GCA (S5) 01
Exam. Code: LSA

Linux System Administration

SEMESTER V

BACHELOR OF COMPUTER APPLICATION

Block 1



KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Anjana Kakati Mahanta, Gauhati University

Prof. (Retd.) Pranhari Talukdar, Gauhati University

Dr. Jyotiprokash Goswami, Associate Professor, Assam Engineering College

Course Co-ordinators

Dr. Tapashi Kashyap Das, Assistant Professor, KKHSOU

Ms. Sruti Sruba Bharali, Assistant Professor, KKHSOU

SLM Preparation Team

UNITS	CONTRIBUTORS
1	Ms. Ishita Chakravorty, Royal Global University
2, 3, 7 & 8	Ms. Sangeeta Borkakoty, USTM
4 & 6	Mr. Bhupesh Talukdar, Dimoria College
5	Mr. Dwipen Laskar, Gauhati University

Editorial Team

Content : Mr. Gautam Chakrabarty, NERIM

Language : Prof. (Retd.) Robin Goswami, Cotton College

Structure, Format & Graphics: Ms. Sruti Sruba Bharali, KKHSOU

June, 2019 ISBN: 978-93-89123-74-6

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State University is made available under a Creative Commons Attribution-Non Commercial-ShareAlike4.0 License (International): http.//creativecommons.org/licenses/by-nc-sa/4.0.

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters : Patgaon, Rani Gate, Guwahati-781 017

City Office : Housefed Complex, Dispur, Guwahati-781 006; Web: www.kkhsou.in

The university acknowledges with thanks the financial support provided by the **Distance Education Bureau, UGC** for the preparation of this study material.

BACHELOR OF COMPUTER APPLICATION

Linux System Administration

Block-1

CONTENTS

		Pages
UNIT 1:	**Introduction to System Adminstration** Introduction to System Administration, Role and Power of System Administrator, Qualities of Good System Administrator	**7-17**
UNIT 2:	**Introduction to LINUX Operating System** Basic Features of the Linux Operating System. A brief overview of the most Popular Linux Distributions (Red Hat Enterprise Linux (RHEL), Ubuntu, Debian, Fedora, SUSE)	**18-32**
UNIT 3:	**Installation of LINUX Operating System** Installation Requirements, Partitioning the Hard Drive in Linux, Installing the Linux System, Installing and Configuring Software in Linux	**33-50**
UNIT 4:	**LINUX Kernel** Linux Kernel and Device Drivers, System Startup and Shutdown, Standard I/O, Standard Error, Redirection and Piping	**51-68**
UNIT 5:	**Basics of LINUX File System** Basics of Linux File System– File System Types (ext3, ext4, xfs, jfs, ReiserFS, iso9660 etc.), Three Basic Types of Files (Ordinary or Regular, Special or Device and Directory), I-nodes and File Attributes	**69-85**
UNIT 6:	**File Organization** Absolute and Relative Path Names, File System Mounting and Unmounting, Organization of the File Tree, Standard Directories and their Contents	**86-95**
UNIT 7:	**LINUX File Handling Commands** Files and Directory Handling Commands– ls, cd, cp, mv, rm, mkdir, rmdir, Commands for Creating and Viewing Ordinary Files– cat, more, pg	**96-110**
UNIT 8:	**LINUX Filter Commands** Filter Commands– wc, head, tail, cut, tr, grep (with Regular Expressions), Setting User and Group Ownership of Files and Access Permissions– chmod, chown, chgrp Commands	**111-128**

COURSE INTRODUCTION

Just like Windows, and Mac OS, Linux is an operating system. It's been around since the mid '90s, and has since reached a user-base that spans industries and continents. Linux is actually everywhere. It's in our phones, in our cars, in our refrigerators. It runs most of the Internet, the supercomputers making scientific breakthroughs, and the world's stock exchanges. Linux is one of the platforms to run desktops, servers, and embedded systems across the globe, and is one of the most reliable, secure, and worry-free operating systems available.Linux was originally developed for personal computers based on the Intel x86 architecture, but has since been ported to more platforms than any other operating system. Linux is the leading operating system on servers and other big systems such as mainframe computers, and is the only OS used on TOP500 supercomputers.

The course is divided into two blocks:

Block 1 introduces the Linux operating system. The installation and description of different Linux OS along with file organizations are covered in this block. The commands for file handing and filter commands along with user and group permissions are also covered in this block.

Block 2 starts with shell programming basics. The block also covers topics like process, printing jobs, different typesof Linux users etc. Topics like networking in Linux and different network protocols like NFS, NIS etc are also described in this block. Another topic of discussion in this block is the security aspects of Linux OS.

BLOCK INTRODUCTION

This is the first block of the course ***Linux System Administration***. After completing this block, learners will be able to understand the various parts of Linux operating environment. Learners will also be able to use Linux commands to perform basic functions in any Linux OS.

This block comprises the following eight units :

Unit 1 gives us an introduction to system administration. The role and qualities of a good system administrator are mentioned in this unit.

Unit 2 introduces us to Linux operating system. The basic features of a Linux operating sytem along with an introduction to different Linux OS like Ubuntu, Fedora etc are discussed in this unit.

Unit 3 describes the installation process of Linux OS. The different steps like partitioning the hard drive, installing and configuring the software in Linux are discussed in this unit.

Unit 4 introduces us to the Linux kernel. System startup, shutdown, standard I/O and error are some of the topics dicussed in this unit.

Unit 5 deals with Linux File system. Different types of files like regular, special etc are discussed in this unit. Different system file types like ext3, ext4 etc are also covered in this unit.

Unit 6 deals with file organization in Linux. Topics like absolute and relative path names, mounting and unmounting, directories and their contents are described in this unit.

Unit 7 gives us an introduction to file handling command of Linux. Different file handling commands like ls, cd etc along with commands for creating and viewing files are described in this unit.

Unit 8 gives us an introduction to filter commands of Linux. Different filter commands like wc, grep etc along with user and group permissions are described in this unit.

Each unit of this block includes some along-side boxes to help you know some of the difficult, unseen terms. Some “EXERCISES” have been included to help you apply your own thoughts. You may find some boxes marked with: “LET US KNOW”. These boxes will provide you with some additional interesting and relevant information. Again, you will get “CHECK YOUR PROGRESS” questions. These have been designed for you to self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with “ANSWERS TO CHECK YOUR PROGRESS” given at the end of each unit.

UNIT 1: INTRODUCTION TO SYSTEM ADMINISTRATION

UNIT STRUCTURE

1.1 Learning Objectives

1.2 Introduction

1.3 Role and Power of System Administrator

1.4 Qualities of Good System Administrator

1.5 Let Us Sum Up

1.6 Further Reading

1.7 Answers to Check Your Progress

1.8 Model Questions

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe the position of the system administrator
- explain the usage of **su** command
- describe the roles and privileges of an administrator
- describe the skills required for being a system administrator
- describe the UNIX architecture that includes the kernel and shell.

1.2 INTRODUCTION

System administration in a multiuser system plays a vital role, mainly when a system is accessed by multiple users. It can be handled by a trusted person who is responsible for the administration of the whole system. The basic task of a system administrator is to ascertain that there is little to complain about the system's performance or service availability.

The system administrator operates in superuser mode also which is called *root*. This superuser has the privilege to issue any command, access any file, and perform every function. Root is the special user-id to log on to the system. The use of root command must be limited due to the following reasons:

- Inexperienced users can cause severe damage.
- The use of root command for non-privileged tasks is unnecessary and can be open to attack.
- Since root can look at anyone's files, it violates security and privacy of the system.

The root command can be used as mentioned in figure 1.1.

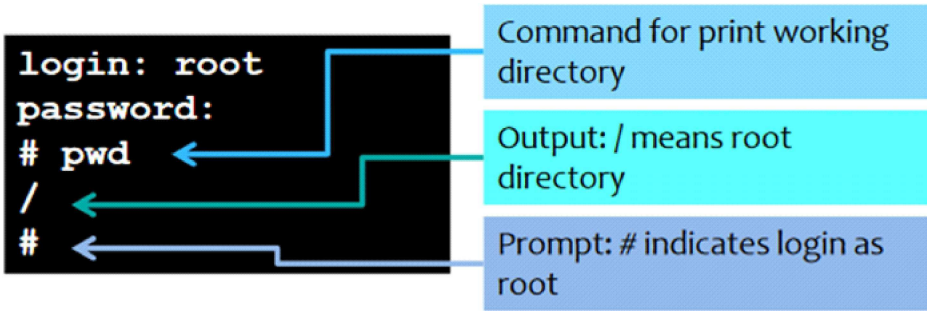


Figure 1.1: Use of Root Command

The tasks of the superuser include starting up and shutting down a system, opening accounts for new users and giving them proper working set-up. Administration tasks also involve installation of new software, distributing user disk space, taking regular back-ups, keeping system logs, ensuring secure operations and providing network services and web access.

su command stands for changing user ID or for becoming superuser. It switches from non-privileged account to the super user account. The **whoami** command is used to view the login name.

The **su** command can be used as described in figure 1.2 and 1.3.

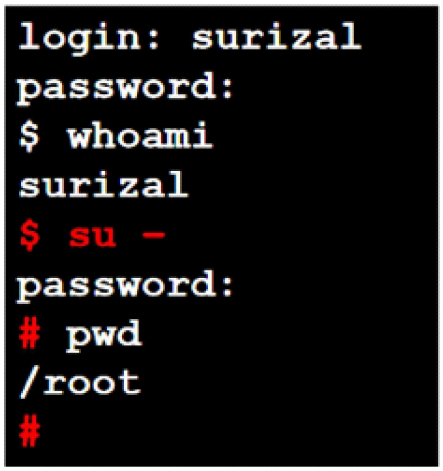


Figure 1.2: Use of su Command Argument

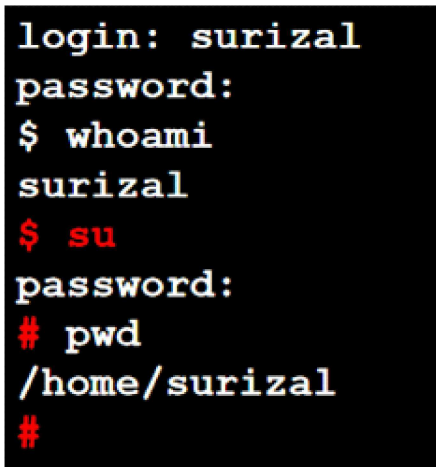


Figure 1.3: Use of su Command without Argument

The optional argument '–' is used to provide an environment similar to what the user had logged in directly. The **su** command without '-' argument executes command as root but retains users current environment.

1.3 ROLE AND POWER OF THE SYSTEM ADMINISTRATOR

The Linux system administrator is responsible for installing the server of the system, updating configuration, operating and maintaining the hardware and software of the system. He/She has to manage failures in the system, troubleshoot and maintain the servers, the user accounts, security, etc. He/She is also responsible for managing the disk space as well as the backup. The system administrator is responsible for changing the contents of the file or its attributes like its ownership or permission and can delete any file even if it is write-protected. This professional also needs to reset new password and constantly monitor the special services and security of the servers. They are expected to remain informed with the recent knowledge in Linux system in order to assess the hardware and software technologies. The key responsibilities that need to be taken care of by a Linux system administrator are:

a) To be totally involved in designing, implementing, securing and maintaining the computer systems for its test, development and production environment.
b) To recognize and evaluate the issues which hamper the performance of the system, and to work with the product development team in close coordination in order to come up with solutions for the issues.
c) To sort out all the supporting tasks that are counted in, but not restricted to code deployment, managing source control systems, virtual servers, scripting, etc.
d) To consider and identify the discrepancies in the system, to evaluate the risks, and to implement solutions by obeying to the security standards.
e) To be involved in writing and modifying scripts for application deployments as well as system monitoring.

f) To be involved in developing and supervising the backup, replication, clustering and fail over strategies.

The Linux system administrators must participate in various cross training activities, so as to attain a comfort level that supports the database. They might also be involved in training the subordinate team members in system administration and software deployment responsibilities. They retain proper documentation of all the procedures, configurations, programs, etc. They offer valuable support to the unsatisfied users by answering their technical queries. They also arrange for the much needed customer support through e-mail, telephone, knowledge based programs and other resources. They also hold security audits on a routine basis as well as plan and increase the current offerings related to services.

The system administrator has the power to modify the contents or the attributes of any files owned by other users, initiate or kill any process that are essentially required for running the system, change the password of any user without even entering the old one, setting the system clock, control user's access to scheduling service and networking service.



CHECK YOUR PROGRESS

Q.1: What is a LINUX system?

.....

.....

.....

.....

