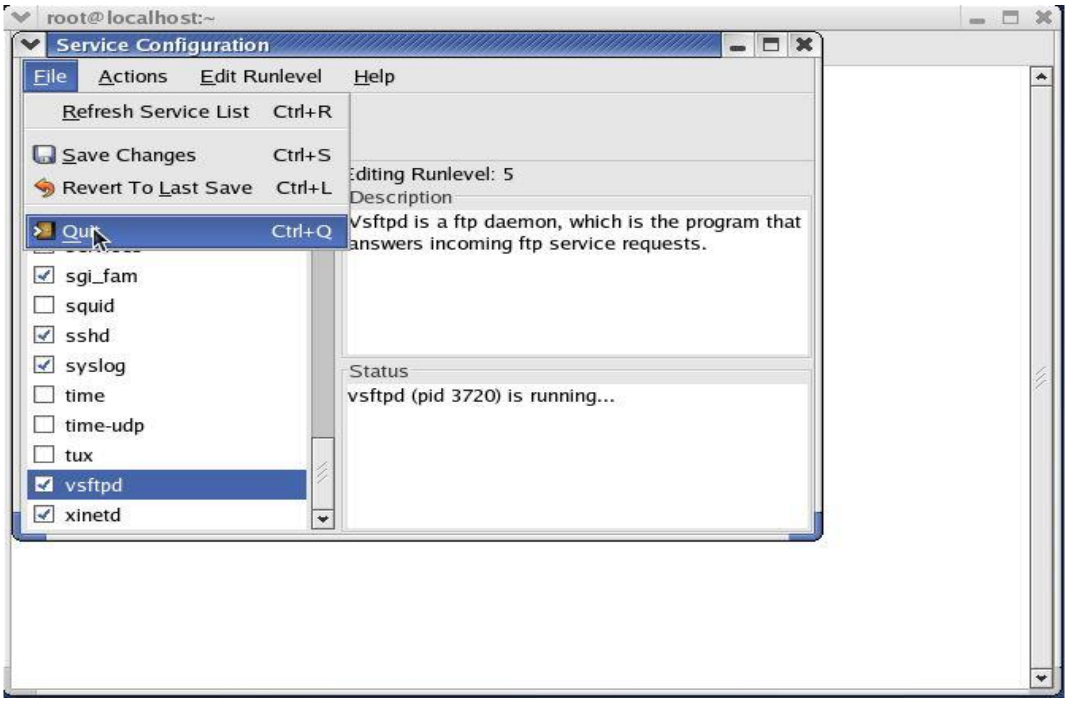
Here we need to start the service. By clicking on start button, it will start the service. After it, we need to save the changes and quit. It's also possible to start and stop these FTP services from the command line, using the service command to start and stop the vsftpd script:

```
# service vsftpd start
Starting vsftpd:                                        [ OK ]
# service vsftpd stop
Stopping vsftpd:                                        [ OK ]
```

Again, if you run the script without an option, the resulting usage message reveals all the available options:

```
# service vsftpd
```

Usage: vsftpd {start|stop|restart|condrestart|status}

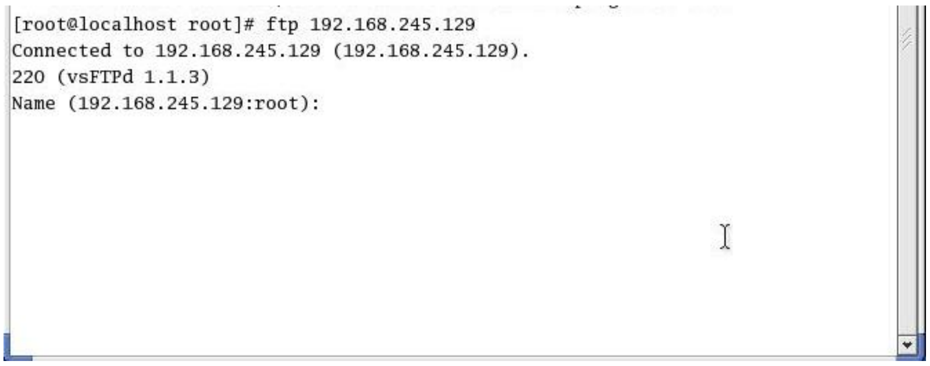


vsftpd is a ftp daemon, which is the program that answers incoming ftp service requests.

vsftpd (pid 3720) is running...

12.4 Testing Your FTP Server

After setting up the FTP server, we need to test it. From the client side, we will test the server whether it is working fine or not. From a command line, issue the ftp command to start an FTP session, naming your FTP server as the server that you want to connect to.You should get a Name login prompt like the one shown above − this is enough to confirm to us that the vsftpd server is running. Press Ctrl−C to terminate this FTP session and return to the command line.



```
[root@localhost root]# ftp 192.168.245.129
Connected to 192.168.245.129 (192.168.245.129).
220 (vsFTPd 1.1.3)
Name (192.168.245.129:root):
```

Configuring an Anonymous FTP Server for File Download

Anonymous users cannot read from just any directory on your Linux server. By default, the vsftpd package creates a directory tree starting at /var/ftp, and enables 'anonymous read access' to this directory and the directory structure beneath it. For this we'll adopt the role of one of these users, and run a client FTP session to access the FTP server,examine the contents of the FTP site, and download a copy of the test file.

Setting up the FTP Server

All we need to do here is place some test content somewhere under the /var/ftp directory, so that other users can access it. The owner of the /var/ftp is the root account, and by default is the only one with permission to write to the directory. For this we need to enter as a root user, so use a command line to switch to the root user:

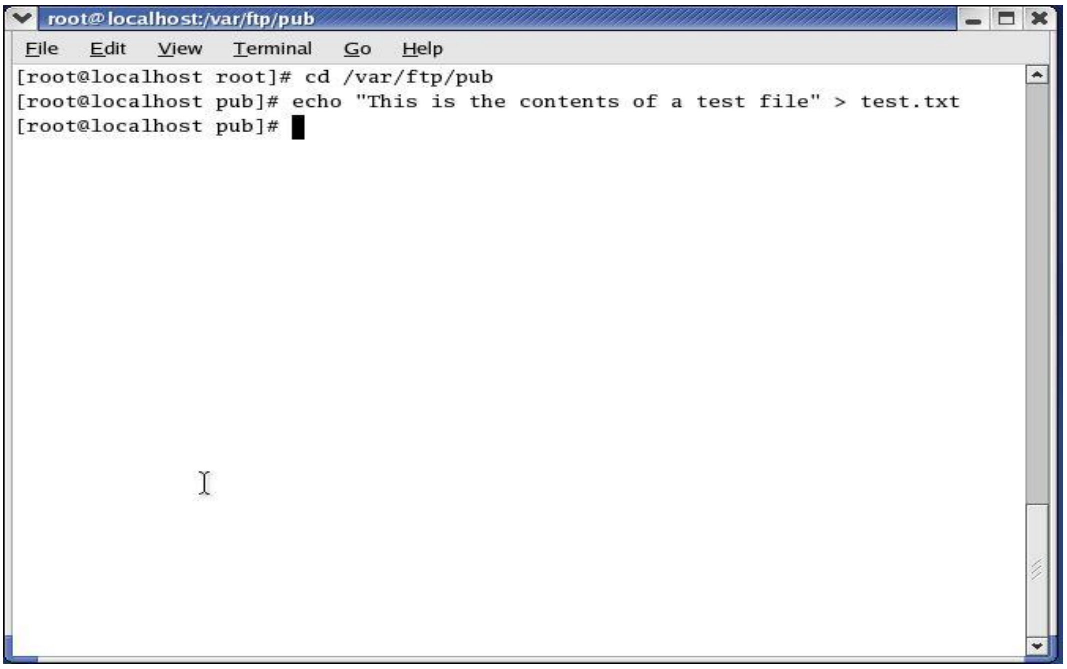
$ su −

Password:

Then you can place whatever content you want under the /var/ftp directory. For example, you can easily use acommand such as echo to create a simple test file:

cd /var/ftp/pub

echo "This is the contents of a test file!" > test.txt



```
[root@localhost root]# cd /var/ftp/pub
[root@localhost pub]# echo "This is the contents of a test file" > test.txt
[root@localhost pub]#
```

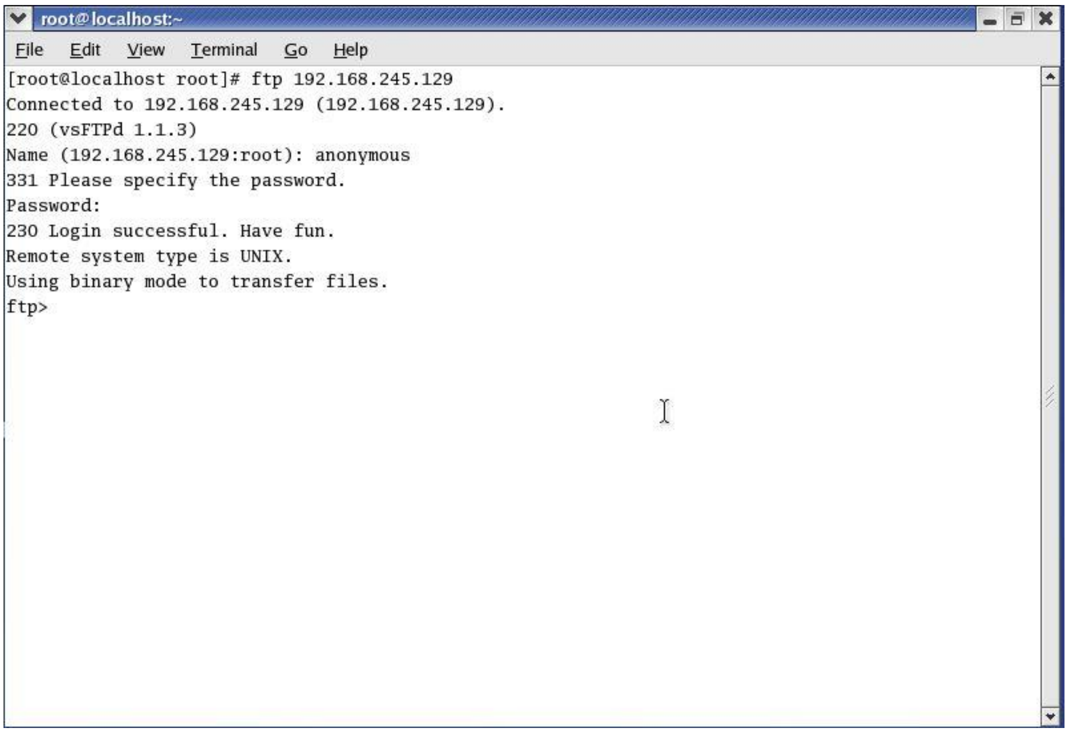
12.5 Using an FTP Client to Test Anonymous Read Access

Now you can test for anonymous read access, by using an FTP client to try to grab a copy of this test file via an FTP connection. You can use any FTP client, and you can test from a Windows or Linux machine −provided the client machine can see the FTP server across a network. You can even use your Linux server as a client if you have only one machine.For example, in both Windows and Linux you can use the ftp program at the command line.In the following, we'll use the ftp program as FTP client to connect to the FTP server, examine the contents of the FTP site, and then download the file test.txt:

1) Start by connecting to the FTP server. When you're prompted for a username, specify anonymous (as shown below) or ftp to indicate that you want anonymous access:

```
$ ftp 192.168.245.129
Connected to 192.168.245.129 (192.168.245.129).
220 (vsFTPd 1.1.3)
```

```
Name (192.168.0.99:none): anonymous
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
```