Q.2: What is a process?

.....

.....

.....

.....

Q.3: Who is a system administrator?

.....

.....

Q.4: Why is ***who am i*** command used?

.....

.....

Q.5: What are the the powers of a super user?

.....

.....

.....

.....

.....

1.4 QUALITIES OF GOOD SYSTEM ADMINISTRATOR

To become a successful Linux system administrator, one should be a self-starter who is highly motivated, detail oriented, flexible and ready to follow the instructions given by their superior. One should be ready to learn new things and apply the up-to-date technologies that would help the organization to grow. The individual in this position should possess excellent interpersonal skills and should be comfortable working in a team environment. He/She should be able to work independently and maintain excellent customer service skills. They should also be able to identify the issues related to system and solve them in an effective way. He/she must be able to work well under pressure and able to respond and prioritize work according to the stringent deadlines.

A good system administrator should have the following skills:

1) Problem-solving skill:
   a) On a call when a computer system malfunctions.
   b) Quickly and correctly diagnose the crisis and produce the best solution.
2) Analysing skill:
   a) Understand the behaviour of software.
   b) Deploy.
   c) Troubleshoot Problems.
3) Programming skill:
   a) Understand different types of programming languages.
   b) Scripting or automation of routine tasks.

4) Soft-skills:

a) Users should feel warm and know that their problems will be fixed in no time.

b) Good interaction among colleagues/team members etc.

Based on the nature of work, the system administrators can be classified as shown in figure 1.4.

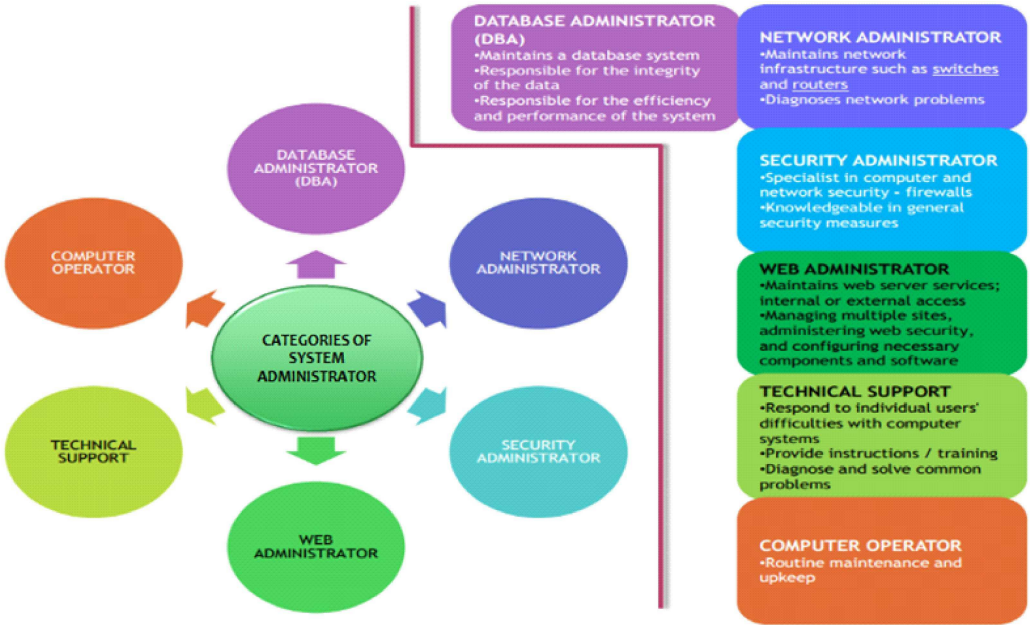


Figure 1.4: Categories of System Administrator



CHECK YOUR PROGRESS

Q.6: What are system administrator responsible for?

.....

.....

.....

.....

.....

.....

.....

.....

.....

Q.7: What skills are required for a good system administrator?

.....

.....

.....

.....

.....

.....

Q.8: What is a Linux Loader?

.....

.....

.....

.....

.....

Q.9: What is pwd?

.....

.....

.....

.....

Q.10: How to print your current working directory?

.....



1.5 LET US SUM UP

- System administrator is the person who is responsible for setting up and maintaining the system or server.
- The root account has full (unrestricted) access, so he/she can do anything with the system.
- The system administrator is responsible for user administration, maintaining system, verify whether peripherals are working properly or not, quickly arrange repair for hardware on the occasion of hardware failure, monitor system performance, create file systems, install software, create a backup and recovery policy.

- The system administrator also monitor network communication, updates system as soon as new version of OS and application software comes out, implements the policies for the use of the computer system and network setup security policies for users.
- A system admin must have a strong grasp of computer security, e.g. firewalls and intrusion detection systems.
- A system administrator should have problem solving skill, analyzing skill, programming skill, soft skills.



1.6 FURTHER READING

1) Das, S. (2012). *Your UNIX/*Linux: *The Ultimate Guide*. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
2) Hahn, H. (2008). *Harley Hahn's Guide to Unix and Linux*. California: McGraw-Hill Higher Education.



1.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: Linux is an operating system or a kernel. It is distributed under an open source license.The main advantage of Linux was that programmers were able to use the Linux Kernel to design their own custom operating systems. It is the backbone of popular operating systems like Debian, Knoppix, Ubuntu, and Fedora.

Ans. to Q. No. 2: A process is an instance of a program running in a computer. In UNIX and some other operating systems, a process is started when a program is initiated (either by a user entering a shell command or by another program).

Ans. to Q. No. 3: System administrator is the person who is responsible for setting up and maintaining the system or server. System administrators may be members of an information technology department.

Ans. to Q. No. 4: The ***who am i*** command is used to view the username of the current user. It is basically the concatenation of the strings "who", "am", "i".

Ans. to Q. No. 5: The root account has full (unrestricted) access, so he/she can do anything with system. For example, root can remove critical system files. In addition it can modify the contents or the attributes of any files owned by other users, initiate or kill any process that are essentially required for running the system, change the password of any user without even entering the old one, setting the system clock etc.

Ans. to Q. No. 6: The system administrator is responsible for the following things:

a) User administration (setup and maintaining account)
b) Maintaining system
c) Verify that peripherals are working properly
d) Quickly arrange repair for hardware in occasion of hardware failure
e) Monitor system performance
f) Create file systems
g) Install software
h) Create a backup and recovery policy
i) Monitor network communication
j) Update system as soon as new version of OS and application software comes out
k) Implement the policies for the use of the computer system and network
l) Set up security policies for users. A system administrator must have a strong grasp of computer security (e.g. firewalls and intrusion detection systems)
m) Documentation in form of internal wiki
n) Password and identity management.

Ans. to Q. No. 7: A good system administrator should have the following skills:

- Problem-Solving Skill:
  a) On a call when a computer system malfunctions.
  b) Quickly and correctly diagnose the crisis and produce the best solution.
- Analysing Skill:
  a) Understand the behaviour of software.
  b) Deploy.
  c) Troubleshoot Problems.
- Programming Skill:
  a) Understand different types of programming languages.
  b) Scripting or automation of routine tasks.
- Soft-Skills:
  a) Users should feel warm and know that their problems will be fixed in no time.
  b) Good interaction among colleagues/team members etc.

Ans. to Q. No. 8: Linux Loader is used to load Linux into memory. It can boot operating systems from floppy disks, hard disks, and it does not depend on a specific file system. Lilo handles some tasks such as to locate the kernel, identify other supporting programs, load memory and to start the kernel. The configuration file of lilo is located at "/etc/lilo.conf". Lilo reads this configuration file and it tells Lilo where it should place the bootloader.

Ans. to Q. No. 9: 'pwd' stands for 'Print Working Directory'. As the name states, command 'pwd' prints the current working directory or simply the directory user id, at present. It prints the current directory name with the complete path starting from root (/). This command is built in shell command and is available on most of the shell – bash, Bourne shell, ksh,zsh, etc.

Ans. to Q. No. 10: /bin/pwd



1.8 MODEL QUESTIONS

Q.1: An employee is working from home and has trouble joining a video call. How would you help him/her?

Q.2: How do you determine a server capacity? Why and how often do you need to upgrade a server? What is the necessity for encrypting the message?

Q.3: Explain the responsibility of a system administrator in detail.

Q.4: Explain the skills required for a good administrator.

Q.5: What is the sudo command used for?

*** ***** ***

UNIT 2: INTRODUCTION TO LINUX OPERATING SYSTEM

UNIT STRUCTURE

2.1 Learning Objectives

2.2 Introduction

2.3 The Linux Operating System

2.4 Linux Architecture

- 2.4.1 Components of Linux System
- 2.4.2 Architecture of Linux System

2.5 Basic Features of the Linux Operating System

2.6 Popular Linux Distributions

- 2.6.1 Red Hat Enterprise Linux
- 2.6.2 Ubuntu
- 2.6.3 Debain
- 2.6.4 Fedora
- 2.6.5 SUSE

2.7 Let Us Sum Up

2.8 Further Reading

2.9 Answers to Check Your Progress

2.10 Model Questions

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define an operating system
- describe the Linux Operating System
- explain the internal architecture of Linux
- list the basic features of the Linux Operating System
- give a brief overview of various Linux distributions like Fedora, Suse, etc.

2.2 INTRODUCTION

In the previous unit, we have learnt about the role and power of a system administrator. In this unit, we will discuss about LINUX operating system.

An operating system is an important part of a computer system. A computer system can be viewed as being built from three general components: the hardware, the operating system and the applications. The hardware includes a central processing unit (CPU), a keyboard, a hard drive, a mouse, a printer, etc. Applications are those which you use to operate the computer. They use the rest of the system to perform the desired task. The operating system is the component that on one side manages and controls the hardware and on the other, manages the applications. In the next unit, we will discuss about the installation process of LINUX operating system.

2.3 THE LINUX OPERATING SYSTEM

Linux operating system is one of the popular versions of the UNIX operating system, which is designed to offer a free or low cost operating system for personal computer users. It gained the reputation as a fast performing and very efficient system. This is a remarkably complete operating system, including a GUI (graphical user interface), TCP/IP, the Emacs editor, X Window System, etc. Linux is a family of free and open-source software operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991 by Linus Torvalds. Linux is typically packaged in a Linux distribution (or distro for short).

Distributions include the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project. Many Linux distributions use the word “Linux” in their name, but the Free Software Foundation uses the name GNU/Linux to emphasize the importance of GNU software.

Popular Linux distributions include Debian, Fedora, and Ubuntu. Commercial distributions include Red Hat Enterprise Linux and SUSE Linux Enterprise Server. Desktop Linux distributions include a windowing system

such as X11 or Wayland, and a desktop environment such as GNOME or KDE Plasma. Distributions intended for servers may omit graphics altogether, and include a solution stack such as LAMP. Because Linux is freely redistributable and anyone may create a distribution for any purpose.

Linux was originally developed for personal computers based on the Intel x 86 architecture, but has since been ported to more platforms than any other operating system. Linux is the leading operating system on servers and other big iron systems such as mainframe computers and is the only OS used on TOP 500 supercomputer (since November 2017, having gradually eliminated all competitors). It is used by around 2.3 percent of desktop computers.

Linux also runs on embedded systems, i.e. devices whose operating system is typically built into the firmware and is highly tailored to the system. This includes routers, automation controls, televisions, digital video recorders, video game consoles, and smart watches. Many smartphones and tablet computers run Android and other Linux derivatives. Because of the dominance of Android on smart phones, Linux has the largest installed base of all general-purpose operating systems.

Linux is one of the most prominent examples of free and open-source software collaboration. The source code may be used, modified and distributed-commercially or non-commercially by anyone under the terms of its respective licenses, such as the GNU General Public License.

2.4 THE LINUX ARCHITECTURE

2.4.1 Components of Linux System

Linux Operating System has primarily the following three components as shown in figure 2.1:

1) **Kernel:** Kernel is the core part of Linux. It is responsible for all major activities of Linux operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

2) **System Library:** System libraries are special functions or programs using which application programs or system utilities access Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.
3) **System Utility:** System Utility programs are responsible to do specialized, individual level tasks.

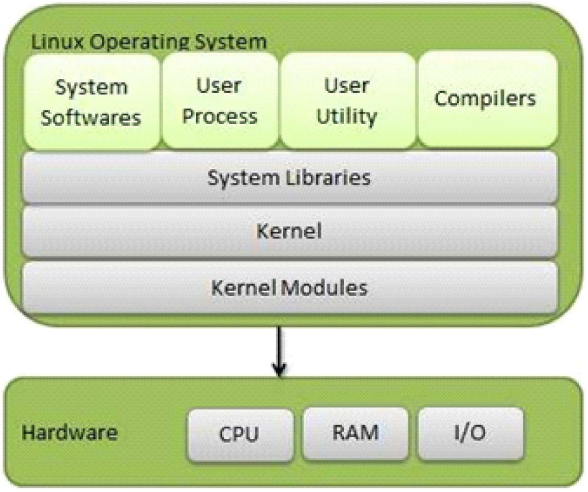


Figure 2.1: Components of Linux OS

Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

2.4.2 Architecture of a Linux System

The following illustration shows the architecture of a Linux system in figure 2.2.