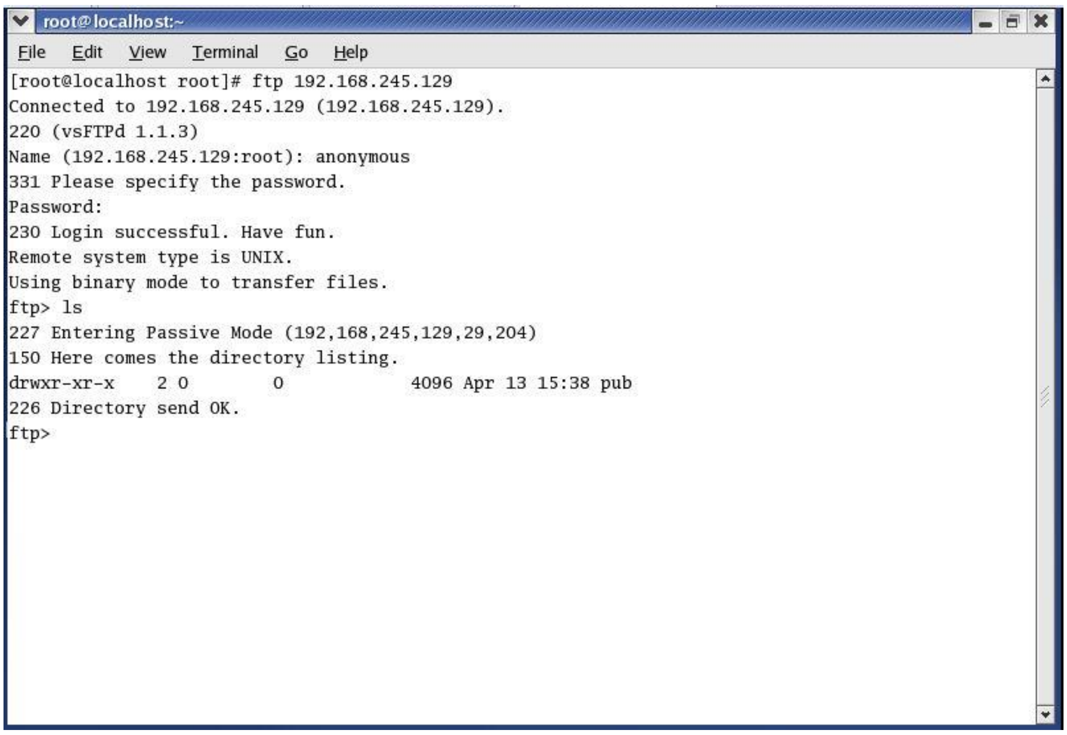


```
[root@localhost root]# ftp 192.168.245.129
Connected to 192.168.245.129 (192.168.245.129).
220 (vsFTPd 1.1.3)
Name (192.168.245.129:root): anonymous
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

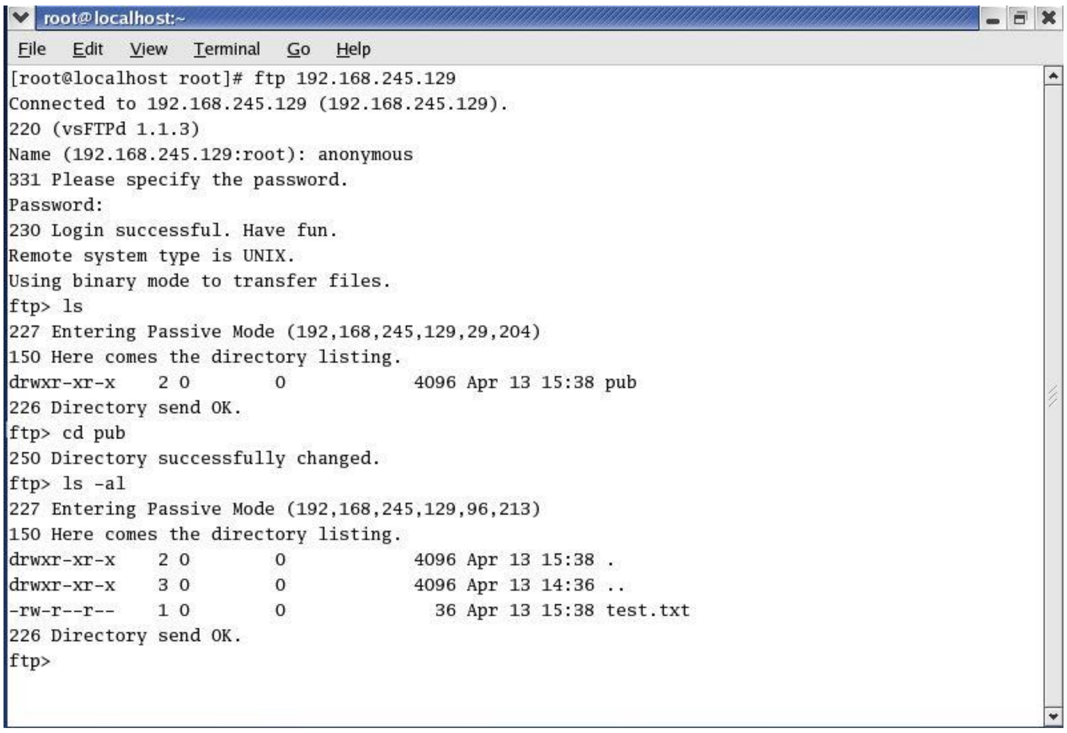
2) Now, we can start to examine the contents of the FTP site that are available to users with anonymous access. For example, here we'll use the ls command to examine the contents of the FTP root directory which happens to be the directory /var/ftp on the server:

```
ftp>ls
220 Entering Passive Mode (192,168,245,129,29,204)
150 Here comes the directory listing.
drwxr−r−x      2 0     0       4096    Apr13   15:38    pub
226 Directory send OK.
```

This shows that the root directory contains just one subdirectory, called pub. Now we'll use cd to change to this directory, and we'll list its contents.



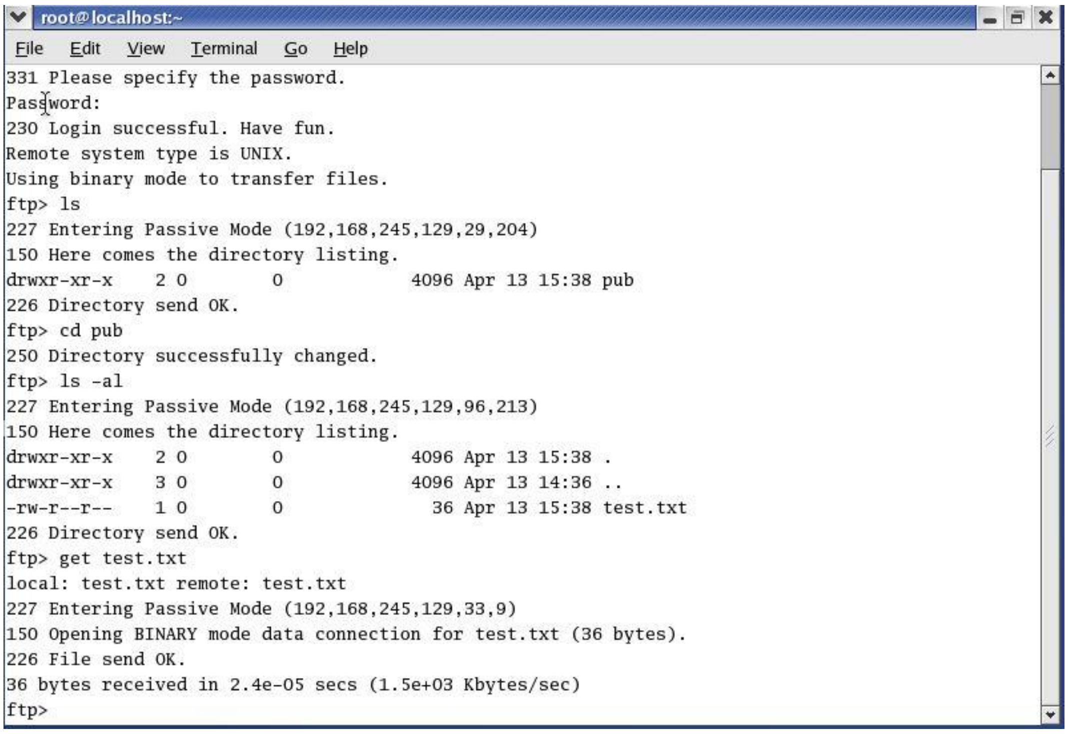
```
[root@localhost root]# ftp 192.168.245.129
Connected to 192.168.245.129 (192.168.245.129).
220 (vsFTPd 1.1.3)
Name (192.168.245.129:root): anonymous
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,245,129,29,204)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13 15:38 pub
226 Directory send OK.
ftp>
```



```
[root@localhost root]# ftp 192.168.245.129
Connected to 192.168.245.129 (192.168.245.129).
220 (vsFTPd 1.1.3)
Name (192.168.245.129:root): anonymous
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,245,129,29,204)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13 15:38 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> ls -al
227 Entering Passive Mode (192,168,245,129,96,213)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13 15:38 .
drwxr-xr-x    3 0        0            4096 Apr 13 14:36 ..
-rw-r--r--    1 0        0              36 Apr 13 15:38 test.txt
226 Directory send OK.
ftp>
```

3) Now, we'll attempt to download the test.txt file we've just located. To do this, we'll use the get command:

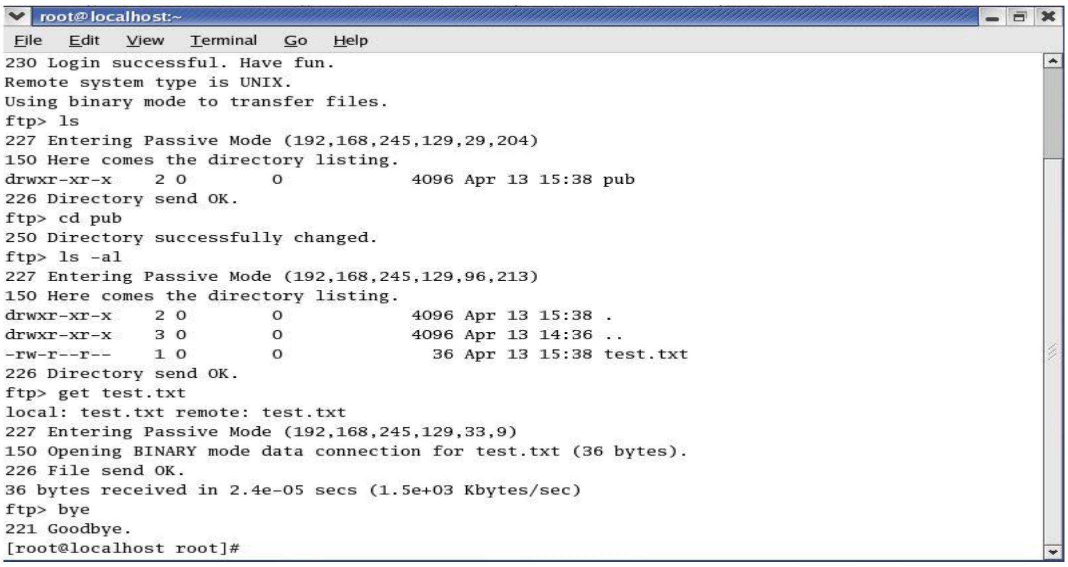
```
ftp> get test.txt
local: test.txt remote: test.txt
227 Entering Passive Mode (192, 168, 245, 129, 33, 9)
150 Opening BINARY mode data connection for test.txt (36 bytes).

226 File send OK.
```



```
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,245,129,29,204)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13 15:38 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> ls -al
227 Entering Passive Mode (192,168,245,129,96,213)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13 15:38 .
drwxr-xr-x    3 0        0            4096 Apr 13 14:36 ..
-rw-r--r--    1 0        0              36 Apr 13 15:38 test.txt
226 Directory send OK.
ftp> get test.txt
local: test.txt remote: test.txt
227 Entering Passive Mode (192,168,245,129,33,9)
150 Opening BINARY mode data connection for test.txt (36 bytes).
226 File send OK.
36 bytes received in 2.4e-05 secs (1.5e+03 Kbytes/sec)
ftp>
```

4) Finally, we'll end the session:

```
ftp> bye
221 Goodbye.

$
```



```
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,245,129,29,204)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13 15:38 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> ls -al
227 Entering Passive Mode (192,168,245,129,96,213)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Apr 13 15:38 .
drwxr-xr-x    3 0        0            4096 Apr 13 14:36 ..
-rw-r--r--    1 0        0              36 Apr 13 15:38 test.txt
226 Directory send OK.
ftp> get test.txt
local: test.txt remote: test.txt
227 Entering Passive Mode (192,168,245,129,33,9)
150 Opening BINARY mode data connection for test.txt (36 bytes).
226 File send OK.
36 bytes received in 2.4e-05 secs (1.5e+03 Kbytes/sec)
ftp> bye
221 Goodbye.
[root@localhost root]#
```

12.6 Configuring an Anonymous FTP Server for File Upload

Anonymous FTP users can write only to the directories that we allow them to write to.
By default, vsftpd does not allow users to upload to the FTP server at all; we must first configure the server toallow anonymous users write access to some directory. So, we'll set up the FTP server for anonymous write access first; then we'll test it again using an FTP client.

Setting up the FTP Server for Anonymous Write Access

There are four steps here. We'll need to create the folder, set the appropriate permissions, and then enable uploading in the FTP server configuration:

1. First, we need to create a writeable directory. Again, you'll need the root account for this.

   Let's createa directory called /upload (in the /var/ftp/pub directory):
   ```
   # cd /var/ftp/pub
   # mkdir upload
   ```

2. Next, we need to set the permission of the upload directory so that it allows write−only access to anonymous FTP users (so that they can write to the directory but not to download from it – this restricts file sharing among FTP users). To do this, we'll first use the chgrp command to change the group associated with the upload directory:
   ```
   # chgrp ftp upload
   ```

   Now, the owner of the folder is still root, but the directory's group is ftp − the set of FTP users. Nowwe'll use the chmod command to assign read/write/execute access to the owner, write/access only tothe group, and deny access to other users:
   ```
   # chmod −R u=rwx, g=wx, o−rxw upload
   ```

3. Finally, we must configure the vsftpd server to allow anonymous upload. To do this, we simply editthe configuration file, /etc/vsftpd/vsftpd.conf. Open this file using gedit (or your favorite text editor),and locate the following lines:
   ```
   # Uncomment this to allow the anonymous FTP user to upload files. This only
   # has an effect if the above global write enable is activated. Also, you will
   # obviously need to create a directory writable by the FTP user.
   #anon_upload_enable=YES
   ```

   Just remove the leading # character in the last line, and save the file:
   ```
   anon_upload_enable=YES
   ```

4. Finally, restart the vsftpd service by using the Restart button in the Server Configuration dialog, ortyping the following at the command line:
   ```
   # service vsftpd restart
   ```

12.7 Using an FTP Client to Test Anonymous Write Access

So, let's test our configuration with another simple session on our FTP client:

1. Connect to the client and log in (using the username anonymous or ftp) as you did before:
```
$ ftp 192.168.0.99
Connected to 192.168.0.99 (192.168.0.99).
220 (vsFTPd 1.1.3)