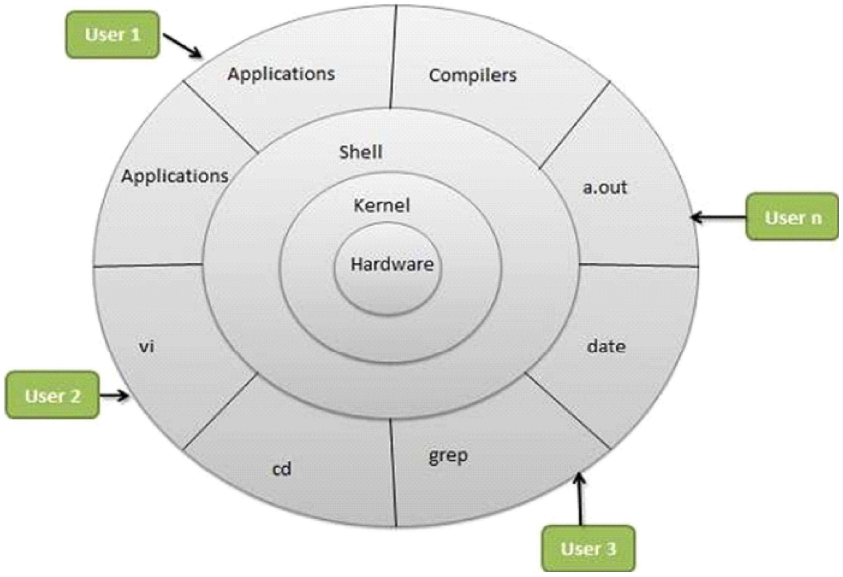


Figure 2.2: Architecture of a Linux System

The architecture of a Linux System consists of the following layers–

- **Hardware Layer:** Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel:** It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell:** An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities:** Utility programs that provide the user most of the functionalities of an operating system.

2.5 BASIC FEATURES OF THE LINUX OPERATING SYSTEM

Following are some of the important features of Linux Operating System.

- **Portable:** Portability means softwares can work on different types of hardwares in the same way. Linux kernel and application programs support their installation on any kind of hardware platform.

- **Open Source:** Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User:** Linux is a multiuser system which means multiple users can access system resources like memory / RAM / application programs at the same time.
- **Multiprogramming:** Linux is a multiprogramming system which means multiple applications can run at the same time.
- **Hierarchical File System:** Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell:** Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations such as call application programing etc.
- **Security:** Linux provides user security by using authentication features like password protection/ controlled access to specific files/ encryption of data.

Linux is fast, free and easy to use, power laptops and servers around the world. Linux has many more features such as:

- **Live CD/USB:** Almost all Linux distributions have Live CD/USB feature by which user can run/try the OS even without installing it on the system.
- **Graphical user interface (X Window System):** People think that Linux is a command line OS; it is somewhat true also but not necessarily, Linux has packages which can be installed to make the whole OS graphic based as Windows.
- **Supports most national or customized keyboards:** Linux is used worldwide and hence it is available in multiple languages, and it supports most of their custom national keyboards.
- **Application Support:** Linux has its own software repository from where users can download and install thousands of applications just by issuing a command in Linux Terminal or Shell. Linux can also run Windows applications if needed.



CHECK YOUR PROGRESS

Q.1: Linux operating system is a version of the Unix System. (True/False)

.....

Q.2: What is kernel?

.....

Q.3: Shell is an interface to the .....

.....

Q.4: The Linux system is a single-user or multi-user system?

.....

2.6 POPULAR LINUX DISTRIBUTIONS

A Linux distribution (often abbreviated as distro) is an operating system made from a software collection, which is based upon the Linux kernel and it is often, a package management system. Linux users usually obtain their operating system by downloading one of the Linux distributions, which are available for a wide variety of systems ranging from embedded devices (for example, OpenWrt) and personal computers (for example, Linux Mint) to powerful supercomputers (for example, Rocks Cluster Distribution).

A typical Linux distribution comprises a Linux kernel, GNU tools and libraries, additional software, documentation, a window system (the most common being the X Window System), a window manager, and a desktop environment. Most of the included software is free and open-source software made available both as compiled binaries and in source code form, allowing modifications to the original software. Usually, Linux distributions optionally include some proprietary software that may not be available in source code form, such as binary blobs required for some device drivers. A Linux distribution may also be described as a particular assortment of application and utility software (various GNU tools and libraries, for example), packaged together with the Linux kernel in such a way that its capabilities meet the needs of many users.The software is usually adapted to the distribution

and then packaged into software packages by the distribution's maintainers. The software packages are available online in so-called repositories, which are storage locations usually distributed around the world.

Besides glue components, such as the distribution installers (for example, Debian-Installer and Anaconda) or the package management systems, there are only very few packages that are originally written from the ground up by the maintainers of a Linux distribution.

2.6.1 Red Hat Enterprise Linux

Red Hat Enterprise Linux is a Linux distribution developed by Red Hat and targeted toward the commercial market. Red Hat Enterprise Linux is released in server versions for x86-64, Power Architecture, ARM64, and IBM Z, and a desktop version for x86-64. All of Red Hat's official support and training, together with the Red Hat Certification Program, focuses on the Red Hat Enterprise Linux platform. Red Hat Enterprise Linux is often abbreviated to RHEL.

The first version of Red Hat Enterprise Linux to bear the name originally came into the market as "Red Hat Linux Advanced Server". In 2003, Red Hat rebranded Red Hat Linux Advanced Server to "Red Hat Enterprise Linux AS", and added two more variants, Red Hat Enterprise Linux ES and Red Hat Enterprise Linux WS.

Red Hat uses strict trademark rules to restrict free re-distribution of their officially supported versions of Red Hat Enterprise Linux, and yet freely provides its source code. Third-party derivatives can be built and redistributed by stripping away the non-free components like Red Hat's trademarks. Examples include community-supported distributions like CentOS and Scientific Linux, and commercial forks like Oracle Linux.

To maintain a stable application binary interface (ABI), Red Hat does not update the kernel version, but instead backports new features to the same kernel version with which a particular version of RHEL has been released. New features are backported throughout the Production 1 phase of the RHEL lifecycle. Consequently, RHEL

may use a Linux kernel with a dated version number, yet the kernel is up-to-date regarding not only security fixes but also certain features.

2.6.2 Ubuntu

Ubuntu is a free and open-source Linux distribution based on Debian. Ubuntu is officially released in three editions: Desktop, Server, and Core. Ubuntu is a popular operating system for cloud computing, with support for OpenStack.

Ubuntu is developed by Canonical who provide security updates and support for each Ubuntu release, starting from the release date and until the release reaches its designated end-of-life (EOL) date. Ubuntu is named after the African philosophy of ubuntu, which Canonical translates as “humanity to others” or “I am what I am because of who we all are”.

A default installation of Ubuntu contains a wide range of software that includes LibreOffice, Firefox, Thunderbird, Transmission, and several lightweight games such as Sudoku and chess. Many additional software packages are accessible from the built in Ubuntu Software as well as any other APT-based package management tools.

Many additional software packages that are no longer installed by default, such as Evolution, GIMP, Pidgin, and Synaptic, are still accessible in the repositories still installable by the main tool or by any other APT-based package management tool. Cross-distribution snap packages and flatpaks are also available, that both allow installing software, such as some of Microsoft’s software, in most of the major Linux operating systems (such as any currently supported Ubuntu version and in Fedora). The default file manager is GNOME Files, formerly called Nautilus.

Ubuntu operates under the GNU General Public License (GPL) and all of the application software installed by default is free software. In addition, Ubuntu installs some hardware drivers that

are available only in binary format, but such packages are clearly marked in the restricted component.

Ubuntu aims to be secure by default. User programs run with low privileges and cannot corrupt the operating system or other users' files. For increased security, the sudo tool is used to assign temporary privileges for performing administrative tasks, which allows the root account to remain locked and helps prevent inexperienced users from inadvertently making catastrophic system changes or opening security holes. Polkit is also being widely implemented into the desktop.

Most network ports are closed by default to prevent hacking. A built-in firewall allows end-users who install network servers to control access. Ubuntu also supports full disk encryption as well as encryption of the home and private directories.

2.6.3 Debain

Debian is a Unix-like operating system consisting entirely of free software. Ian Murdock started the Debian Project on August 16, 1993. Debian 0.01 was released on September 15, 1993, and the first stable version, 1.1, was released on June 17, 1996. The Debian stable branch is the most popular edition for personal computers and network servers, and is used as the basis for many other distributions.

Debian is one of the earliest operating systems based on the Linux kernel. The project's work is carried out over the Internet by a team of volunteers guided by the Debian Project Leader and three foundational documents: the Debian Social Contract, the Debian Constitution, and the Debian Free Software Guidelines. New distributions are updated continually, and the next candidate is released after a time-based freeze.

Debian has been openly developed and freely distributed according to the principles of the GNU Project founded by Richard Stallman. This philosophy drew the support of the Free Software

Foundation, which sponsored the project from November 1994 to November 1995. When the sponsorship ended, the Debian Project formed Software in the Public Interest to continue financially supporting development.

Debian has access to online repositories that contain over 51,000 packages making it the largest collection of software in the world. Debian officially contains only free software, but non-free software can be downloaded and installed from the Debian repositories. Debian includes popular free programs such as LibreOffice, Firefox web browser, Evolution mail, K3b disc burner, VLC media player, GIMP image editor, and Evince document viewer. Debian is a popular choice for servers.

2.6.4 Fedora

Fedora is a Linux distribution developed by the community-supported Fedora Project and sponsored by Red Hat. Fedora contains software distributed under various free and open-source licenses and aims to be on the leading edge of such technologies. Fedora is the upstream source of the commercial Red Hat Enterprise Linux distribution.

Since the release of Fedora 21, *three* different editions are currently available: *Workstation*, focused on the personal computer, *Server* for servers, and *Atomic* focused on cloud computing.

Fedora has a reputation for focusing on innovation, integrating new technologies early on and working closely with upstreamLinux communities. Making changes upstream instead of specifically for Fedora ensures that the changes are available to all Linux distributions.

The default desktop environment in Fedora is GNOME and the default user interface is the GNOME Shell. Other desktop environments, including KDE Plasma, Xfce, LXDE, MATE and Cinnamon, are available and can be installed.

Fedora uses Security-Enhanced Linux by default, which implements a variety of security policies, including mandatory access

controls, which Fedora adopted early on. Fedora provides hardening wrapper, and does hardening for all of its packages by using compiler features such as position-independent executable (PIE).

Fedora comes preinstalled with a wide range of software such as LibreOffice and Firefox. Additional software is available from the software repositories and can be installed using the DNF package manager or GNOME Software.

Additionally, extra repositories can be added to the system, so that software not available in Fedora can be installed easily.

2.6.5 SUSE

SUSE Linux Enterprise Server is an operating system and it is built on top of the free and open source Linux kernel. It is distributed with system and application software from other open source projects. SUSE Linux is of German origin, basically an acronym of “Software und System-Entwicklung” (software and systems development), and was mainly developed in Europe. The first version appeared in early 1994, making SUSE one of the oldest existing commercial distributions. It is known for its YaST configuration tool.

SUSE Linux is available under two brands, openSUSE and SUSE Linux Enterprise. openSUSE is a free, community distribution driven by the openSUSE Project. It includes some of the latest Linux technologies and is designed for home users and enthusiasts. SUSE Linux Enterprise is Suse’s tested and certified open-source solution for major enterprises.

openSUSE is a freely available, community project that releases versions on a comparatively frequent basis, and generally uses the latest versions of the various open source projects that it includes.

SUSE Linux Enterprise is SUSE’s commercial edition, which SUSE releases much less frequently, enabling it to offer support more effectively for enterprise and production deployments. It is certified for a wide variety of enterprise applications and offers a

number of special enterprise features including, High Availability and Point of Sale extensions. SUSE historically uses a heavily-tested subset of packages from openSUSE Linux as the basis for SUSE Linux Enterprise.



CHECK YOUR PROGRESS

Q.5: On which Linux architecture is Ubuntu based on?

.....

Q.6: Which edition of Fedora is focused on cloud computing?

.....

Q.7: What are the three major Ubuntu editions?

.....

Q.8: Which is the default desktop environment in Fedora?

.....



2.7 LET US SUM UP

- An operating system is an important part of a computer system. A computer system can be viewed as being built from three general components: the hardware, the operating system and the applications.
- The operating system is the component that on one side manages and controls the hardware and on the other, manages the applications.
- Linux operating system is one of the popular versions of the UNIX operating system, which is designed to offer a free or low cost operating system for personal computer users.
- Linux is a family of free and open-source software operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991 by Linus Torvalds.Linux is typically packaged in a Linux distribution (or distro for short).
- Linux Operating System has primarily three components: kernel, system library and system utilities.

- The architecture of Linux Operating System has the following layers– hardware, kernel, shell, and utilities.
- Some basic features of Linux are: portability, multi-user, multiprogramming, hierarchical file system, shell and security.
- Linux also provides many additional features like live CD/USB, graphical user interface, support most national or customized keyboards and application support.
- A typical Linux distribution comprises a Linux kernel, GNU tools and libraries, additional software, documentation, a window system (the most common being the X Window System), a window manager, and a desktop environment.
- Some popular Linux distributions are - Red Hat, Ubuntu, Fedora, SUSE and Debain.



2.8 FURTHER READING

1) Das, S. (2012). *Your UNIX/*Linux: *The Ultimate Guide*. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



2.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: TRUE

Ans. to Q. No. 2: It is a core component of Linux

Ans. to Q. No. 3: kernel

Ans. to Q. No. 4: multi-user

Ans. to Q. No. 5: Debain

Ans. to Q. No. 6: Atomic

Ans. to Q. No. 7: Desktop, Server and Core

Ans. to Q. No. 8: GNOME



2.10 MODEL QUESTIONS

Q.1: What is Linux? What are the core components of Linux?

Q.2: What are the different layers of Linux architecture?

Q.3: What are the basic features of the Linux Operating System?

Q.4: What is a Linux distribution? What are some of the popular Linux distributions?

Q.5: What is Ubuntu? What are the features of Ubuntu?

Q.6: What is Fedora? What are the features of Fedora?

*** ***** ***

UNIT 3: INSTALLATION OF LINUX OPERATING SYSTEM

UNIT STRUCTURE

3.1 Learning Objectives

3.2 Introduction

3.3 Installation Requirements,

3.4 Partitioning the Hard Drive in Linux

- 3.4.1 Partitions
- 3.4.2 Extended and Logical Partitions
- 3.4.3 Partitioning the Hard Drive in Linux

3.5 Installing the Linux System

3.6 Installing and Configuring Software in Linux

- 3.6.1 Debian, Ubuntu: apt
- 3.6.2 Fedora, Red Hat: yum

3.7 Let Us Sum Up

3.8 Further Reading

3.9 Answers to Check Your Progress

3.10 Model Questions

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe the hardware and software requirements for installing Linux
- describe partitions
- partition the hard drive
- install Linux system on a computer
- install software in Linux Operating System.

3.2 INTRODUCTION

In the previous unit, we have learnt about the Linux O.S. In this unit, we will learn how to install a Linux O.S. The Linux open source operating system, or Linux OS, is a freely distributable, cross-platform operating system based on Unix that can be installed on PCs, laptops, netbooks, mobile and tablet devices, video game consoles, servers, supercomputers and more.

Before installing Linux Operating System on a computer, certain hardware requirements have to be met. Once this is done, the hard drive needs to be partitioned to install the operating system. In the next unit, we will explore the Linux kernal.

3.3 INSTALLATION REQUIREMENTS

Both Red Hat Enterprise Linux versions on both 32-bit and 64-bit platforms have the same system requirements as listed below.

Table 3.1: System Requirements for Red Hat Enterprise Linux Versions

Criteria	Requirements
Operating System	Red Hat Enterprise Linux 4 or 5 with the latest patches and upgrades
CPU Type	Pentium 4 or higher; 2 GHz or higher
Memory/RAM	1 GB minimum
Hard Disk	4 GB minimum

3.4 PARTITIONING THE HARD DRIVE IN LINUX

3.4.1 Partitions

A hard disk can be divided into several partitions. Each partition functions as if it were a separate hard disk. The idea is that if you have one hard disk, and want to have two operating systems on it, you can divide the disk into two partitions. Each operating system uses its partition as it wishes and doesn't touch the other ones. In this way the two operating systems can co-exist on the same hard disk.

The information about how a hard disk has been partitioned is stored in its first sector (that is, the first sector of the first track on the first disk surface). The first sector is the master boot record (MBR) of the disk; this is the sector that the BIOS reads in and starts when the machine is first booted. The master boot record contains a small program that reads the partition table, and checks which partition is active (that is, marked bootable), and reads the

first sector of that partition, the partition's boot sector (the MBR is also a boot sector, but it has a special status and therefore a special name). This boot sector contains another small program that reads the first part of the operating system stored on that partition (assuming it is bootable), and then it starts it.

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

3.4.2 Extended and Logical Partitions

The original partitioning scheme for PC hard disks allowed only four partitions. This quickly turned out to be too little in real life, partly because some people want more than four operating systems (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD, or Windows/NT, to name a few), but primarily because sometimes it is a good idea to have several partitions for one operating system. For example, swap space is usually best put in its own partition for Linux instead of in the main Linux partition for reasons of speed.

To overcome this design problem, extended partitions were invented. This trick allows partitioning a primary partition into sub-partitions. The primary partition thus subdivided is the extended partition; the sub-partitions are logical partitions. They behave like primary partitions but are created differently. There is no speed difference between them. By using an extended partition you can now have up to 15 partitions per disk.

The partition structure of a hard disk might look like that in Figure 3.1. The disk is divided into three primary partitions, the second of which is divided into two logical partitions. Part of the disk is not partitioned at all. The disk as a whole and each primary partition has a boot sector.

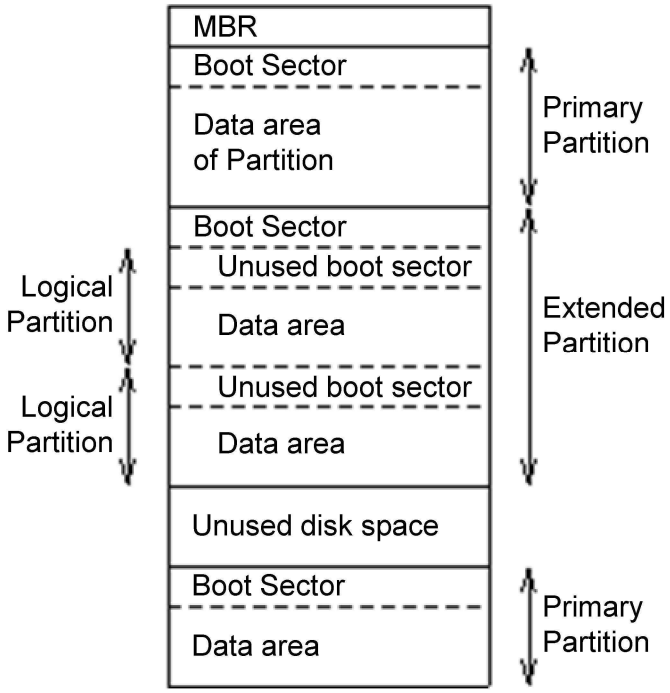


Figure 3.1: A Sample Hard Disk Partitioning

3.4.3 Partitioning the Hard Drive in Linux

For a proper Linux installation in a single-user desktop system, three partitions are recommended: *swap, root,* and *home*.

Swap: This is a partition that will be used as Linux swap space. This is the space on the hard drive that can be used as *virtual memory*. Virtual memory allows the computer to run large programs and perform complex tasks even if it does not have enough physical RAM to do the job.

There are no fixed guidelines for the necessary amount of swap space that is required. To keep it simple, create one swap partition that is 50% to 100% the size of the physical RAM in your system.

root (/): The *root* file system is represented by a forward slash (/). It is the top of the directory tree and contains Linux and everything that you install with Linux. You *must* create a partition for the root directory.

The size of your root partition will vary depending on what you install or plan to install. Check your distribution's documentation and reserve enough space for a *maximum* installation, plus *at least* 100MB more for temporary space and installation of new software. If you plan to download and try out lots of software, leave more space. If you have a small hard drive, you can trim back on your installed packages to save space.

In general, you should be fine with a root partition between 2GB and 8GB.

/home: The third and final partition you should create will hold your /home directory. This is the place where all the user-specific files are stored. On a multi-user system, each user will have their own directory under /home.

Strictly speaking, it is not necessary to create a separate partition for /home. If you do not, it will reside on the root partition like everything else. If you are cramped for space, you may need to configure your machine this way.



CHECK YOUR PROGRESS

Q.1: What is the minimum CPU speed required for Red Hat Enterprise Linux installation?

.....

Q.2: What is master boot record (MBR)?

.....

.....

Q.3: What are the partitions recommended for a proper Linux installation in a single-user desktop system?

.....

Q.4: What is the top of the directory tree in Linux?

.....

3.5 INSTALLING THE LINUX SYSTEM

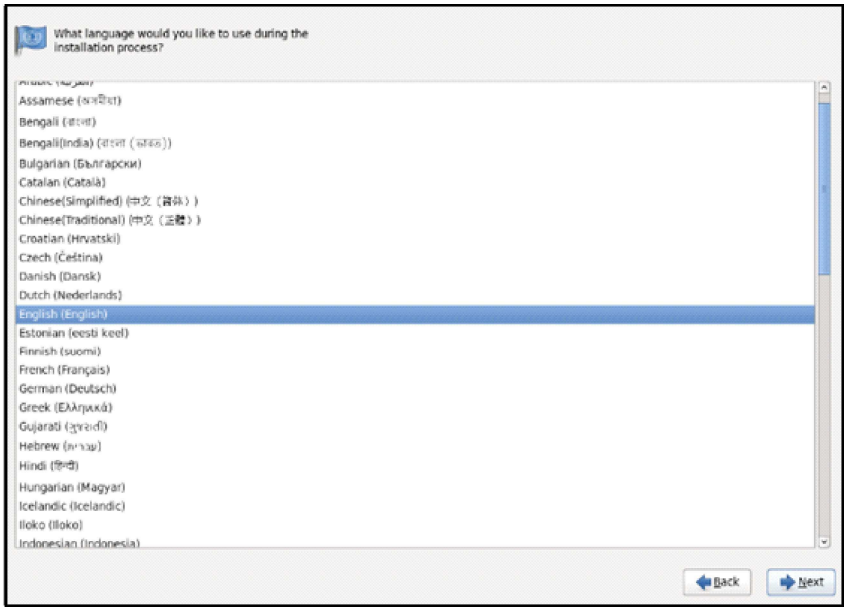
The steps to install Red Hat Enterprise Linux are as below:

Step 1: Boot Computer Using Red Hat 6 Installation CD/DVD.