Name (192.168.0.99:none): anonymous

331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
```
2. Change directory to the pub/upload directory. Try to list its contents − you'll find that you can't,

because that's the way we configured the permissions on the upload directory:
```
ftp> cd /pub/upload
250 Directory successfully changed.
ftp> ls
227 Entering Passive Mode (192, 168, 0, 99, 95, 148)
150 Here comes the directory listing.
226 Transfer done (but failed to open directory).
```

3. However, you can upload a file. To prove it, use the put command to upload a simple file like this:

```
ftp> put uploadtest.txt
local: uploadtest.txt remote: uploadtest.txt
227 Entering Passive Mode (192,168,0,99,133,229)
150 Ok to send data.
226 File receive OK.
40 bytes send in 0.000101 secs (2.1e+02 Kbytes/sec)
```

4. That's it. Now you can close the FTP session:

```
ftp> bye
221 Goodbye.
#
```

Summary

- If you want to enable other users to download files from a location on your server's hard disk, and/or to upload files to that location, then one solution is to install an FTP server.
- FTP is not considered a secure protocol, because communication between the FTP client and server are unencrypted.
- The easiest way to install the vsftpd FTP Server package is via the RPM GUI tool. One way is: Main Menu | System Settings | Add/Remove Applications and another is $ redhat-config-packages.
- Press Ctrl−C to terminate this FTP session and return to the command line.
- FTP is a TCP protocol that is designed specifically for the transfer of files over a network, and it's oneof the oldest Internet protocols still in widespread use.

Keywords

- **FTP Client:** When users want to upload or download from your FTP server, they use a program called an FTP client.
- **File Transfer Protocol:** These communications between FTP server and FTP client take place using the File Transfer Protocol (FTP).
- **vsftpd:** It is a simplified FTP server implementation. It is designed to be a very secure FTP server and can also be configured to allow anonymous access.
- **FTP Server:**FTP servers are the solutions used to facilitate file transfers across the internet. If you send files using FTP, files are either uploaded or downloaded to the FTP server.
- **TUX:** It is a kernel−based, threaded, extremely high-performance HTTP server, which also has FTP capabilities. TUX is perhaps the best in terms of performance but offers less functionality than other FTP server software. TUX is installed by default with Red Hat Linux 9.

Self Assessment

1. FTP is

A. Easy to use

B. Free

C. Internet standard protocol for file transfer

D. All of the above mentioned

2. How can we open the RPM's GUI tool?
A. Main Menu | System Settings | Add/ Remove Applications
B. $ redhat-config-packages
C. By using either of these ways
D. None of the above

3. FTP stands for
A. File transfer protocol
B. First transfer protocol
C. First temperature protocol
D. None of the above

4. Who is the owner of /var/ftp?
A. All normal users
B. Only root
C. Owner can be anyone
D. None of the above

5. Who can install FTP in the system?
A. Only root
B. Normal user
C. Any of the above
D. None of the above

6. Why FTP client program is used?
A. To upload the files to FTP server
B. To download the files from FTP server
C. Both upload and download
D. None of the above

7. Which utility is used to change the directories?
A. cd
B. changedir
C. chdirectory
D. None of the above

8. Which key combination is used to terminate the FTP session?
A. CTRL-A
B. CTRL-B
C. CTRL-C
D. CTRL-D
9. Why FTP is not considered secure?
A. Communications are fake
B. Communications are unencrypted

C. Communications are available to everyone

D. None of the above

10. FTP server can be considered as

A. Area of disk space used for storing files

B. Software to allow access

C. Configuration files to allow access

D. All of the above mentioned

11. FTP is a

A. TCP protocol

B. HTTP protocol

C. SMTP protocol

D. None of the above

12. SFTP stands for

A. Second file transfer protocol

B. Secure file transfer protocol

C. Steamed file transfer protocol

D. None of the above

13. In FTP server, the software and configuration files are required for

A. Giving the user access for download

B. Giving the user access for upload

C. Both mentioned above

D. None of the above

14. What indicates the relevance of FTP today?

A. Availability of different FTP client programs

B. Many OS come with FTP preinstalled

C. Both above mentioned

D. None of the above

15. Which of these are FTP servers?

A. vsftpd

B. TUX

C. Both of the above

D. None of the above

Answers for Self Assessment

1.	D	2.	C	3.	A	4.	B	5.	A
6.	C	7.	A	8.	C	9.	B	10.	D

11.	A	12.	B	13.	C	14.	C	15.	C

Review Questions

1. What is a FTP server? Write the FTP servers available in Red Hat Linux Distribution.
2. Write the installation procedure of vsftpd FTP server. How can we start the FTP server?
3. After installation and testing of FTP server, how can we use it? Explain various issues encountered in this.
4. How can we configure anonymous FTP server for file download?
5. How can we configure anonymous FTP server for file upload?

Further Readings

Sandip Bhattacharya, Pancrazio De Mauro, Shishir Gundavaram, Mark Mamone, Kalip Sharma, Deepak Thomas and Simon Whiting, Beginning Red Hat Linux 9, Wiley Publishing, Inc.

Web Links

https://www.techopedia.com/definition/26108/ftp-server

Unit 13: Samba Servers

CONTENTS

Objectives

After studying this unit, you will be able to

- Understand the SAMBA server
- Installing, starting and stopping the server
- SAMBA configuration with SWAT, starting SWAT service
- Adding SAMBA user
- Creating and configuring SAMBA share

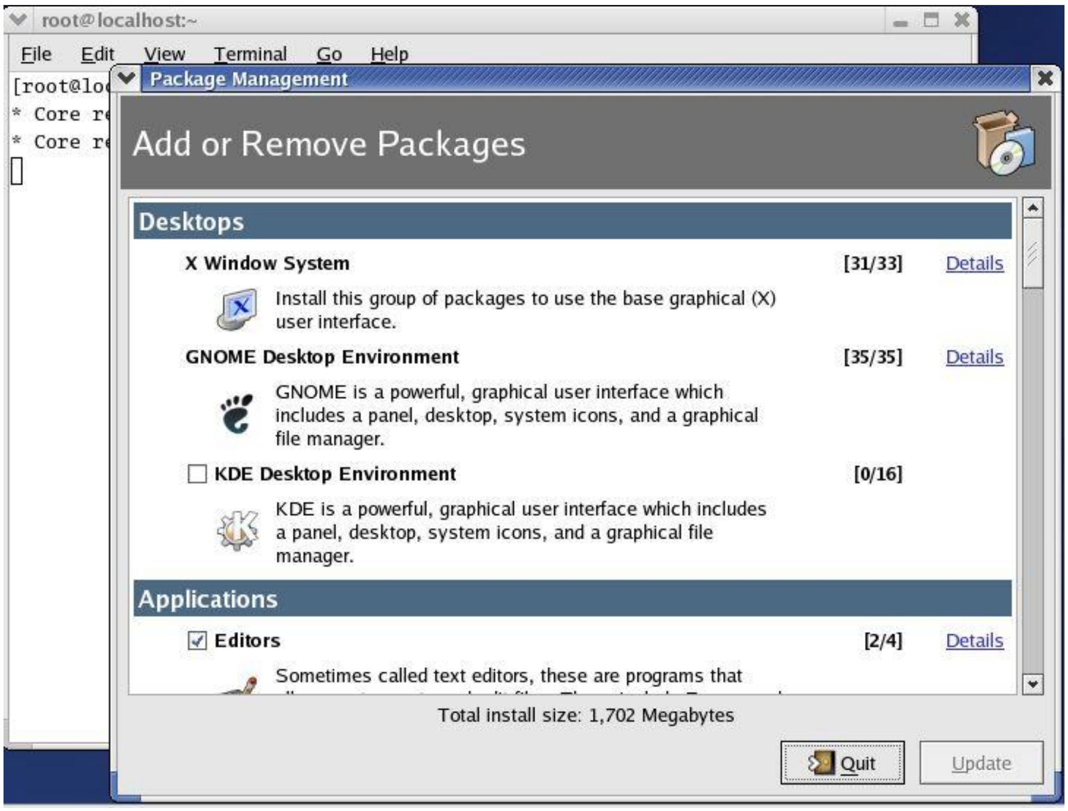
Introduction

Samba is an implementation of the Windows SMB and CIFS protocols on Unix. The Samba project was started by Andrew Tridgell, who wanted to mount his Unix server disk space onto a DOS PC. When he'd solved the problem, Tridgell discovered that what he'd built was an implementation of the SMB (servermessage block) protocol − a protocol for sharing files and other resources. Tridgell named his implementation

Samba, and published version 1.0 in early 1992. Since that time, the Samba project has grown tremendously,and today there is still Samba development going on in the open-source community.So, Samba is a collection of programs that make it possible to share files and printers between computersequipped to use the SMB protocol − Windows by default, Linux/Unix with Samba, and (more recently) MacOS X. Samba is freely available under GNU General Public License and is included as part of the Red HatLinux 9 distribution.

13.1 Installing SAMBA

The installation of SAMBA Server suite can be done via the RPM GUI tool. One of the way of this is Main Menu | System Settings | Add/Remove Applications and another way is by using the terminal and writing the command $ redhat-config-packages.



root@localhost:~

File Edit View Terminal Go Help

```
[root@lo
* Core r
* Core r
```

Package Management

Add or Remove Packages

Desktops

X Window System [31/33] Details

Install this group of packages to use the base graphical (X) user interface.

GNOME Desktop Environment [35/35] Details

GNOME is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

☐ KDE Desktop Environment [0/16]

KDE is a powerful, graphical user interface which includes a panel, desktop, system icons, and a graphical file manager.

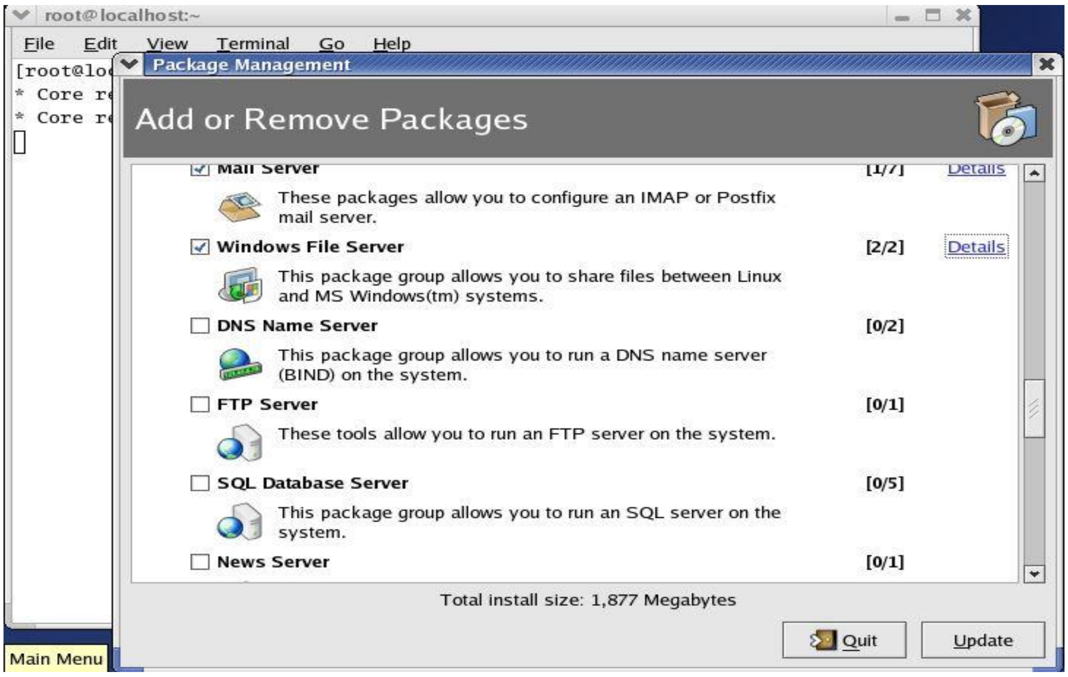
Applications

☑ Editors [2/4] Details

Sometimes called text editors, these are programs that

Total install size: 1,702 Megabytes

In package management window, we will look for the servers category. In servers, we will go for Windows File Server.



root@localhost:~

File Edit View Terminal Go Help