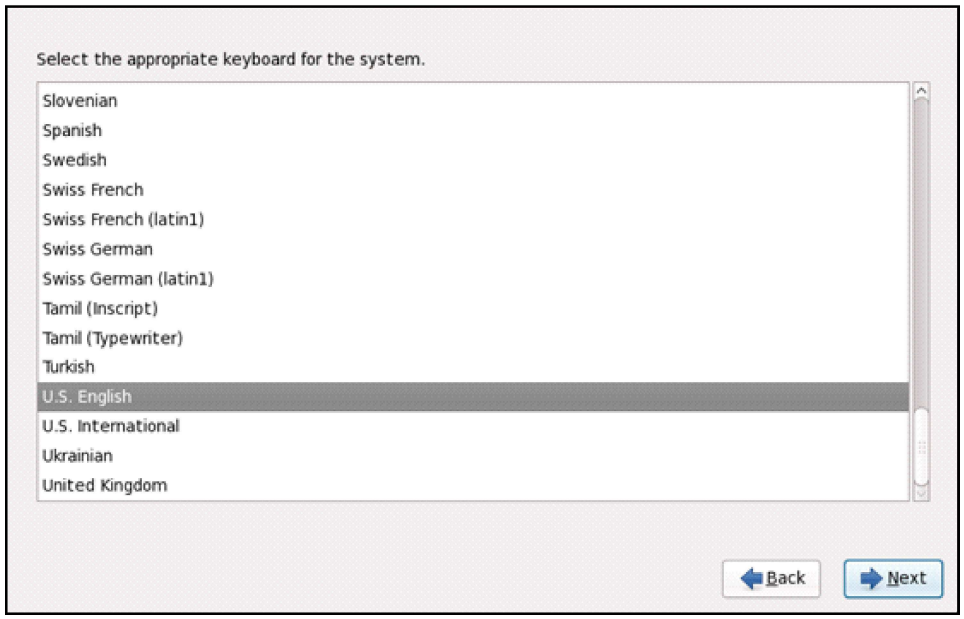
Select “Install or upgrade an existing system”



Step 2: Select Language



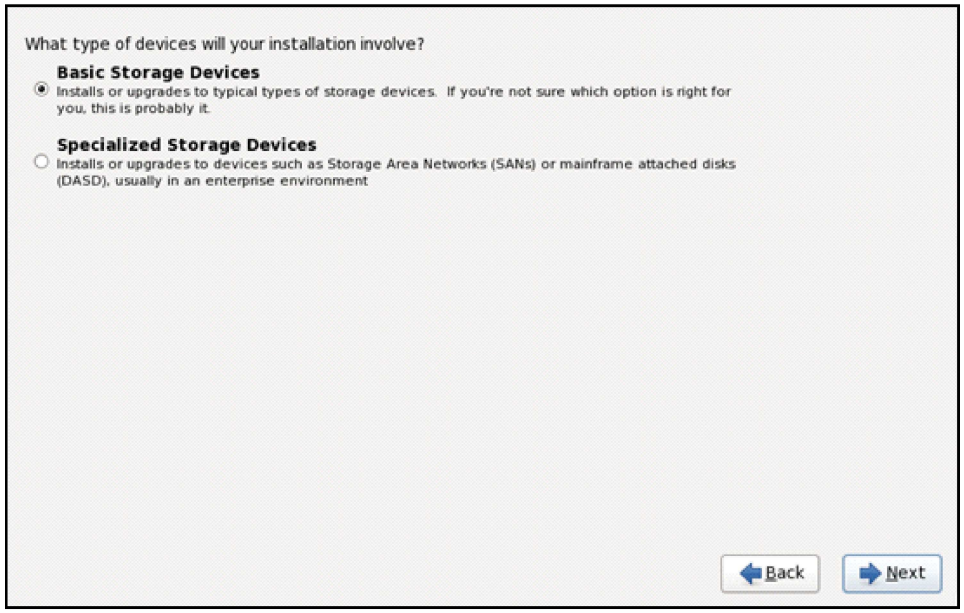
Step 3: Select the appropriate keyboard layout.



Note: To change your keyboard layout type after you have completed the installation, use the Keyboard Configuration Tool.

Type the system-config-keyboard command in a shell prompt to launch the Keyboard Configuration Tool. If you are not root, it prompts you for the root password to continue.

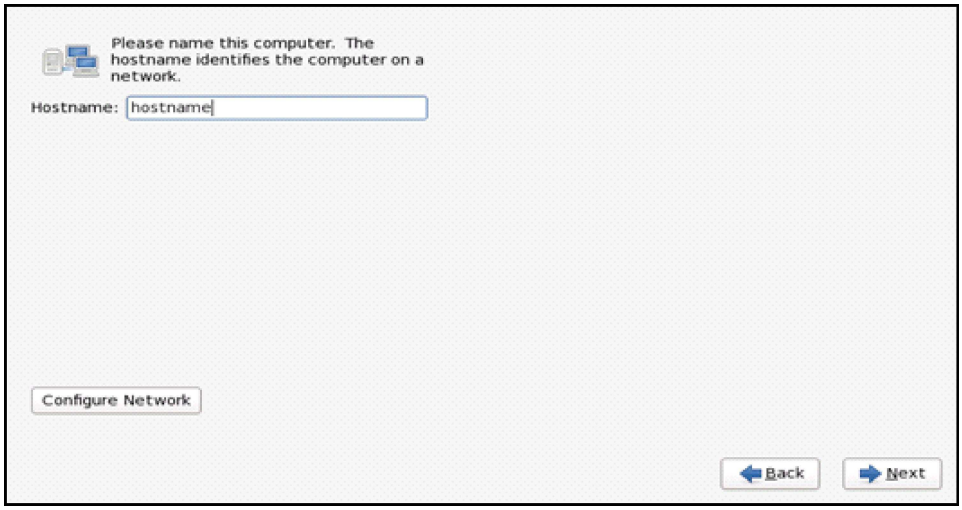
Step 4: Select storage devices



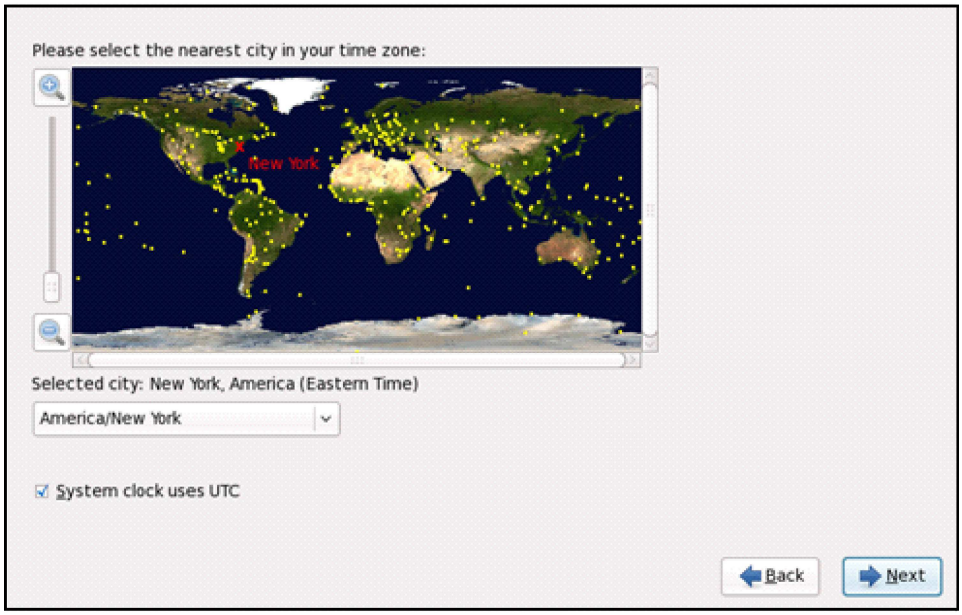
You can install Red Hat Enterprise Linux on a large variety of storage devices - either basic or specialized storage devices. Basic Storage

Devices are hard drives or solid-state drives connected directly to the local system. Specialized Storage Devices include Storage area networks (SANs), Direct access storage devices (DASDs), Firmware RAID devices and Multipath devices.

Step 5: Setting the hostname

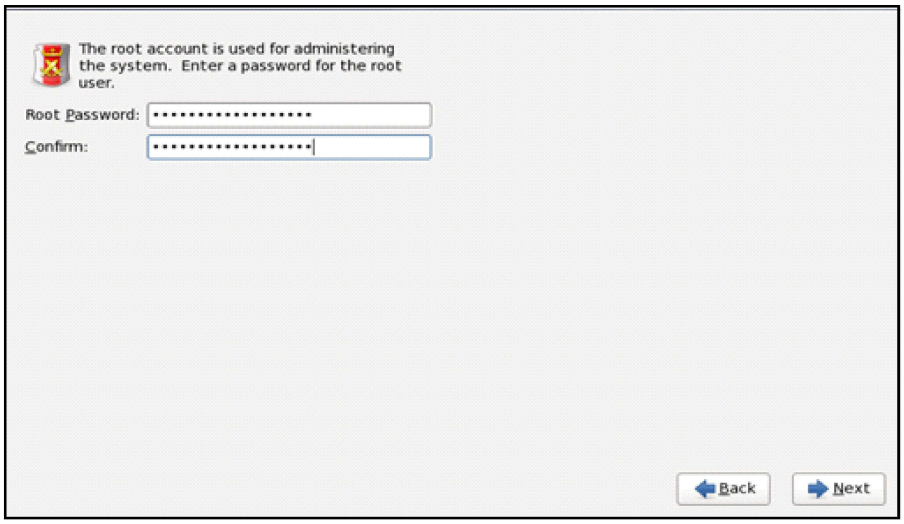


Step 6: Configuring the Time Zone



Step 7: Set root password

Setting up a root account and password is one of the most important steps during your installation. The root account is used to install packages, upgrade RPMs, and perform most system maintenance. Logging in as root gives you complete control over your system.

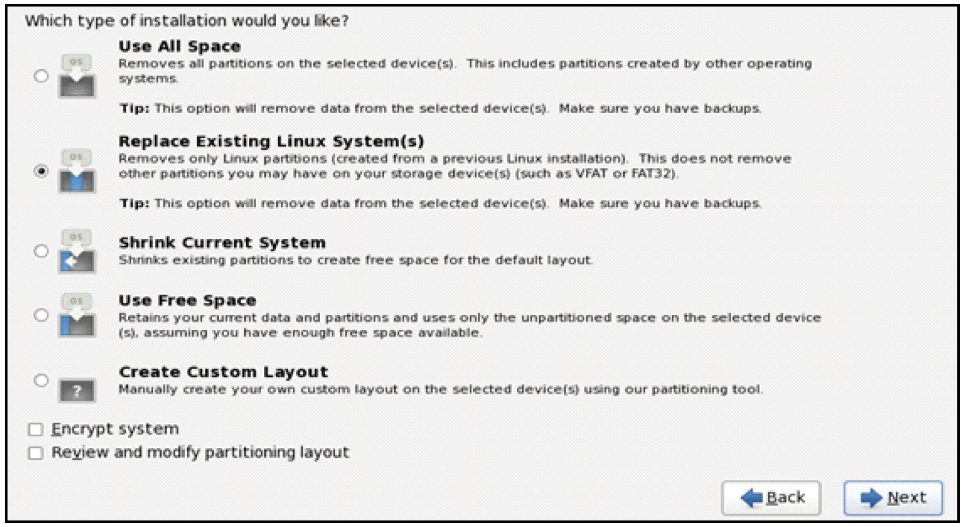


Step 8: Disk Partitioning Setup

Partitioning allows you to divide your hard drive into isolated sections, where each section behaves as its own hard drive. Partitioning is particularly useful if you run multiple operating systems.

On this screen you can choose to create the default partition layout in one of four different ways, or choose to partition storage devices manually to create a custom layout.

The first four options allow you to perform an automated installation without having to partition your storage devices yourself. If you do not feel comfortable with partitioning your system, choose one of these options and let the installation program partition the storage devices for you. Depending on the option that you choose, you can still control what data (if any) is removed from the system.



Your options are:

Use All Space: Select this option to remove all partitions on your hard drives (this includes partitions created by other operating systems such as Windows VFAT or NTFS partitions).

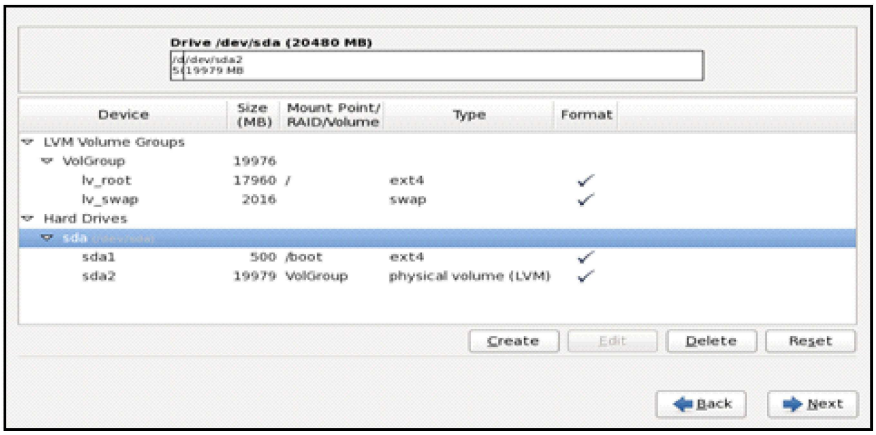
Replace Existing Linux System(s): Select this option to remove only partitions created by a previous Linux installation. This does not remove other partitions you may have on your hard drives (such as VFAT or FAT32 partitions).

Shrink Current System: Select this option to resize your current data and partitions manually and install a default Red Hat Enterprise Linux layout in the space that is freed.

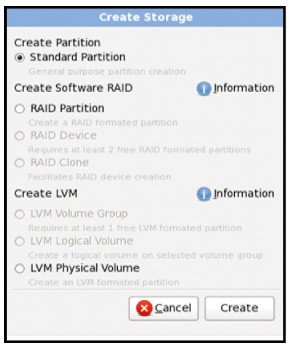
Use Free Space: Select this option to retain your current data and partitions and install Red Hat Enterprise Linux in the unused space available on the storage drives. Ensure that there is sufficient space available on the storage drives before you select this option.

Create Custom Layout: Select this option to partition storage devices manually and create customized layouts.

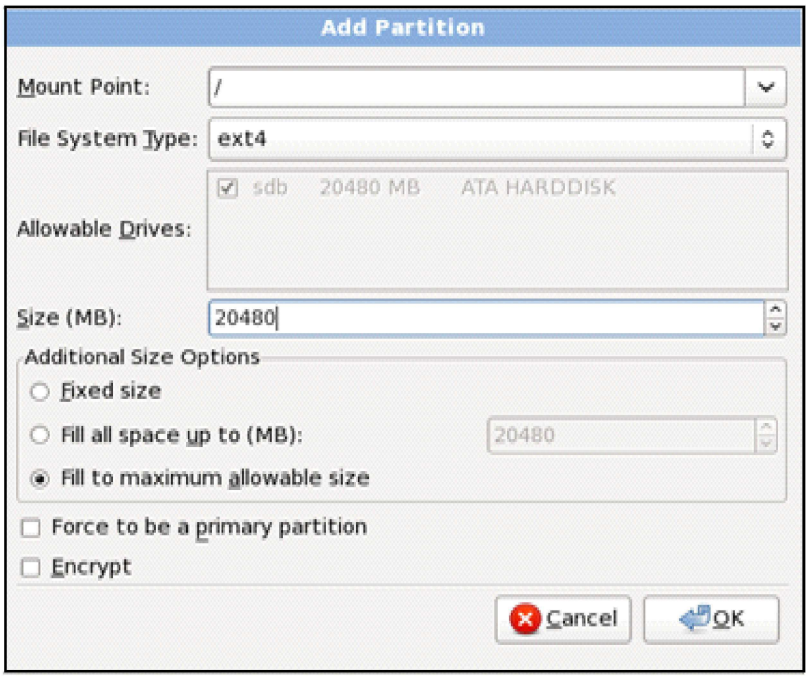
Step 9: Choose RHEL 6 Filesystem type



Step 10: Manually configuration of LVM and RAID storage

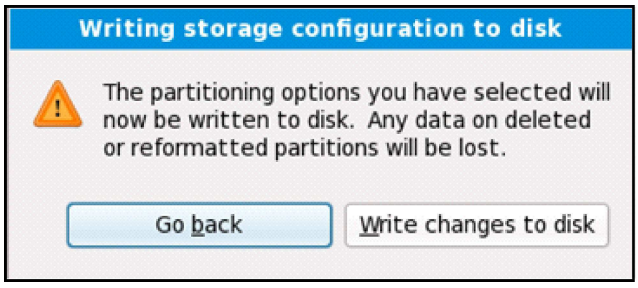


Step 11: Adding Partitions



Step 12: Write Changes to Disk

The installer prompts you to confirm the partitioning options that you selected. Click Write changes to disk to allow the installer to partition your hard drive and install Red Hat Enterprise Linux.



Step 13: Package Group Selection

By default, the Red Hat Enterprise Linux installation process loads a selection of software that is suitable for a system deployed as a basic server. Note that this installation does not include a graphical environment. To include a selection of software suitable for other roles, click the radio button that corresponds to one of the following options:

Basic Server: This option provides a basic installation of Red Hat Enterprise Linux for use on a server.

Database Server: This option provides the MySQL and PostgreSQL databases.

Web Server: This option provides the Apache web server.

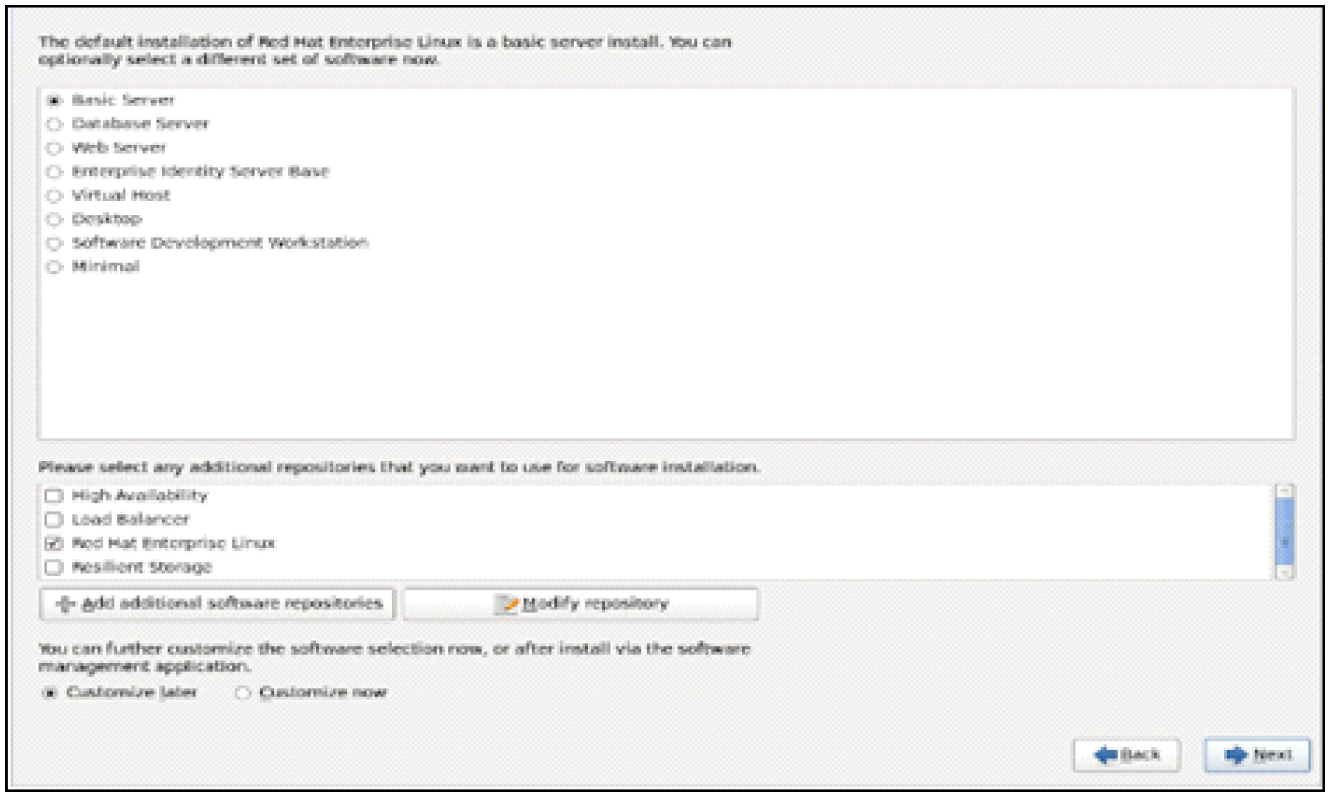
Enterprise Identity Server Base: This option provides OpenLDAP and Enterprise Identity Management (IPA) to create an identity and authentication server.

Virtual Host: This option provides the KVM and Virtual Machine Manager tools to create a host for virtual machines.

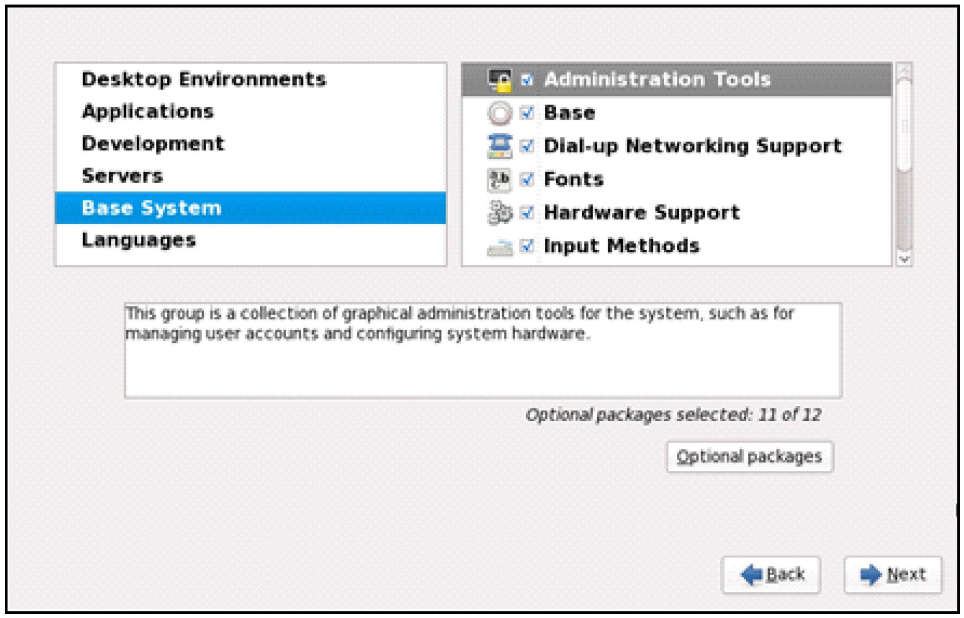
Desktop: This option provides the OpenOffice.org productivity suite, graphical tools such as the GIMP, and multimedia applications.

Software Development Workstation: This option provides the necessary tools to compile software on your Red Hat Enterprise Linux system.

Minimal: This option provides only the packages essential to run Red Hat Enterprise Linux. A minimal installation provides the basis for a single-purpose server or desktop appliance and maximizes performance and security on such an installation.

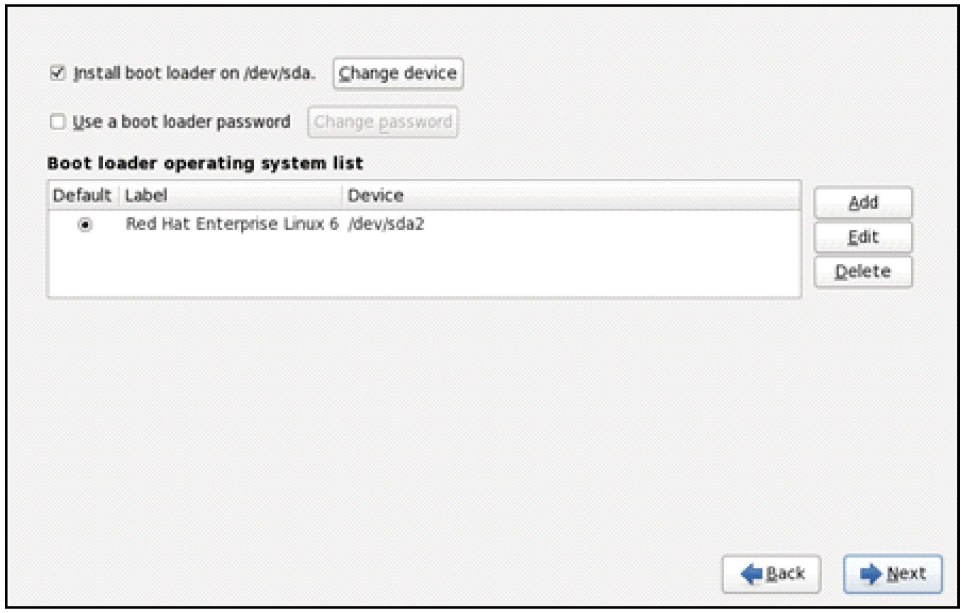


Step 14: Customizing the Software Selection



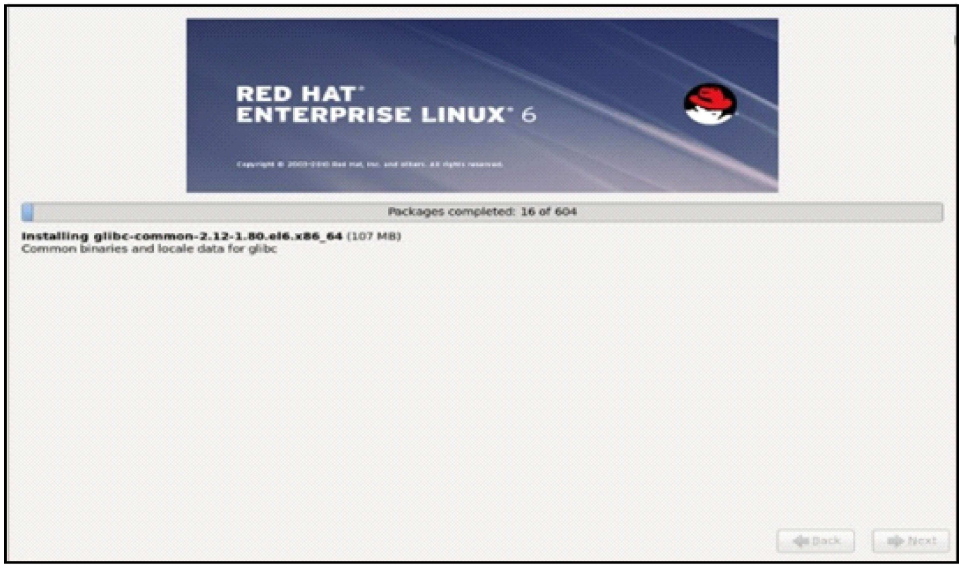
Step 15: Boot Loader Configuration

To boot the system without boot media, you usually need to install a boot loader. A boot loader is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the operating system kernel software. The kernel, in turn, initializes the rest of the operating system.