```
[root@lo
* Core r
* Core r
```

Package Management

Add or Remove Packages

☑ Mail Server [1/7] Details

These packages allow you to configure an IMAP or Postfix mail server.

☑ Windows File Server [2/2] Details

This package group allows you to share files between Linux and MS Windows(tm) systems.

☐ DNS Name Server [0/2]

This package group allows you to run a DNS name server (BIND) on the system.

☐ FTP Server [0/1]

These tools allow you to run an FTP server on the system.

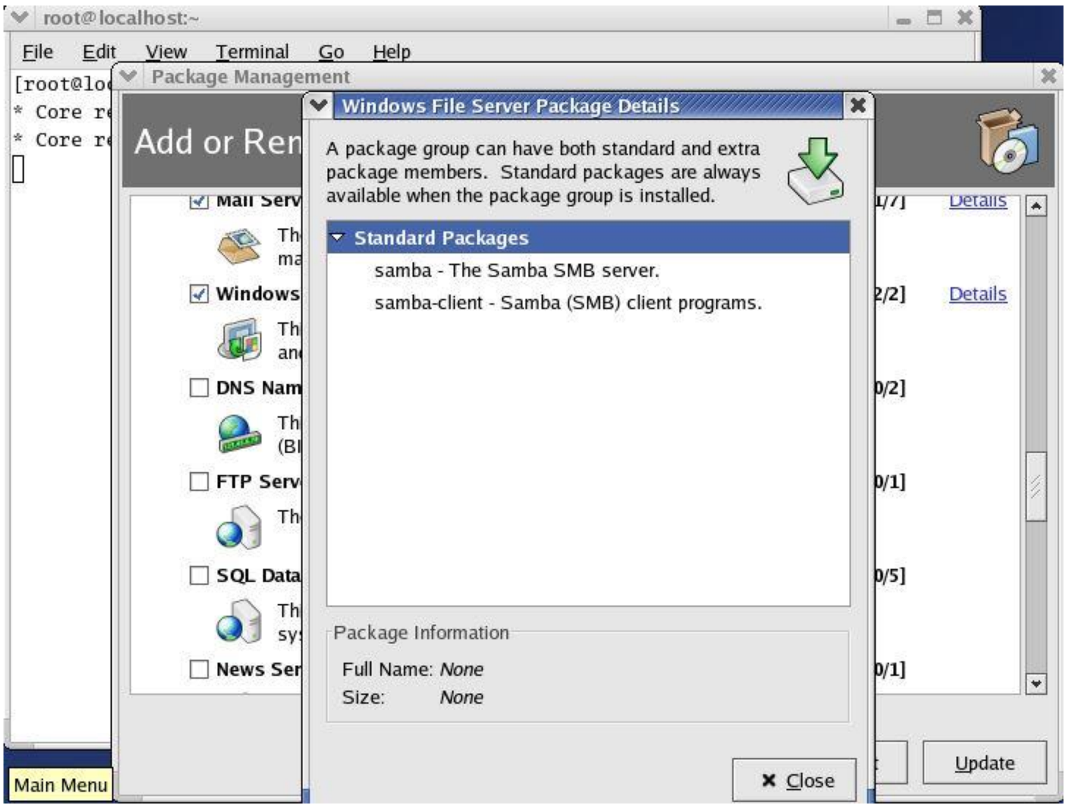
☐ SQL Database Server [0/5]

This package group allows you to run an SQL server on the system.

☐ News Server [0/1]

Total install size: 1,877 Megabytes

By clicking on the details, we can see the packages available in this. There are two standard packages available in this. These are: samba which is the Samba SMB server and samba-client which is SMB client programs.


Windows File Server Package Details

A package group can have both standard and extra package members. Standard packages are always available when the package group is installed.

Standard Packages

samba - The Samba SMB server.

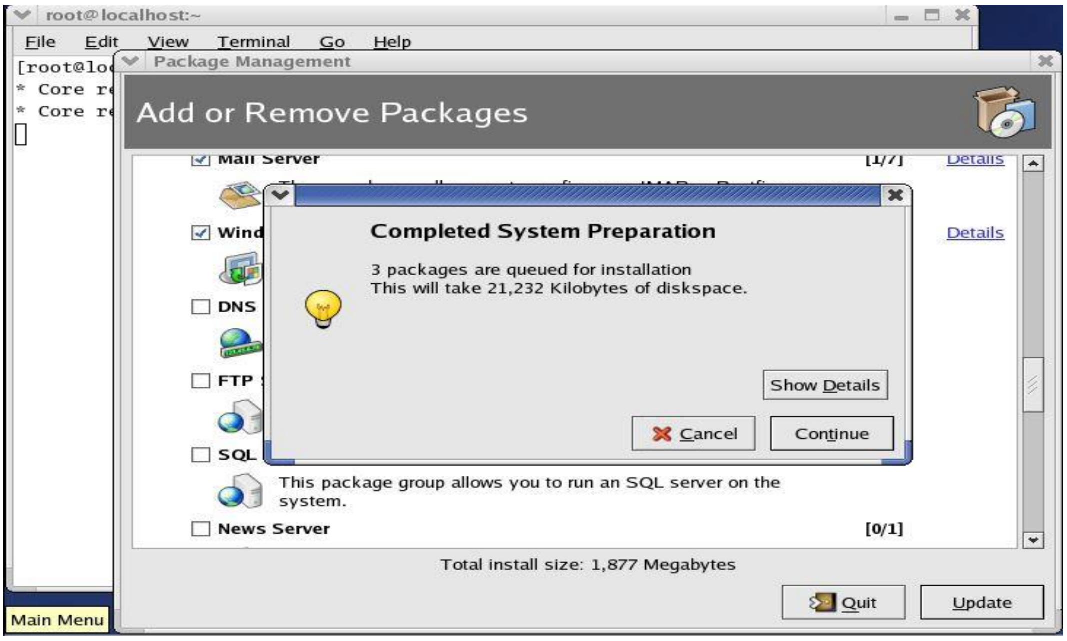
samba-client - Samba (SMB) client programs.

Package Information

Full Name: *None*

Size: *None*

After clicking on update button, the packages will be installed. The popup will show the required diskspace. After clicking on continue, it will queue the packages for installation.

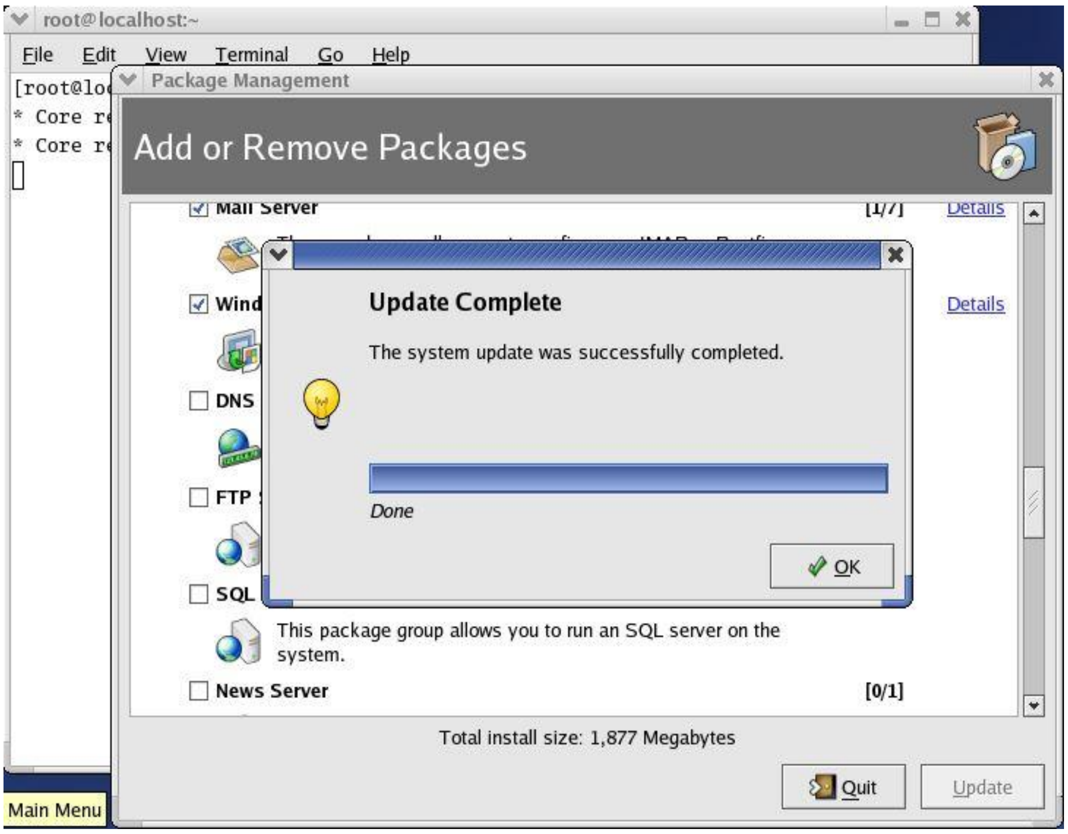

Completed System Preparation

3 packages are queued for installation
This will take 21,232 Kilobytes of diskspace.

This package group allows you to run an SQL server on the system.

Total install size: 1,877 Megabytes

After few seconds, the update will complete. Click on OK.



Update Complete

The system update was successfully completed.

Done

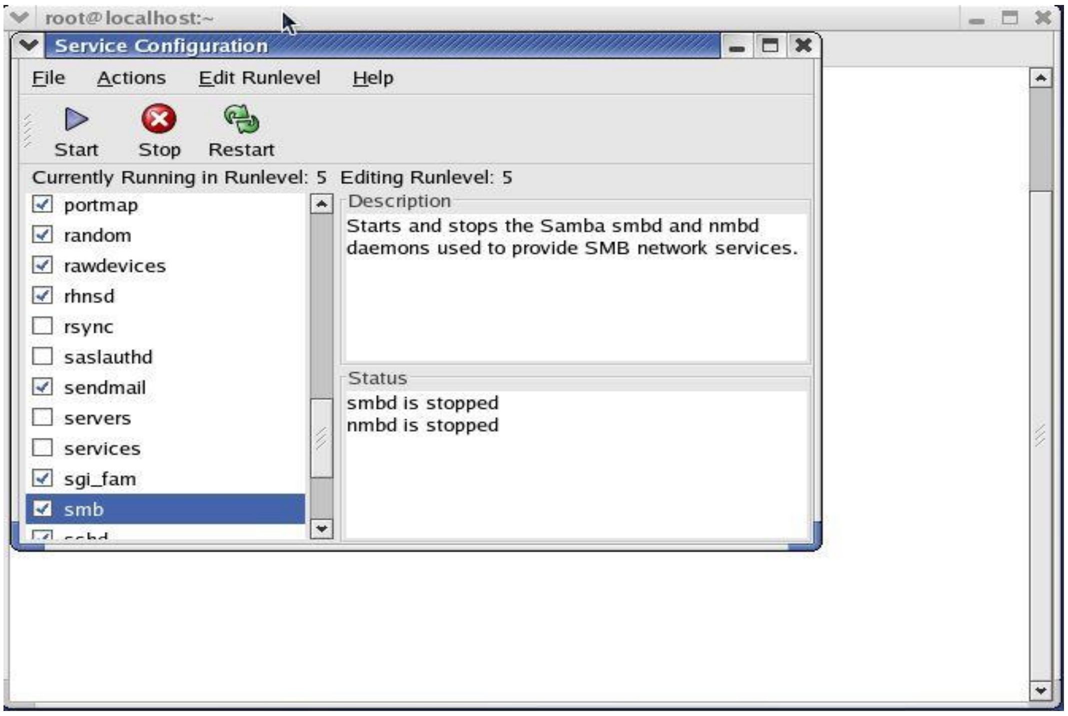
OK

This package group allows you to run an SQL server on the system.

Total install size: 1,877 Megabytes

13.2 Starting and Stopping the Samba Service

There are several ways to start and stop the Samba service. Once again,we can do so via the Service Configuration GUI tool. To launch the tool, select Main Menu | System Settings |Server Settings | Services or type the following command at the command line:`$ redhat-config-services`. Click on smb in this. Now the service is stpped. We need to start it by clicking on the start button.



Currently Running in Runlevel: 5 Editing Runlevel: 5

Description
Starts and stops the Samba smbd and nmbd daemons used to provide SMB network services.

Status
smbd is stopped
nmbd is stopped

It's also a good idea to check the checkbox, to configure the samba service to start automatically whenever you boot up the system. For example, if you ever have to perform an emergency reboot on your file server, then the "automatic start" configuration means that the file server is immediately available to users after the reboot. When you've done this, select File | Save Changes to save your new setting.

Alternatively, you can also stop and start smb service at the command line, using the service command to run the /etc/rc.d/init.d/smb script we mentioned earlier. Typing the script name at the command line like this reveals the possible usages:

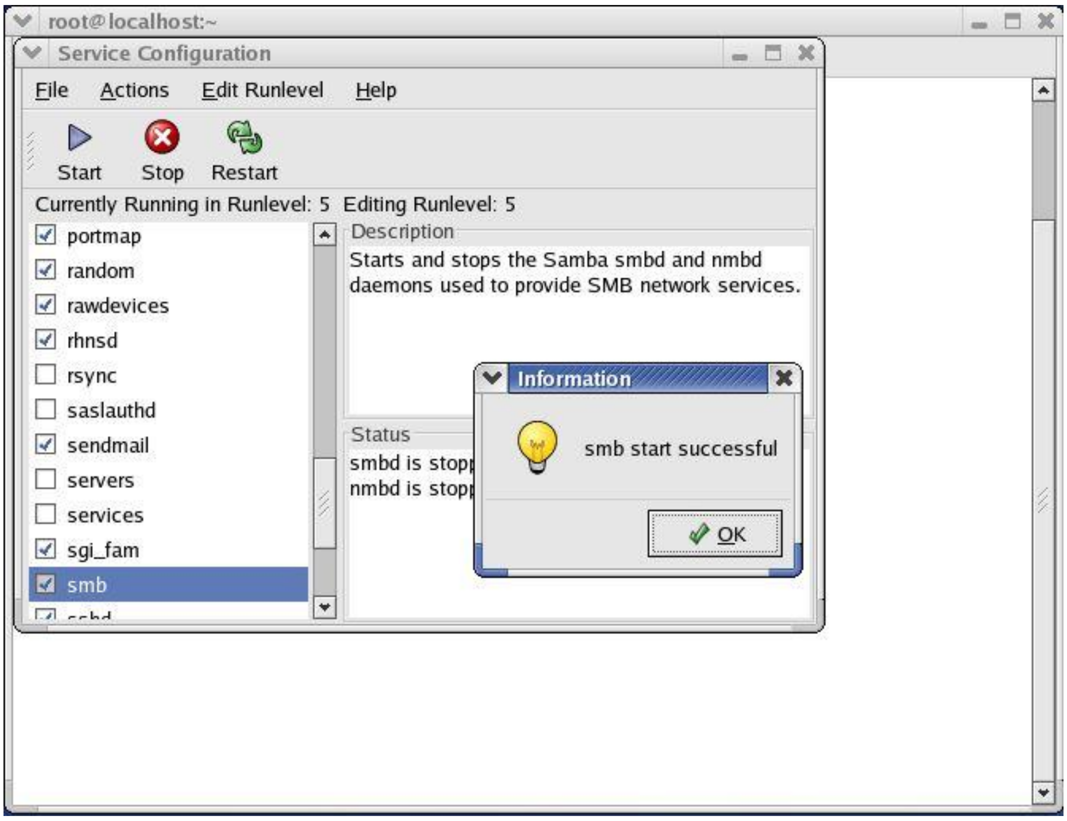
```
# servicesmb