Step 16: Installation Progress

After installation completes, select Reboot to restart your computer. Red Hat Enterprise Linux ejects any loaded discs before the computer reboots.



3.6 INSTALLING AND CONFIGURING SOFTWARE IN LINUX

The standard Linux package format (according to the Linux Standard Base) is RPM. RPM is a packaging system originally developed by Red Hat and widely used in the Linux community. Distributions using it include Fedora, Mandriva, Red Hat (naturally), and SUSE. An RPM package file normally will be named something like program-version-other.rpm

Another popular package format is DEB, the Debian software package. Debian packages and the Advanced Packaging Tool (APT) were the first to introduce several advanced features that are now common, such as automatic dependency resolution and signed packages. Debian packages are used by Debian GNU/Linux (naturally), and distributions based on it, including Ubuntu, Knoppix, and Mepis. A Debian package file normally will be named something like program-version-other.deb

Remember, you will need to become SuperUser to install software.

3.6.1 Debian, Ubuntu: apt

There is a broad array of tools for working with DEB packages, but the one you will commonly use is apt-get, arguably the easiest of Linux package management tools. Apt-get is so easy because it not only keeps track of what packages are installed, but also what other packages are available. It will even download them from the Internet for you (if properly configured).

```
apt-get install ${packagename}
```

To remove software is just as easy.

```
apt-get remove ${packagename}
```

Although the repositories that contain installable packages might live on the Internet or on a disc somewhere, APT keeps a local database on your hard drive with a list of all available packages and where to find them. This database needs to be explicitly updated. To update the APT database:

```
apt-get update
```

A common idiom is to update your package database, and then upgrade all the packages that have patches or security updates to install. The following command will do this all at once.

```
apt-get update; apt-get upgrade
```

3.6.2 Fedora, Red Hat: yum

yum does for RPM packages roughly what apt-get does for Debian packages. Like apt-get, yum can download and install packages from a configured repository.

```
yum install ${packagename}
```

To remove software is just as easy.

```
yum remove ${packagename}
```

yum does not keep a local copy of your package database by default, so normally there is no need to update it.

To install all available security patches and bug fixes, use this command:

```
yum update
```

You can also explicitly update a single package with:

```
yum update ${packagename}
```



CHECK YOUR PROGRESS

Q.5: Linux cannot be installed on Storage Area Networks (SANs). (TRUE/FALSE)

Q.6: Which account gives you total control of the Linux system?

.....

Q.7: Which package group provides MySQL and PostgreSQL databases?

.....

Q.8: Which package group provides OpenOffice.org productivity suite?

.....

Q.9: What needs to be installed to boot the system without boot media?

.....

Q.10: What is the command to install software in Debain?

.....



3.7 LET US SUM UP

- Both Red Hat Enterprise Linux versions on both 32-bit and 64-bit platforms have the same system requirements for installation: Pentium 4 or higher CPU of 2Ghz or higher speed, 1 GB minimum RAM and 4 GB minimum hard disk.
- A hard disk can be divided into several partitions. Each partition functions as if it were a separate hard disk.
- The information about how a hard disk has been partitioned is stored in its first sector (that is, the first sector of the first track on the first disk surface). The first sector is the master boot record (MBR) of the disk;

this is the sector that the BIOS reads in and starts when the machine is first booted.

- For a proper Linux installation in a single-user desktop system, three partitions are recomended: swap, root, and home.
- The swap partition is the space on the hard drive that can be used as virtual memory. Virtual memory allows the computer to run large programs and perform complex tasks even if it does not have enough physical RAM to do the job.
- The root (/) partition is the top of the directory tree, and contains Linux and everything that you install with Linux.
- The home directory is the place where all the user-specific files are stored. On a multi-user system, each user will have their own directory under/home.
- You can install Red Hat Enterprise Linux on a large variety of storage devices– either basic or specialized storage devices. Basic Storage Devices are hard drives or solid-state drives connected directly to the local system. Specialized Storage Devices include Storage area networks (SANs), Direct access storage devices (DASDs), Firmware RAID devices and Multipath devices.
- The root account is used to install packages, upgrade RPMs, and perform most system maintenance. Logging in as root gives you complete control over your system.
- By default, the Red Hat Enterprise Linux installation process loads a selection of software that is suitable for a system deployed as a basic server.
- To boot the system without boot media, you usually need to install a boot loader. A boot loader is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the operating system kernel software. The kernel, in turn, initializes the rest of the operating system.
- To install packages on Debain and Ubuntu apt-get install ${packagename}. To remove software use apt-get remove ${packagename}
- To install packages on Fedora and Red Hat yum install ${packagename}. To remove software use yum remove ${packagename}



3.8 FURTHER READING

1) Das, S. (2012). *Your UNIX*/Linux: *The Ultimate Guide*. (Third Edition). (Science/Engineering/Math). McGraw-Hill.

2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



3.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: 2 GHz

Ans. to Q. No. 2: MBR is the sector that the BIOS reads in and starts when the machine is first booted.

Ans. to Q. No. 3: swap, root, and home.

Ans. to Q. No. 4: root **Ans. to Q. No. 5:** FALSE

Ans. to Q. No. 6: root **Ans. to Q. No. 7:** Database Server

Ans. to Q. No. 8: Desktop **Ans. to Q. No. 9:** boot loader

Ans. to Q. No. 10: apt-get install ${packagename}



3.10 MODEL QUESTIONS

Q.1: What are the system requirements for installing Red Hat Enterprise Linux?

Q.2: What are partitions? What are extended and logical partitions?

Q.3: What are the different recommended partitions for a proper Linux installation in a single-user desktop computer?

Q.4: What are the different kinds of storage devices where Linux can be installed?

Q.5: What are the different options available in the Disk Partition Setup of Linux installation?

Q.6: What are the different package groups available during Linux installation?

*** ***** ***

UNIT 4: LINUX KERNEL

UNIT STRUCTURE

4.1 Learning Objectives

4.2 Introduction

4.3 Linux Kernel Concepts

- 4.3.1 Functions of Linux Kernel
- 4.3.2 Architecture of Linux Kernel
- 4.3.3 Features of Linux Kernel

4.4 Device Drivers

- 4.4.1 Block and Character Devices
- 4.4.2 Major and Minor Numbers

4.5 System Startup

4.6 System Shutdown

4.7 Redirection

- 4.7.1 Standard Input
- 4.7.2 Standard Output
- 4.7.3 Standard Error

4.8 Piping

4.9 Let Us Sum Up

4.10 Further Reading

4.11 Answers To Check Your Progress

4.12 Model Questions

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define Linux kernel
- describe the architecture of the Linux kernel
- describe the role of the kernel in the Linux system
- define device driver
- explain how a device is represented in Linux system
- define block device and character device.

4.2 INTRODUCTION

We have already learned the basic concepts of the Linux operating system like what type of operating system it is. We have learnt about the basic features of the Linux operating system. We are also acquainted with the basic ideas about the Linux Distributions throughout the previous units as well as the installation process and the requirement for the installation, how to partition the Hard drive in Linux, how to install the Linux system. Besides these, we have also learned how to install and configure different software or application in Linux in the previous units.

In this unit, we will learn about the Linux kernel, the role of the kernel in the Linux system and the architecture of the Linux kernel. By going through this unit, we will get to know about the concepts of device drivers in Linux system in addition to booting and shutting down process of the system. The important features such as Redirection and Piping and how they work will also be learnt through this unit besides knowing about the standard input, standard output and the standard error in Linux system. In the next unit, we will learn about Linux file system.

4.3 LINUX KERNEL CONCEPTS

Kernel is a special code written in C language which directly interacts with hardware. It is at the core of the Linux system. Kernel runs simultaneously a number of processes. It acts as a manager for various resources like CPU, memory, process etc. Linux kernel is efficient in terms of both memory and CPU utilization.

Kernels are of two types - *monolithic kernel* and *micro kernel.* Linux kernel is a monolithic kernel because it lumps all of the basic services into the kernel.

The Linux kernel is layered into a number of distinct subsystems. The major subsystems are shown in the following figure 4.1.

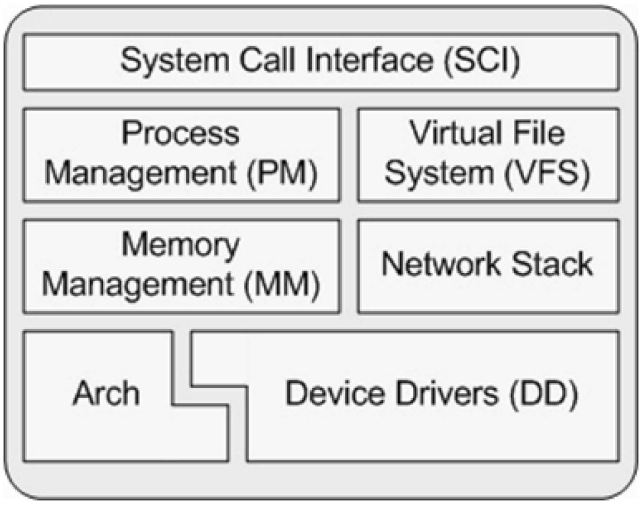


Figure 4.1: Subsystems of Linux Kernel

4.3.1 Functions of Linux Kernel

The functions performed by kernel include all the subsystems of the kernel. These functions are described below.

1) **Process Management:** The kernel handles the creation and destruction of the processes and their connection to the input and output systems. The interprocess communication, i.e. the communication among different processes is also handled by the kernel. Process management gives the information of what's happening with a process, file allocated to process, state of process (like running, waiting, stop) etc.
2) **Memory Management:** The memory management is the most important function of kernel. Kernel handles assignment of address space to processes and applications, basically assigning virtual memory instead of physical memory, whereby later the actual address space in RAM is assigned and the conversion of physical address to virtual address is done by MMU (Memory Management Unit).
3) **File System Management:** In Linux, almost everything is treated as a file. The Linux file system is a hierarchically data structured tree which organizes the file system. The file system is organized as a tree structured directory, where a directory is a group of

files. Two types of directory available in Linux system are: 1) root and 2) Sub directory. The root is the main directory and sub directories can be created or renamed by the user under the root directory. The Linux file system supports a number of file types like ext2, ext3, device file, block file, networking file system etc. and each file system is stored in different disk partition.

4) **System Call Interface:** The system call interface performs function calls from user space into the kernel. It can be architecture dependent, even within the same processor family.
5) **Device controlling:** The system operations that map to different physical devices need to perform device control operations which are performed by code that is specific to the device being addressed. Those codes are called device drivers. For every peripheral connected to a system, there is a device driver embedded in the kernel
6) **Network Management:** The system is responsible delivering data packets across program and network interfaces. The kernel handles the routing and address resolution issues in networking.

4.3.2 Architecture of Linux Kernel

The architecture of the Linux kernel can be divided into two levels- User Space and Kernel Space. The architecture is shown in the following figure.

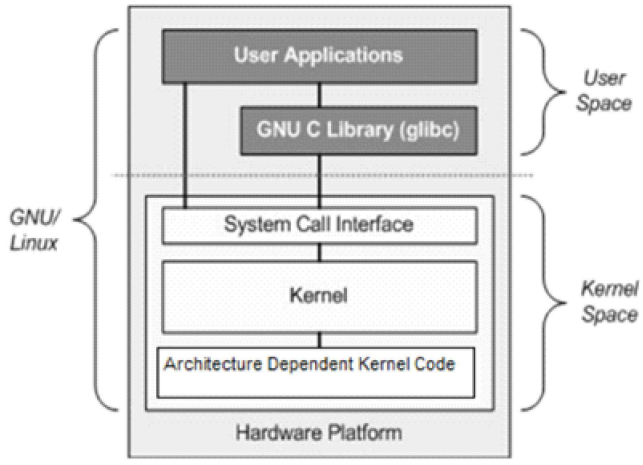


Figure 4.2: Linux Kernel Architecture

User Space: User space is where all the user programs and applications are executed. User Space cannot directly access the memory and hardware. It accesses those through kernel space. So, user space application needs to be switched to kernel space and the switching mechanism is provided by the GNU C library.