Usage: /etc/rc.d/smb        {start|stop|restart|reload]status|condrestart}
```

As you can see, it works in much the same way as the httpd and vsftpd scripts we've seen in earlier sections of this chapter. So, to start the service we'd type this:

```
# servicesmb start

Starting SMB services:                                    [ OK ]

Starting NMB services:                                    [ OK ]
```

This command starts both SMB and NMB (NetBIOS name server), which are both services related to Samba. To stop the service, we'd type this:

```
# servicesmb stop

Shutting down SMB services:                               [ OK ]

Shutting down NMB services:                               [ OK ]
```



Currently Running in Runlevel: 5 Editing Runlevel: 5

- ✓ portmap
- ✓ random
- ✓ rawdevices
- ✓ rhnsd
- ☐ rsync
- ☐ saslauthd
- ✓ sendmail
- ☐ servers
- ☐ services
- ✓ sgi_fam
- ✓ smb

Description

Starts and stops the Samba smbd and nmbd daemons used to provide SMB network services.

Status

smbd is stopp
nmbd is stopp

smb start successful

13.3 Samba Configuration Files and Utilities

Samba's configuration files are contained in the directory /etc/samba.

Configuration File	Description
smb.conf	This is the main configuration file for Samba.
lmhosts	This contains Samba's NetBIOS−to−IP address mappings.
secrets.tdb	This is the Samba secrets database. It stores private information such as the local SID and machine trust password. It is generated by machine and cannot be read in a text editor.
smbusers	This is a text file that maps your Linux system's users to various SMB−specific usernames.
smbpasswd	This is an encrypted password file. This file doesn't exist when you first install Samba, but is created when you add Samba users.

Some of themost important utilities provided by Samba. They're all contained in the directory /usr/bin.

Program	Purpose
smbclient	This is an FTP−like client, used to access SMB/CIFS resources on a file server.
smbadduser	This is a script, used for adding Samba users. It updates the smbusers and smbpasswd files.
smbpasswd	This changes a Samba user's SMB password. It is similar to the Unix passwd command
smbmount	This is used to mount an SMB filesystem.
smbumount	This is used to unmount an SMB file system.
smbstatus	This lists the current Samba connections.
testparm	This checks the smb.conf configuration file for correctness.
nmblookup	This is used to query NetBIOS names and map them to IP addresses in a network using NetBIOS over TCP/IP.

In addition, we'll also make use of the script /etc/rc.d/init.d/smb, which we use to start and stop the Samba fileservice.

13.4 Samba Configuration with SWAT

Perhaps the easiest way to configure Samba is by using the Samba Web Administration Tool(SWAT). SWAT is a web−based interface, which means that you can use it to configure and manage your Samba server through a web browser − if you want, you can even do it remotely across a network or even across the Internet.

Installing SWAT

1) Launch a terminal window, and switch to the root user account by using this command:
```
$ su–
```
2) Insert Red Hat Linux 9 distribution Disk 2. Change to the directory on the CD that contains the RPM package files: `# cd /mnt/cdrom/RedHat/RPMS`
3) Use the ls command to find out the exact version of samba−swat contained on the disk. For example:
```
# ls samba−swat*.rpm

samba−swat−2.2.7a−6.i386.rpm
```
   If this command doesn't find any matches, then remove the disk, replace it with Disk 1 or Disk 2, and return to Step 2.
4) Install the samba−swat package you've just found, by using the rpm command:
```
# rpm −ivh samba−swat−2.2.7a−6.i386.rpm
```

Starting the SWAT Service

Starting the SWAT service is a two step process:

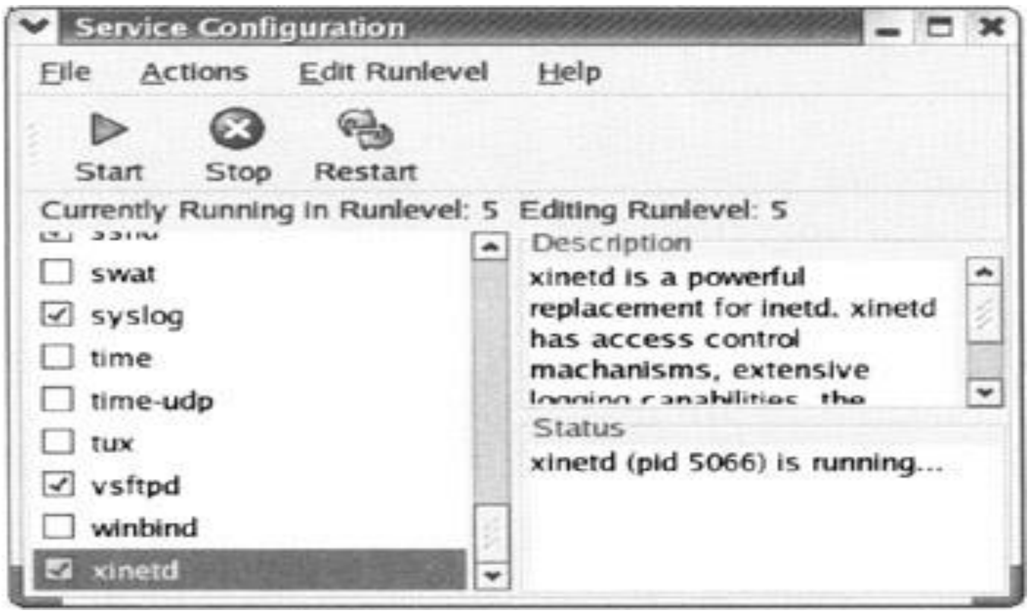
1) Launch gedit or your favorite text editor, and open the file /etc/xinetd.d/swat. This is the configuration file for the SWAT service, and it looks like this:

```
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
# to configure your Samba server. To use SWAT, \
# connect to port 901 with your favorite web browser.
service swat
{
port = 901
socket_type = stream
wait = no
only_from = 127.0.0.1
user = root
server = /usr/sbin/swat
log_on_failure += USERID
disable = yes
}
```

Change disable value, like this: disable = no. Save the file, and close your text editor.

2) Restart the xinetd service. (We must do this because SWAT runs as an xinetd service.

To do this, first launch the Service Configuration GUI tool by selecting Main Menu | System Settings | Server Settings | Services. Then locate and select the service called xinetd, and click Restart. You'll get a dialog to confirm that the restart was successful; then you can exit the tool by clicking on Quit.



Currently Running in Runlevel: 5 Editing Runlevel: 5

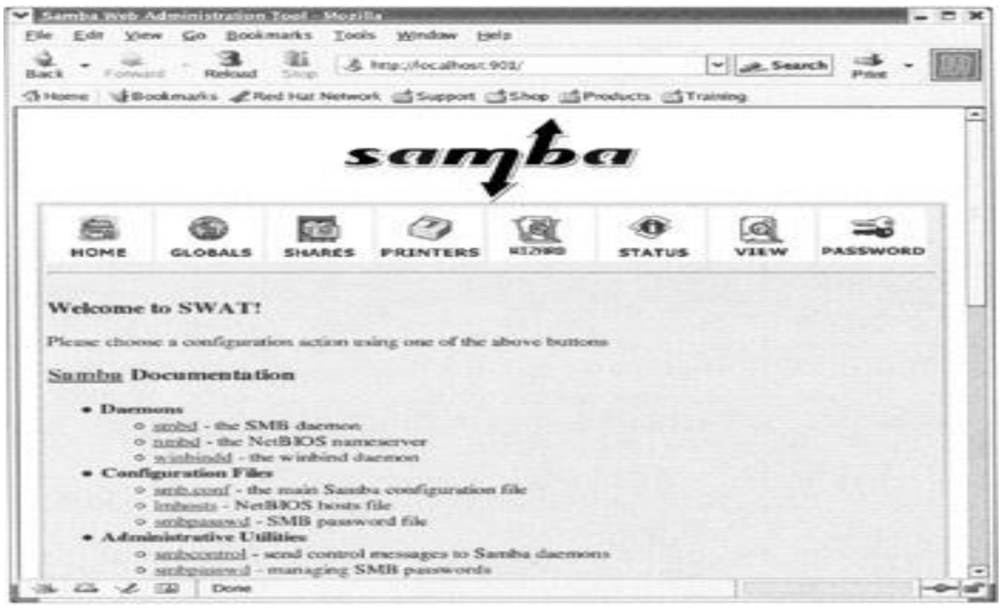
- ☐ swat
- ✓ syslog
- ☐ time
- ☐ time-udp
- ☐ tux
- ✓ vsftpd
- ☐ winbind
- ✓ xinetd

xinetd is a powerful replacement for inetd. xinetd has access control machanisms, extensive

xinetd (pid 5066) is running...

Using SWAT for the First Time

Now we can test SWAT. Open a browser on the Linux server and type in the URL http://localhost:901. You'llbe prompted for a username and password − use the system's root username and password. Then you shouldsee the SWAT HOME page:



Welcome to SWAT!

Please choose a configuration action using one of the above buttons

Samba Documentation

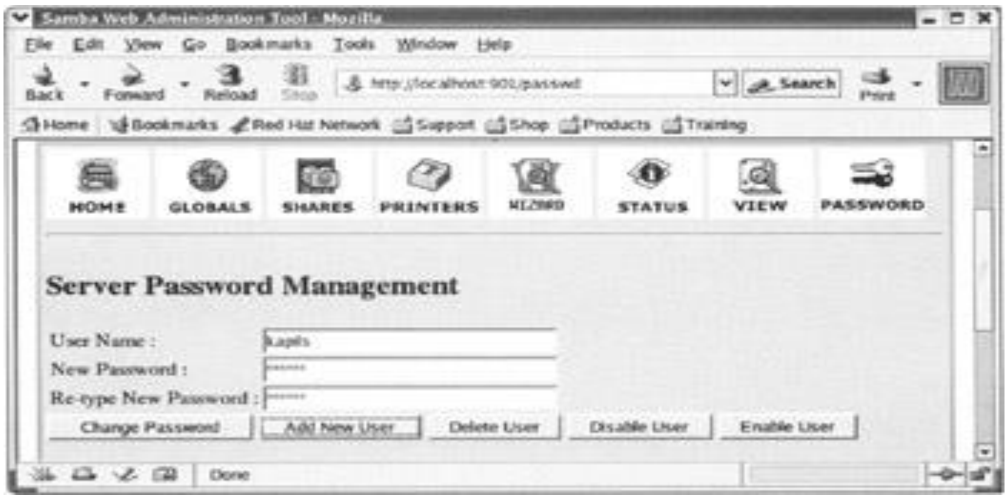
- **Daemons**
  - smbd - the SMB daemon
  - nmbd - the NetBIOS nameserver
  - winbindd - the winbind daemon
- **Configuration Files**
  - smb.conf - the main Samba configuration file
  - lmhosts - NetBIOS hosts file
  - smbpasswd - SMB password file
- **Administrative Utilities**
  - smbcontrol - send control messages to Samba daemons
  - smbpasswd - managing SMB passwords

There are also eight buttons listed across the top of this page, which provide access to SWAT's various utilities:

- **HOME** − the home page of SWAT, and the first page that you see when you fire up the SWAT interface.
- **GLOBALS** − for setting the global variable values of the /etc/samba/smb.conf configuration file.
- **SHARES −** for creating and deleting Samba shares, and setting Samba parameters.
- **PRINTERS −** for creating and deleting Samba printer shares and setting printer parameters.
- **WIZARD −** like GLOBALS, this is also for setting various values in /etc/samba/smb.conf.
- **STATUS −** for viewing the Samba server's status, and starting and stopping Samba−related services
- **VIEW −** for viewing the content of the /etc/samba/smb.conf configuration file
- **PASSWORD −** for adding, removing, enabling, and disabling Samba users, and for setting Samba users' passwords

Adding a Samba User

To grant a system account access to the Samba services, you can use SWAT's PASSWORD feature. Once you've logged into SWAT using the root username and password, click the PASSWORD button, and you'll see the following screen:


Samba Web Administration Tool - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:901/passwd
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
HOME GLOBALS SHARES PRINTERS WIZARD STATUS VIEW PASSWORD
Change Password
Add New User
Delete User
Disable User
Enable User
Done

Server Password Management

User Name : kapils
New Password : ******
Re-type New Password : ******

In the Server Password Management section, enter the name of an existing account on the system, and supply a password. Then click the Add New User button. This will add an entry to Samba's smbpasswd configuration file, to indicate that this user has access to Samba's services.

Creating and Configuring a Samba Share

1) Create a directory, which we'll call /share, to be used for the file server. With root permission, you can do this at the command line using this command: # mkdir /share
2) If you haven't done so already, use the Mozilla browser (or your favorite web browser) to browse to the SWAT home page at http://localhost:901, and log in using the root user account.
3) Click on the GLOBALS toolbar icon. In the Base Options section, use the Workgroup field to enter the name of the workgroup that you want your server to appear in when clients use it. (If you haven't set up a workgroup, then this is probably the default value,

WORKGROUP). You should also name the service, by entering a value in the server string field − it can be any string that you want your Samba clients to see.

4) Now click on SHARES toolbar button at the top of the screen. We'll share our /share directory by giving it a share name; let's call it linuxbox−share. You should first check the entries in the drop−down list on this page, to ensure that your chosen share name hasn't already been used for a different share; then, enter the share name in the Create Share field.
5) Now click on Create Share button to create the share. This will present you with another screen in which you can specify the properties of your share. You will certainly need to set the path field to the path of your share directory (in this example, it's /share).
6) Here, we've set the read only field to No to make it a writeable share; the browseable field to Yes, to allow the contents of the /share directory to be visible; and the available field to Yes, to "enable" the share (that is, make it available to users). We've also added a comment to remind us what the share is for. When you're done, click on the Commit Changes button to commit these settings to the Samba configuration file.
7) Restart the Samba service, so that the configuration changes can take effect. To do this, click on STATUS button and then click on the Restart smbd button to restart the service. Wait for the page to reload, and then click on the Restart nmbd button to restart that service too.

Summary

- Samba is an implementation of the Windows SMB and CIFS protocols on Unix.
- Samba is freely available under GNU General Public License and is included as part of the Red Hat Linux 9 distribution.
- There are two ways to install the SAMBA Server suite via the RPM GUI tool: Main Menu | System Settings | Add/Remove Applications and $ redhat-config-packages.
- There are two standard packages available in this: samba and samba-client.
- There are two ways to start the SAMBA Service, we can use the Service Configuration tool. These are: Main Menu | System Settings | Server Settings | Services and $ redhat−config−services.
- If you ever have to perform an emergency reboot on your file server, then the "automatic start" configuration means that the file server is immediately available to users after the reboot.

Keywords

- **Samba:** It is is a collection of programs that make it possible to share files and printers between computers equipped to use the SMB protocol − Windows by default and Linux/Unix with Samba.
- **Windows File Server:** This package group allows you to share files between Linux and MS windows system.
- **smb.conf:** This is the main configuration file for Samba.
- **Smbclient:** This is an FTP−like client, used to access SMB/CIFS resources on a file server.
- **SWAT:** It is a web−based interface, which means that you can use it to configure and manage your Samba server through a web browser − if you want, you can even do it remotely across a network or even across the Internet.

- **Xinetd:** xinetd is an open-source super-server daemon which runs on many Unix-like systems and manages Internet-based connectivity.

Self Assessment

1. In package group "Windows File Server", which standard packages are available?
A. samba
B. samba-client
C. Both of the above
D. None of the above

2. Which of these is the main configuration file of SAMBA?
A. secrets.tdb
B. smb.conf
C. samba.conf
D. imhosts

3. Starting of SWAT service is _______ step process.
A. One
B. Two
C. Three
D. Four

4. SAMBA server on UNIX is an implementation of
A. Windows SMB
B. CIFS protocol
C. Both of the above
D. None of the above

5. Which server group will be chosen for installing SAMBA server?
A. DNS name server
B. FTP server
C. Mail server
D. Windows File server

6. Which of these commands will stop the smb service?
A. # servicesmb stop
B. # servicesmb end
C. # service stop smb
D. # service end smb

7. Which of these is unmount an SMB file system?
A. smbunmount

B. smbumount

C. smbmount

D. None of the above

8. For configuration of SAMBA, which service is required?

A. SWAT

B. FTP

C. TCP

D. None of the above

9. SWAT service will run as ________ service.

A. FTP

B. TCP

C. xinted

D. None of the above

10. Which of these utility is used to create a directory?

A. crtdir

B. mkdir

C. dircreate

D. None of the above

11. Because of the SAMBA server, it is possible to

A. Share files between computers

B. Share printers between computers

C. Both of the above

D. None of the above

12. SAMBA server on UNIX is an implementation of

A. Windows SMB

B. CIFS protocol

C. Both of the above

D. None of the above

13. How can we start the RPM GUI tool?

A. Main Menu | System Settings | Add/Remove Applications

B. $ redhat-config-packages

C. By using either of the way mentioned above

D. Something other than this

14. Which service needs to be started for SAMBA server through service configuration?

A. smbd

B. nmbd

C. Both of the above mentioned
D. None of the above

15. By using which way, we can start the SAMBA service?
A. Main Menu | System Settings | Server Settings | Services
B. $ redhat−config−services
C. By using either of the way mentioned above
D. None of the above

Answers for Self Assessment

1.	C	2.	B	3.	B	4.	C	5.	D
6.	A	7.	B	8.	A	9.	C	10.	B
11.	C	12.	C	13.	C	14.	C	15.	C

Review Questions

1. What is a SAMBA server? What are the different ways to start RPM GUI tool for installation of SAMBA server?
2. What are the standard packages for SAMBA servers? How can we install the SAMBA server in the Linux system?
3. How can we start and stop the SAMBA service?
4. What are the different configuration files and utilities of SAMBA server? Explain.
5. What is SWAT? How can we install SWAT?
6. How can we start the SWAT service? How to use the SWAT service for first time?
7. How can we create and configure a SAMBA share?

Further Readings

Sandip Bhattacharya, Pancrazio De Mauro, Shishir Gundavaram, Mark Mamone, Kalip Sharma, Deepak Thomas and Simon Whiting, Beginning Red Hat Linux 9, Wiley Publishing, Inc.

Web Links

https://linux.die.net/man/8/swat

Unit 14: Network File Systems

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand NFS
- Plan NFS installation
- Configure NFS server and client
- Use automount service
- Examine NFS security

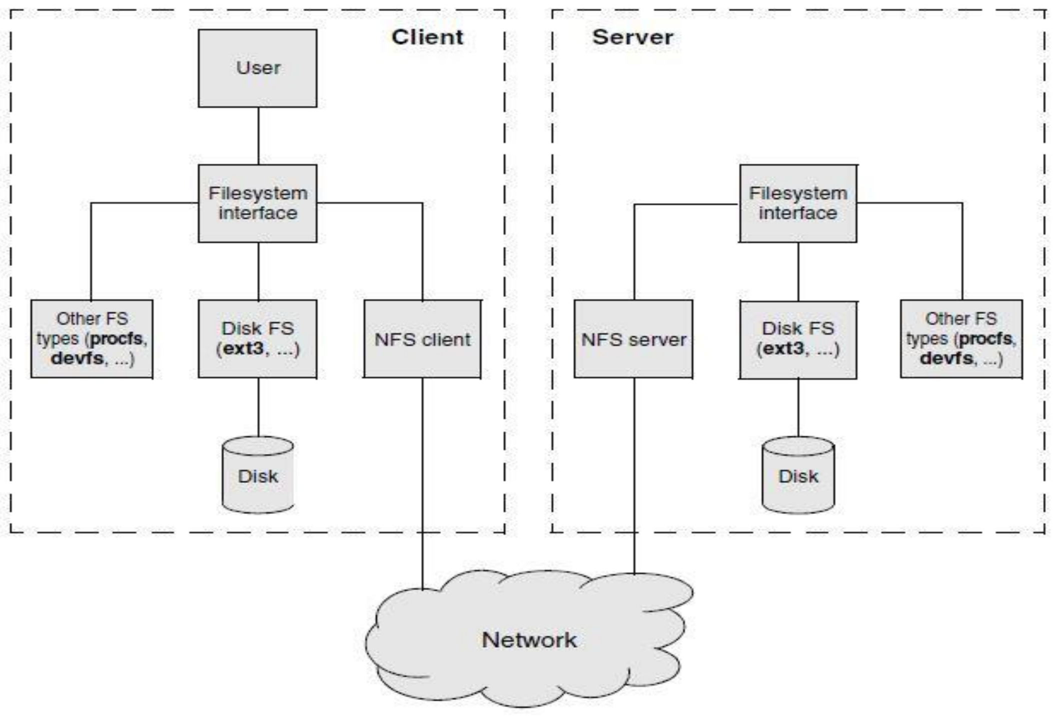
Introduction

NFS stands for Network Filesystem protocol. It is a UNIX de facto standard originally developed by Sun Microsystems. It allows a server to share selected local directory hierarchies with client systems on a heterogeneous network. NFS runs on UNIX, DOS, Windows, VMS, Linux, and more. Files on the remote computer (the fileserver) appear as if they are present on the local system (the client).The physical location of a file is irrelevant to an NFS user.

NFS reduces storage needs and system administration workload. As an example, each system in a company traditionally holds its own copy of an application program. To upgrade the program, the administrator needs to upgrade it on each system. NFS allows you to store a copy of a program on a single system and give other users access to it over the network. This scenario minimizes storage requirements by reducing the number of locations that need to maintain the same data. In addition to boosting efficiency, NFS gives users on the network access to the same data (not just application programs), thereby improving data consistency and reliability. By consolidating data, NFS reduces administrative overhead and provides a convenience to users. The flow of data from a client to server can be seen in the figure given below.

An NFS directory hierarchy appears to users and application programs as just another directory hierarchy. By looking at it, you cannot tell that a given directory holds a remotely mounted NFS directory hierarchy and not a local ext3 filesystem. The NFS server translates commands from the client into operations on the server’s filesystem.

1) **Diskless systems**: In many computer facilities, user files are stored on a central fileserver equipped with many large-capacity disk drives and devices that quickly and easily make backup copies of the data. A diskless system boots from a fileserver (netboots), a CD, or a floppy diskette and loads system software from a fileserver. The Linux Terminal Server Project (ltsp.org) Web site says it all: "Linux makes a great platform for deploying diskless workstations that boot from a network server. The LTSP is all about running thin client computers in a Linux environment. "Because a diskless workstation does not require a lot of computing power, you can give older, retired computers a second life by using them as diskless systems.



2) **Netboot/PXE:** You can netboot systems that are appropriately set up. Fedora/RHEL includes the PXE (Preboot Execution Environment) server package for netbooting Intel systems. Older systems sometimes use tftp (Trivial File Transfer Protocol) for netbooting. Non-Intel architectures have historically included netboot capabilities, which Fedora/RHEL also supports. You can build the Linux kernel so that it mounts root (/) using NFS. Given the many ways to set up a system, the one you choose depends on what you want to do. See the Remote-Boot mini-HOWTO for more information.
3) **Dataless systems:** Another type of Linux system is a dataless system, in which the client has a disk but stores no user data (only Linux and the applications are kept on the disk). Setting up this type of system is a matter of choosing which directory hierarchies are mounted remotely.

   df: shows where directory hierarchies are mounted: The df utility displays a list of the directory hierarchies available on the system, along with the amount of disk space, free and used, on each. The –h (human) option makes the output more intelligible. Directory hierarchy names that are prepended with hostname: are available through NFS.

4) **Errors:** Sometimes you may lose access to remote files. For example, a network problem or a remote system crash may make these files temporarily unavailable. When you try to access a remote file in these circumstances, you get an error message, such as NFS server speedy not responding. When the local system can contact the remote server again, you see another message, such as NFS server speedy OK. Setting up a stable network and server (or not using NFS) is the best defense against these kinds of problems.

5) **Security:** NFS is based on the trusted-host paradigm and therefore has all the security shortcomings that plague other services based on this paradigm. In addition, NFS is not encrypted. Because of these issues, you should implement NFS on a single LAN segment only, where you can be (reasonably) sure that systems on a LAN segment are what they claim to be. Make sure a firewall blocks NFS traffic from outside the LAN and never use NFS over the Internet.

14.1 Setting Up an NFS Client

The Prerequisites for setting up an NFS client is to install the following packages:

nfs-utils

system-config-nfs (optional)

Under RHEL, the portmap utility must be running to enable reliable file locking. Under FEDORA, this function is served by rpcbind.

JumpStart I: Mounting a Remote Directory Hierarchy

To set up an NFS client, mount the remote directory hierarchy the same way you mount a local directory hierarchy.

- **mount: Mounts a Remote Directory Hierarchy**

There are various assumption for mounting a remote directory hierarchy. These are: (1) Speedy is on the same network as the local system and is sharing /home and /export with the local system.

(2) The /export directory on speedy holds two directory hierarchies that you want to mount: /export/progs and /export/oracle.

(3) The example mounts speedy's /home directory on /speedy.home on the local system, /export/progs on /apps, and /export/oracle on /oracle.

First use mkdir to create the directories that are the mount points for the remote directory hierarchies:

- **# mkdir /speedy.home /apps /oracle**

You can mount any directory from an exported directory hierarchy. In this example, speedy exports /export and the local system mounts /export/progs and /export/oracle. The following commands manually mount the directory hierarchies one time:

```
# mount speedy:/home /speedy.home
# mount -o ro,nosuid speedy:/export/progs /apps
# mount -o ro speedy:/export/oracle /oracle
```

The first command mounts the /home directory hierarchy from speedy on the local directory /speedy.home. The second and third commands use the –o ro option to force a readonly mount. The second command adds the nosuid option, which forces setuid executables in the mounted directory hierarchy to run with regular permissions on the local system. If you receive the error mount: RPC: Program not registered, it may mean NFS is not running on the server. By default, directory hierarchies are mounted read-write, assuming the NFS server is exporting them with read-write permissions.

- **nosuid option:**

If a user could run a setuid program, that user has the power of Superuser. This ability should be limited. Unless you know that a user will need to run a program with setuid permissions from a mounted directory hierarchy, always mount a directory hierarchy with the nosuid option. For example, you would need to mount a directory hierarchy with setuid privileges when a diskless workstation has its root partition mounted using NFS.

- **nodev option**

Mounting a device file creates another potential security hole. Although the best policy is not to mount untrustworthy directory hierarchies, it is not always possible to implement this policy. Unless a user needs to use a device on a mounted directory hierarchy, mount directory hierarchies with the nodev option, which prevents character and block special files on the mounted directory hierarchy from being used as devices.

- **fstab file**

If you mount directory hierarchies frequently, you can add entries for the directory hierarchies to the **/etc/fstab file.**

```
$ cat /etc/fstab
...
speedy:/home              /speedy.home    nfs  -   0  0
speedy:/export/progs      /apps           nfsr,nosuid 0  0
speedy:/export/oracle     /oracle         nfs r    0   0
```

A file that is mounted using NFS is always type nfs on the local system, regardless of what type it is on the remote system. Typically, you do not run fsck on or back up an NFS directory hierarchy. The entries in the third, fifth, and sixth columns of fstab are usually nfs (filesystem type), 0 (do not back up this directory hierarchy with dump), and 0 (do not run fsck on this directory hierarchy).The options for mounting an NFS directory hierarchy differ from those for mounting an ext3 or other type of filesystem.

- **umount: Unmounts a Remote Directory Hierarchy**

Use umount to unmount a remote directory hierarchy the same way you would unmount a local filesystem

mount: Mounts a Directory Hierarchy

The mount utility associates a directory hierarchy with a mount point (a directory). You can use mount to mount an NFS (remote) directory hierarchy.

Attribute Caching

File attributes, which are stored in a file's inode, provide information about a file, such as file modification time, size, links, and owner. File attributes do not include the data stored in a file. Typically file attributes do not change very often for an ordinary file; they change even less often for a directory file. Even the size attribute does not change with every write instruction: When a client is writing to an NFS-mounted file, several write instructions may be given before the data is transferred to the server

ac (noac) (**attribute cache):** It permits attribute caching (default). The noac option disables attribute caching. Although noac slows the server, it avoids stale attributes when two NFS clients actively write to a common directory hierarchy.

acdirmax=n (attribute cache directory file maximum): The n is the number of seconds, at a maximum, that NFS waits before refreshing directory file attributes (default is 60 seconds).

acdirmin=n (attribute cache directory file minimum): The n is the number of seconds, at a minimum, that NFS waits before refreshing directory file attributes (default is 30 seconds).

acregmax=n (attribute cache regular file maximum): The n is the number of seconds, at a maximum, that NFS waits before refreshing regular file attributes (default is 60 seconds).

acregmin=n (attribute cache regular file minimum): The n is the number of seconds, at a minimum, that NFS waits before refreshing regular file attributes (default is 3 seconds).

actimeo=n (attribute cache timeout): This option sets acregmin, acregmax, acdirmin, and acdirmax to *n seconds* (without this option, each individual option takes on its assigned or default value).

Error Handling

Various options control what NFS does when the server does not respond or when an I/O error occurs. To allow for a mount point located on a mounted device, a missing mount point is treated as a timeout.

- **fg (bg) (foreground):** The option fg retries failed NFS mount attempts in the foreground (default). The bg (background) option retries failed NFS mount attempts in the background.
- **hard (soft) :** This option displays server not responding on the console on a major timeout and keeps retrying (default). The soft option reports an I/O error to the calling program on a major timeout. In general, it is not advisable to use soft. As the mount man page says of soft, "Usually it just causes lots of trouble."
- **nointr (intr) (no interrupt):** This option does not allow a signal to interrupt a file operation on ahard mounted directory hierarchy when a major timeout occurs (default). The intr option allows this type of interrupt.
- **retrans=n (retransmission value):** After n minor timeouts, NFS generates a major timeout (default is 3). A major timeout aborts the operation or displays server not responding on the console, depending on whether hard or soft is set.
- **retry=n (retry value):** The number of minutes that NFS retries a mount operation before giving up (default is 10,000).
- **timeo=n (timeout value):** The n is the number of tenths of a second that NFS waits before retransmitting following an RPC, or minor, timeout (default is 7). The value is increased at each timeout to a maximum of 60 seconds or until a major timeout occurs. On a busy network, in case of a slow server, or when the request passes through multiple routers/gateways, increasing this value may improve performance.

Miscellaneous Options

- **lock (nolock):** Permits NFS locking (default). The nolock option disables NFS locking (does not start the lockd daemon) and is useful with older servers that do not support NFS locking.
- **mounthost=name:** The name of the host running mountd, the NFS mount daemon.
- **mountport=n:** The port used by mountd.
- **nodev (no device):** Causes mounted device files not to function as devices**.**
- **port=n:** The port used to connect to the NFS server (defaults to 2049 if the NFS daemon is not registered with rpcbind/portmap). When n=0 (default), NFS queries rpcbind/portmap on the server to determine the port.
- **rsize=n (read block size)** The number of bytes read at one time from an NFS server. The default block size is 4096.
- **wsize=n (write block size):** The number of bytes written at one time to an NFS server. The default block size is 4096.

- **tcp:** Use TCP in place of the default UDP protocol for an NFS mount. This option may improve performance on a congested network; however, some NFS servers support UDP only.
- **udp:** Use the default UDP protocol for an NFS mount.

Improving Performance:

- **hard/soft:** Several parameters can affect the performance of NFS, especially over slow connections such as a line with a lot of traffic or one controlled by a modem. If you have a slow connection, make sure hard is set (this is the default) so that timeouts do not abort program execution.
- **Block size:** One of the easiest ways to improve NFS performance is to increase the block size—that is, the number of bytes NFS transfers at a time. The default of 4096 is low for a fast connection using modern hardware. Try increasing **rsize and wsize to** 8192 or higher. Experiment until you find the optimal block size. Unmount and mount the directory hierarchy each time you change an option.
- **Timeouts:** NFS waits the amount of time specified by the timeo option for a response to a transmission. If it does not receive a response in this amount of time, it sends another transmission. The second transmission uses bandwidth that, over a slow connection, may slow things down further. You may be able to increase performance by increasing **timeo.**

/etc/fstab: Mounts Directory Hierarchies Automatically

The **/etc/fstab**file lists directory hierarchies that the system mounts automatically as it comes up. This example line from fstab mounts grape's /gc1 filesystem on the /grape.gc1 mount point:

```
grape:/gc1          /grape.gc1          nfsrsize=8192,wsize=8192 0          0
```

A mount point should be an empty, local directory. Files in a mount point are hidden when a directory hierarchy is mounted on it.The type of a filesystem mounted using NFS is always nfs, regardless of its type on the local system. You can increase the rsize and wsize options to improve performance.This example from fstab mounts a filesystem from speedy:

```
speedy:/export    /speedy.export    nfstimeo=50,hard            0          0
```

Because the local system connects to speedy over a slow connection, timeo is increased to 5 seconds (50 tenths of a second). This example from fstab shows a remote-mounted home directory. Because speedy is a local server and is connected via a reliable, high-speed connection, timeo is decreased and rsize and wsize are increased substantially:

```
speedy:/export/home /home nfstimeo=4,rsize=16384,wsize=16384       0          0
```

14.2 Setting Up an NFS Server

The prerequisites for setting up NFS server is to install the following packages:

nfs-utils

system-config-nfs (optional)

After this run chkconfig to cause nfs to start when the system enters multiuser mode:

```
# /sbin/chkconfignfs on
```

The nfs can be started by:

```
# /sbin/service nfs start
```

The nfsinit script starts mountd, nfsd, and rquotad.RHEL Under RHEL, the portmap daemon must be running to enable reliable file locking.

JumpStart II: Configuring an NFS ServerUsing system-config-nfs:

To display the NFS Server Configuration window, enter the command system-config-nfs or select Main Menu: System | Administration | Server Settings | NFS. From this window you can generate an /etc/exports file, which is almost all there is to setting up an NFS server. The system-config-nfs utility allows you to specify which directory hierarchies are shared and how they are shared using NFS. Each exported hierarchy is called a share. To add a share, click Add on the toolbar. To modify a share, highlight the share and click Properties on the toolbar. Clicking Add displays the Add NFS Share window, while clicking Properties displays the Edit NFS Share window. The Add/Edit NFS Share window has three tabs: Basic, General Options, and User Access. These are given as:

- **Basic tab:** You can specify the pathname of the root of the shared directory hierarchy, the names or IP addresses of the systems (hosts) that the hierarchy will be shared with, and whether users from the specified systems will be able to write to the shared files. The selections in the other two tabs correspond to options that you can specify in the **/etc/exports** file.

- **General Options tab:** In this tab, make the following changes:

Allow connections from ports 1024 and higher: **insecure**

Allow insecure file locking: **no_auth_nlm or insecure_locks**

Disable subtree checking:**no_subtree_check**

Sync write operations on request: **sync**

Force sync of write operations immediately: **no_wdelay**

Hide filesystems beneath: **nohide**

Export only if mounted: **mountpoint**

- **User Access tab:** In this tab, make the following changes:

Treat remote root user as local root: **no_root_squash**

Treat all client users as anonymous users: **all_squash**

Local user ID for anonymous users: **anonuid**

Local group ID for anonymous users: **anongid**

After making the changes you want, click OK to close the Add/Edit NFS Share window and click OK again to close the NFS Server Configuration window. There is no need to restart any daemons.

Exporting a Directory Hierarchy

Exporting a directory hierarchy makes the directory hierarchy available for mounting by a client on the network. “Exported” does not mean “mounted”: When a directory hierarchy is exported, it is placed in the list of directory hierarchies that can be mounted by other systems. An exported directory hierarchy may be mounted (or not) at any given time. A server holds three lists of exported directory hierarchies:

- **/etc/exports—** Access control list for exported directory hierarchies. The system administrator can modify this file by editing it or by running system-config-nfs.

- **/var/lib/nfs/xtab—** Access control list for exported directory hierarchies. Initialized from /etc/exports when the system is brought up. Read by mountd when a client asks to mount a directory hierarchy. Modified by exportfs as directory hierarchies are mounted and unmounted by NFS.

- **Kernel's export table—** List of active exported directory hierarchies. The kernel obtains this information from /var/lib/nfs/xtab. You can display this table by giving the command cat /proc/fs/nfs/exports.The /etc/exports file is the access control list for exported directory hierarchies that NFS clients can mount; it is the only file you need to edit to set up an NFS server.The exports file controls the following aspects:

1) Which clients can access files on the server?
2) Which directory hierarchies on the server each client can access
3) How each client can access each directory hierarchy
4) How client usernames are mapped to server usernames
5) Various NFS parameters

Each line in the exports file has the following format:

export-point client1(options) [client2(options) ...]

where export-point is the absolute pathname of the root directory of the directory hierarchy to be exported, client1-n is the name of one or more clients or is one or more IP addresses, separated by SPACEs, that are allowed to mount the export-point. You can either use system-config-nfs to make changes to exports or you can edit this file directly. The following simple exports file gives grape read and write access and gives speedy readonly access to the files in /home:

```
# cat /etc/exports
/home grape(rw,no_subtree_check)
/home speedy(ro,no_subtree_check)
```

General options

- **auth_nlm (no_auth_nlm) or secure_locks (insecure_locks):** This causes the server to require authentication of lock requests (using the NLM [NFS Lock Manager] protocol). Use no_auth_nlm for older clients when you find that only files that anyone can read can be locked.
- **mountpoint[=path]:** It allows a directory to be exported only if it has been mounted. This option prevents a mount point that does not have a directory hierarchy mounted on it from being exported and prevents the underlying mount point from being exported.
- **nohide (hide):** When a server exports two directory hierarchies, one of which is mounted on the other, a client must mount both directory hierarchies explicitly to access both. When the second (child) directory hierarchy is not explicitly mounted, its mount point appears as an empty directory and the directory hierarchy is hidden. The nohide option causes the underlying second directory hierarchy to appear when it is not explicitly mounted, but this option does not work in all cases.
- **ro (rw) (readonly):** It permits only read requests on an NFS directory hierarchy. Use rw to permit read and write requests.
- **secure (insecure):** It requires that NFS requests originate on a privileged port so that a program without root permissions cannot mount a directory hierarchy. This option does not guarantee a secure connection.

- **subtree_check (no_subtree_check):** It checks subtrees for valid files. Assume that you have an exported directory hierarchy that has its root below the root of the filesystem that holds it (that is, an exported subdirectory of a filesystem). When the NFS server receives a request for a file in that directory hierarchy, it performs a subtree check to confirm the file is in the exported directory hierarchy.
- **sync (async) (synchronize):** It specifies that the server is to reply to requests only after disk changes made by the request are written to disk. The **async option** specifies that the server does not have to wait for information to be written to disk and can improve performance, albeit at the cost of possible data corruption if the server crashes or the connection is interrupted.
- **Wdelay (no_wdelay) (write delay):** Causes the server to delay committing write requests when it anticipates that another, related request follows, thereby improving performance by committing multiple write requests within a single operation. The **no_wdelay option** does not delay committing write requests and can improve performance when the server receives multiple, small, unrelated requests.

User ID Mapping Options

Each user has a UID number and a primary GID number on the local system. The local /etc/passwd and /etc/group files map these numbers to names. When a user makes a request of an NFS server, the server uses these numbers to identify the user on the remote system. It raises several issues: the user may not have the same ID numbers on both systems and may therefore have owner access to files of another user. You may not want the root user on the client system to have owner access to root-owned files on the server. You may not want a remote user to have owner access to some important system files that are not owned by root (such as those owned by bin).

Owner access means that the remote user can execute, remove, or—worse—modify the file. NFS gives you two ways to deal with these cases:

- You can use the root_squash option to map the ID number of the root user on a client to the nfsnobody user on the server.
- You can use the all-squash option to map all NFS users on the client to nfsnobody on the server

NIS and NFS: When you use NIS for user authorization, users automatically have the same UIDs on both systems. If you are using NFS on a large network, it is a good idea to use a directory service such as LDAP or NIS for authorization. Without such a service, you must synchronize the passwd files on all the systems manually.

root_squash (no_root_squash): It maps requests from root on a remote system so that they appear to come from the UID for nfsnobody, an unprivileged user on the local system, or as specified by a nonuid. It does not affect other sensitive UIDs such as bin. The no_root_squash option turns off this mapping so that requests from root appear to come from root.

no_all_squash (all_squash): It does not change the mapping of users making requests of the NFS server. The all_squash option maps requests from all users, not just root, on remote systems to appear to come from the UID for nfsnobody, an unprivileged user on the local system, or as specified by a nonuid. This option is useful for controlling access to exported public FTP, news, and other directories.

anonuid=un and anongid=gn: It sets the UID or the GID of the anonymous account to un or gn, respectively. NFS uses these accounts when it does not recognize an incoming UID or GID and when instructed to do so by root_squash or all_squash.

showmount:

It displays NFS status information. Without any options, the showmount utility displays a list of systems that are allowed to mount local directories.To display information for a remote system, give the name of the remote system as an argument. You typically use showmount to display a list of directory hierarchies that a server is exporting. The information that showmount provides may not be complete, however, because it depends on mountd and trusts that remote servers are reporting accurately. In the following example, bravo and grape can mount local directories, but you do not know which ones:

```
# /usr/sbin/showmount
Hosts on localhost:
bravo.tcorp.com
grape.tcorp.com
```

If showmount displays an error such as RPC: Program not registered, NFS is not running on the server. Start NFS on the server with the nfsinit script

–a (all): It tells which directories are mounted by which remote systems. This information is stored in /etc/exports.

```
# /usr/sbin/showmount -a
All mount points on localhost:
bravo.tcorp.com:/home
grape.tcorp.com:/home
```

–e (exports): It displays a list of exported directories.

```
# /usr/sbin/showmount -e
Export list for localhost:
/home bravo.tcorp.com,grape.tcorp.com
```

exportfs: Maintains the List of ExportedDirectory Hierarchies

The exportfs utility maintains the kernel's list of exported directory hierarchies. Without changing /etc/exports, exportfs can add to or remove from the list of exported directory hierarchies.Anexportfs command has the following format:

```
/usr/sbin/exportfs [options] [client:dir ...]
```

where options is one or more options (as detailed in the next section), client is the name of the system that dir is exported to, and dir is the absolute pathname of the directory at the root of the directory hierarchy being exported.The system executes the following command when it comes up (it is in the nfsinit script). This command reexports the entries in /etc/exports and removes invalid entries from /var/lib/nfs/xtab so that /var/lib/nfs/xtab is synchronized with /etc/exports:

```
# exportfs -r
```

Options: There are various options which we can use:

- **–a (all):** Exports directory hierarchies specified in /etc/exports. This option does not unexport entries you have removed from exports (that is, it does not remove invalid entries from /var/lib/nfs/xtab); use –r to perform this task.

- **–i (ignore):** Ignores /etc/exports; uses what is specified on the command line only.
- **–o (options):** Specifies options. You can specify options following –o the same way you do in the exports file. For example, exportfs –i –o ro speedy:/home/sam exports /home/sam on the local system to speedy for readonly access.
- **–r (reexport):** Reexports the entries in /etc/exports and removes invalid entries from /var/lib/nfs/xtab so that /var/lib/nfs/xtab is synchronized with /etc/exports.
- **–u (unexport)** Makes an exported directory hierarchy no longer exported. If a directory hierarchy is mounted when you unexport it, you will see the message Stale NFS file handle if you try to access the directory hierarchy from the remote system.
- **–v (verbose):** Provides more information. Displays export options when you use exportfs to display export information.

14.3 Testing the Server Setup

From the server, run the nfsinit script with an argument of status. If all is well, the system displays something like the following:

```
# /sbin/service nfs status
rpc.mountd (pid 15795) is running...
nfsd (pid 15813 15812 15811 15810 15809 15808 15807 15806) is running...
rpc.rquotad (pid 15784) is running...
```

Next, from the server, use rpcinfo to make sure NFS is registered with rpcbind/portmap:

```
$ /usr/sbin/rpcinfo -p localhost | grep nfs
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
```

Repeat the preceding command from the client, replacing localhost with the name of the server. The results should be the same. Finally, try mounting directory hierarchies from remote systems and verify access.

14.4 Automount: Automatically MountsDirectory Hierarchies

With distributed computing, when you log in on any system on the network, all of your files, including startup scripts, are available. In a distributed computing environment, all systems are commonly able to mount all directory hierarchies on all servers: Whichever system you log in on, your home directory is waiting for you.As an example, assume that /home/alex is a remote directory hierarchy that is mounted on demand. When you issue the command ls /home/alex, autofs goes to work: It looks in the /etc/auto.home map, finds that alex is a key that says to mount bravo:/export/home/alex, and mounts the remote directory hierarchy. Once the directory hierarchy is mounted, ls displays the list of files you want to see. If you give the command ls /home after this mounting sequence, ls shows that alex is present within the /home directory. The df utility shows that alex is mounted from bravo.

Prerequisites

The prerequisites is to install the following package: **autofs**

Run chkconfig to cause autofs to start when the system enters multiuser mode:

```
# /sbin/chkconfigautofs on
```

Start autofs:

```
# /sbin/service autofs start
```

An **autofs directory hierarchy** is like any other directory hierarchy, but remains unmounted until it is needed, at which time the system mounts it automatically (demand mounting). The system unmounts an autofs directory hierarchy when it is no longer needed—by default after five minutes of inactivity. Automatically mounted directory hierarchies are an important part of administrating a large collection of systems in a consistent way. The automount daemon is particularly useful when an installation includes many servers or many directory hierarchies. It also helps to remove server–server dependencies. When you boot a system that uses traditional fstab-based mounts and an NFS server is down, the system can take a long time to come up as it waits for the server to time out. Similarly, when you have two servers, each mounting directory hierarchies from the other, and both systems are down, both may hang as they are brought up and each tries to mount a directory hierarchy from the other. This situation is called a server–server dependency. The automount facility gets around these issues by mounting a directory hierarchy from another system only when a process tries to access it. When a process attempts to access one of the directories within an unmounted autofs directory hierarchy, the kernel notifies the automount daemon, which mounts the directory hierarchy. You must give a command, such as cd /home/alex, that accesses the autofs mount point (in this case /home/alex) to create the demand that causes automount to mount the autofs directory hierarchy so you can see it. Before you issue the cd command, alex does not appear to be in /home. The main file that controls the behavior of automount is /etc/auto.master. Example:

```
# cat /etc/auto.master
/free1    /etc/auto.misc    --timeout    60
/free2    /etc/auto.misc2   --timeout    60
```

The auto.master file has three columns.

- The first column names the parent of the autofs mount point—the location where the autofs directory hierarchy is to be mounted (/free1 and /free2 in the example are not mount points but will hold the mount points when the directory hierarchies are mounted).
- The second column names the files, called map files, that store supplemental configuration information.
- The optional third column holds mount options for map entries that do not specify an option.

Although the map files can have any names, one is traditionally named auto.misc. Following are the two map files specified in auto.master:

```
# cat /etc/auto.misc
sam -fstype=ext3 :/dev/sda8
# cat /etc/auto.misc2
helen -fstype=ext3 :/dev/sda9
```

Before the new setup can work, you must create directories for the parents of the mount points (/free1 and /free2 in the preceding example) and start (or restart) the automount daemon using the autofsinit script.The following command displays information about configured and active autofs mount points:

```
# /sbin/service autofs status
```

Summary:

- NFS runs on UNIX, DOS, Windows, VMS, Linux, and more. Files on the remote computer (the fileserver) appear as if they are present on the local system (the client). The physical location of a file is irrelevant to an NFS user.

- NFS is based on the trusted-host paradigm and therefore has all the security shortcomings that plague other services based on this paradigm.
- To display the NFS Server Configuration window, enter the command system-config-nfs or select Main Menu: System |Administration |Server Settings | NFS.
- A file that is mounted using NFS is always type nfs on the local system, regardless of what type it is on the remote system.
- The mount utility associates a directory hierarchy with a mount point (a directory). You can use mount to mount an NFS (remote) directory hierarchy.
- A server holds three lists of exported directory hierarchies. These are: /etc/exports, /var/lib/nfs/xtab and kernel's export table.

Keywords:

- **NFS:** NFS stands for Network Filesystem protocol. It is a UNIX de facto standard originally developed by Sun Microsystems. It allows a server to share selected local directory hierarchies with client systems on a heterogeneous network.
- **NFS Server:** The NFS server translates commands from the client into operations on the server's filesystem.
- **df utility**: The df utility displays a list of the directory hierarchies available on the system, along with the amount of disk space, free and used, on each.
- **portmap utility:** The portmap utilitymust be running to enable reliable file locking.
- **umount:** It unmounts a remote directory hierarchy.Use umount to unmount a remote directory hierarchy the same way you would unmount a local filesystem
- **Share:** The system-config-nfs utility allows you to specify which directory hierarchies are shared and how they are shared using NFS. Each exported hierarchy is called a share.
- **exportfs:** The exportfs utility maintains the kernel's list of exported directory hierarchies. Without changing /etc/exports, exportfs can add to or remove from the list of exported directory hierarchies

Self Assessment

1. The problem in NFS security is

A. NFS is encrypted
B. NFS is not encrypted
C. NFS does not respond
D. None of the above

2. Which of these options disables a signal to interrupt a file operation on hardmounted directory hierarchy?

A. intr
B. nointr
C. unintr
D. None of the above

3. Which of these options displays a list of exported directories?

A. -a
B. -e
C. -list
D. None of the above

4. In FEDORA, which utility is used for setting up an NFS client?

A. portmap
B. rpcbind
C. setupnfs
D. None of the above

5. By default block size in wsize and rsize is _____________

A. 128
B. 256
C. 1468
D. 4096

6. Which of these options are used to mount a remote directory hierarchy?

A. mount
B. automount
C. Both of the above
D. None of the above

7. Which utility displays a list of directory hierarchies available on the system?

A. df
B. dir
C. dirhier
D. None of the above

8. Which of these options disables the attribute caching?

A. ac
B. noac
C. unac
D. None of the above

9. Which of these options permits only read access on an NFS directory hierarchy?

A. r
B. ro
C. rw
D. or

10. NFS

A. Reduces the storage requirement

B. Boosts efficiency
C. Reduces administration workload
D. All of the above

11. If a user can run a setuid program, that user
A. Is a normal user
B. Has the power of a superuser
C. Hides the identity
D. None of the above
12. Kernel's export table consists of
A. Active exported directory hierarchies
B. Inactive exported directory hierarchies
C. Both active and inactive
D. None of the above

13. NFS stands for
A. Network FileSystem
B. Not a FileSystem
C. New Filesystem
D. None of the above

14. In RHEL, which utility is used for setting up an NFS client?
A. portmap
B. rpcbind
C. setupnfs
D. None of the above

15. The NFS performance can be improved by ___________
A. Increasing the block size
B. Decreasing the block size
C. Block size should remain constant
D. None of the above

Answers for Self Assessment

1.	B	2.	B	3.	B	4.	B	5.	D
6.	C	7.	A	8.	B	9.	B	10.	D
11.	B	12.	C	13.	A	14.	A	15.	A

Review Questions:

1. What is NFS? Explain the flow of data from client to server with the help of a diagram.
2. Explain the various features of NFS.
3. How to set up an NFS client? Explain.
4. Explain the utility which is used to mount a directory hierarchy with various options available with it.
5. What is error handling? Also explain various options available with it.
6. How to set up an NFS server?
7. How to export a directory hierarchy? Explain the lists associated with this.
8. How to test the server setup? Explain autofs.

Further Readings

Mark G Sobell, A Practical Guide to Fedora and RedHat Enterprise Linux, Fifth Edition, Prentice Hall

Web Links

https://cloud.netapp.com/blog/azure-anf-blg-linux-nfs-server-how-to-set-up-server-and-client

LOVELY PROFESSIONAL UNIVERSITY
Jalandhar-Delhi G.T. Road (NH-1)
Phagwara, Punjab (India)-144411
For Enquiry: +91-1824-300360
Fax.: +91-1824-506111
Email: odl@lpu.co.in

ISBN 978-93-94068-25-4
9 789394 068254