Kernel Space: Kernel space is for executing kernel programs. It accesses full part of memory and directly interacts with the hardware. The kernel space is subdivided into three levels– System Call Interface, Kernel and Architecture-Dependent code. System call interface is the intermediate layer between user space and kernel space that implements the basic functions such as *read* and *write.* Kernel is architecture-independent code which is common to all hardware processors. The architecture-dependent code serves as the processor and platform specific code for the given architecture.

4.3.3 Features of Linux Kernel

Linux is an open source operating system. It supports a large number of networking protocols, including the typical TCP/IP. Linux kernel is a dynamic kernel that supports the addition and removal of software components on the fly and these are called dynamically loadable kernel modules, and they can be inserted at boot when they are needed. While much of Linux is independent of the architecture on which it runs, there are elements that must consider the architecture for normal operation and for efficiency.



CHECK YOUR PROGRESS

Q.1: What is is kernel in Linux System? Describe the role of the Linux Kernel.

.....

.....

Q.2: The architecture of Linux kernel can be divided into user space and .....

4.4 DEVICE DRIVERS

The Device drivers have a special role in the kernel of the Linux system. All the devices are represented by files. Almost all system operations map to different physical devices and all device control operations are performed by some routines that are specific to a device. Those routines are known as device driver. The kernel calls the appropriate device driver, whenever a particular device is accessed and then the kernel passes some parameter to it for it to act properly.

The device files are stored in /dev directory or in its sub-directories. The device files contain no data. The device names connected to the Linux system can be found by using the following command.

```
$ ls  -l   /dev
```

```
total 0
crw———     1 root root     10, 55 Feb  3 18:11 acpi_thermal_rel
crw———     1 root root    10, 235 Feb  3 18:11 autofs
drwxr-xr-x  2 root root        400 Feb  3 18:10 block
drwxr-xr-x  2 root root         80 Feb  3 18:10 bsg
crw———     1 root root    10, 234 Feb  3 18:10 btrfs-control
drwxr-xr-x  3 root root         60 Feb  3 18:10 bus
lrwxrwxrwx  1 root root          3 Feb  3 18:11 cdrom –> sr0
lrwxrwxrwx  1 root root          3 Feb  3 18:11 cdrw –> sr0
drwxr-xr-x  2 root root       3900 Feb  3 18:11 char
crw———     1 root root     5,   1 Feb  3 18:11 console
lrwxrwxrwx  1 root root         11 Feb  3 18:10 core –> /proc/kcore
drwxr-xr-x  2 root root         60 Feb  3 18:10 cpu
crw———     1 root root     10,  59 Feb  3 18:11 cpu_dma_latency
crw———     1 root root    10, 203 Feb  3 18:10 cuse
drwxr-xr-x  8 root root        160 Feb  3 18:10 disk
drwxr-xr-x  2 root root         80 Feb  3 18:10 dri
crw———     1 root root   246, 0 Feb3 18:11 drm_dp_aux0
crw———     1 root root     246, 1 Feb  3 18:11 drm_dp_aux1
crw———     1 root root     246, 2 Feb  3 18:11 drm_dp_aux2
```

```
lrwxrwxrwx   1 root root        3 Feb  3 18:11 dvd –> sr0
lrwxrwxrwx   1 root root        3 Feb  3 18:11 dvdrw –> sr0
crw———       1 root root   10, 61 Feb  3 18:11 ecryptfs
crw-rw——     1 root video   29, 0 Feb  3 18:11 fb0
lrwxrwxrwx   1 root root       13 Feb  3 18:10 fd –> /proc/self/fd
crw-rw-rw-   1 root root    1, 7 Feb  3 18:11 full
```

4.4.1 Block and Character Devices

Most of the devices are either block or character devices. These devices are indicated by the first character (either **b** or **c**) in the permission field. If the first character of the permission field is **b** then it is a block device or block special device and if it is **c** then it means that it is a character device or character special device.

A block device can store or transfer a block of data at a time, whereas a character device can display, process or store a character at a time.

The floppy drive, CD-ROM and the hard disk drive are the block special devices. All data are read from or written into these devices in blocks and use the buffer cache. The terminal, tape drive and printer are the character special devices. The read/write operations for character devices do not use the buffer cache and access the device directly.

4.4.2 Major and Minor Numbers

The fifth field in the file list representing the size for the other files, consists of a pair of two numbers, separated by a comma. These numbers called the *major* and *minor* device numbers respectively, where the major number represents the type of device and the minor number represents the special characteristics of the device. For example, **tty0** and **tty1** have the same major number (4) but different minor numbers (0 and 1).

```
crw—w——   1 root tty     4, 0 Feb  3 18:11 tty0
crw—w——   1 root tty     4, 1 Feb  3 18:11 tty1
```



CHECK YOUR PROGRESS

Q.3: What are block device and character device?

.....

.....

.....

Q.4: What do the major and minor number for a device represent?

.....

.....

Q.5: What is the functionality of a device driver?

.....

.....

4.5 SYSTEM START UP

To start up the system, you have first power on the machine such that the power is supplied to the peripherals that are immediately required. Because, when the kernel is loaded those peripherals are also be loaded.

The start up or booting process goes through a number of stages. The kernel is loaded into the memory at first. It is represented by the file **/vmlinuz.**

BIOS (Basic Input/Output System) performs some system integrity checks. It searches, loads, and executes the boot loader program. It searches the boot loader in floppy, CD-ROM, or hard drive. In simple terms BIOS loads and executes the **MBR** (Master Boot Record) which is located in the 1st sector of the bootable disk. Typically **/dev/hda**, or **/dev/sda**. The MBR then loads and executes the **GRUB** (Grand Unified Bootloader) that has the knowledge of the file system. The GRUB configuration file is **/boot/grub/grub.conf**. The GRUB loads and executes Kernel and then the kernel starts spawning further processes, among which the most important is **init**. The kernel executes init by executing the program **/sbin/init**. Since, init is the first process to be executed by the kernel, its PID (process id) is 1. The init then in turn spawns further processes.

The init controls a number of modes called run level. The available run levels are–

- 0 – halt
- 1 – Single user mode
- 2 – Multiuser, without NFS
- 3 – Full multiuser mode
- 4 – unused
- 5 – X11
- 6 – reboot

- Looks at the /etc/inittab file to decide the Linux run level. init identifies the default init level from /etc/inittab and uses that to load all appropriate program. Depending on the default init level, the system will execute the programs from one of the following directories.
  - Run level 0 – /etc/rc.d/rc0.d/
  - Run level 1 – /etc/rc.d/rc1.d/
  - Run level 2 – /etc/rc.d/rc2.d/
  - Run level 3 – /etc/rc.d/rc3.d/
  - Run level 4 – /etc/rc.d/rc4.d/
  - Run level 5 – /etc/rc.d/rc5.d/
  - Run level 6 – /etc/rc.d/rc6.d/

4.6 SYSTEM SHUTDOWN

The **shutdown** command brings the system down in a secured way. The **/sbin/shutdown** command controls the shut down sequence. **shutdown** does its job by signalling the **init** process, asking it to change the run level. The activities performed by **shutdown** are–

- Notifies the users that the system is going down.
- Notifies all the running processes that the system is going to be shut down by sending. signals so that the processes can terminate normally.
- Unmounts all the secondary file system.

Different options used with ***shutdown*** command are given in Table 4.1.

Table 4.1: Options used with Shutdown Command

Option	Activity
–a	Control access to the shutdown command using the control access file /etc/shutdown. allow. See Access Control below for more information.
–k	Do not shut down, but send the warning messages as if the shutdown were real.
–r	Reboot after shutdown.
–h	Instructs the system to shut down and then halt.
–P	Instructs the system to shut down and then power down.
–H	If –h is also specified, this option instructs the system to drop into boot monitor on systems that support it.
–f	Skip fsck after reboot.
–F	Force fsck after reboot.
–n	Don't call init to do the shutdown of processes; instruct shutdown to do that itself. The use of this option is discouraged, and its results are not always predictable.
–c	Cancel a pending shutdown. (This does not apply to "shutdown now", which does not wait before shutting down.) With this option, it is not possible to give the *time* argument, but you can still specify an explanatory message that will be sent to all users.
–t *sec*	Tell init to wait *sec* seconds between sending processes the warning and the kill signal, before changing to another runlevel.
Time	The *time* argument specifies when to perform the shutdown operation. The *time* can be formatted in different ways: First, it can be an absolute time in the format *hh*:*mm*, in which *hh* is the hour (1 or 2 digits, from 0 to 23) and *mm* is the minute of the hour (in two digits).

	Second, it can be in the format +*m*, in which *m* is the number of minutes to wait. Also, the word now is the same as specifying +0; it shuts the system down immediately.
message	A message to be sent to all users, along with the standard shutdown notification.

Example 4.1: Shutdown examples

shutdown 8:00

Schedule the system to shut down at 8 A.M.

shutdown 20:00

Schedule the system to shut down at 8 P.M.

shutdown now

Bring down the system immediately.

shutdown –r now

Bring down the system immediately, and automatically reboot it.

shutdown –P now

Bring down the system immediately, and automatically power off the system.

4.7 REDIRECTION

Redirection is a special feature of Linux system through which one can direct the input or output to some other stream other than the standard ones. The redirection operators are **>**, **<** and **>>**. Here, **<** is the input redirection operator, **>** is the output redirection operator and **>>** appends output to an existing file.

The input redirection operator (<) instructs the shell to redirect the input. It can be used to take input from a file. For example, the following command:

```
$ wc  <  thisfile.txt
```

will count and display the number of lines, words and characters in the file thisfile.txt. If the content of the thisfile.txt is as follows–

This is a text file.
It is used to test the cat and the wc commands.
The redirection and the piping can also be tested by using this file.
And this is the end of the content of the file.

Then the wc < thisfile.txt command will work as follows–

```
$ wc  <  thisfile.txt
```

4 40 187

Here, 4 is the number of lines, 40 is the number of words and 187 is the number of characters.
But, without redirection it will work as-

```
$ wc   thisfile.txt
```

4 40 187 thisfile.txt

i.e., it gives the file name too.

The output of any command can be redirected by using the output redirection operator. For example, the command:

```
$ cat  thisfile.txt  >  thatfile.txt
```

will write the contents of the file into an another file thatfile.txt.
It works as given below–

```
$ cat  thisfile.txt  >  thatfile.txt
```

```
$ cat  thisfile.txt
```

This is a text file.
It is used to test the cat and the wc commands.
The redirection and the piping can also be tested by using this file.
And this is the end of the content of the file.

```
$ cat  thatfile.txt
```

This is a text file.
It is used to test the cat and the wc commands.
The redirection and the piping can also be tested by using this file.

And this is the end of the content of the file.

The input and output redirection operators can also be combined to redirect both the input and output. For example,

```
$ wc  <  thisfile.txt  >  count_file.txt
```

command will save the number of lines word and characters of the file ‘‘thisfile.txt’’ into another file ‘‘count_file.txt’’.

```
$ cat
count_file.txt
```

4 40 187

So, in redirection the command does not know about the source from where the character stream come or destination to where the character stream is to be written.

4.7.1 Standard Input

The standard input is a character stream that comes from a particular source. A source may be a device or a file or a program or command.

Some commands (like cat and wc) can operate on operate on a character stream without knowing the source of the stream. If you write cat and wc command without any argument as follows–

```
cat
wc
```

Then these commands will receive character stream from the default source (i.e. from the *keyboard*). This stream is called standard input. The sources from which the standard input can come from are–

- the keyboard, which is the default source.
- a file (given by using the redirection operators >,<,>>)
- a program or command (given by using pipe)

For example,

```
$ wc
```

Hello, I am wc command.

can count the number of lines, words and characters.

<Ctrl d>

2 15 79

```
$ wc  <  thisfile.txt
```

4 40 187

4.7.2 Standard Output

Standard output writes the output generated by a program or command in the form of a character stream. When the standard output is not redirected (i.e. not given by using redirection operator), it will send the output to the *terminal*, which is the default destination. For example, if you write the **cat** and **wc** command as follows–

```
cat
wc
```

Then these command will send the output as a character stream to the terminal. This stream is called the standard output. Here, the shell sets up a connection between the command and the terminal to display the output on the screen.

Again, the standard output can be redirected. For example,

```
$ wc   thisfile.txt  >  thatfile.txt
```

This command sends the number of lines, number of words and the number of characters in the file **thisfile.txt** to the file **thatfile.txt.**

The standard output can also serve as input to an another program.

Thus the standard output has three destinations– the terminal (which is default destination), a file (redirected by using redirection

operator) and another command or program (given by suing the concept of piping).

4.7.3 Standard Error

The standard error writes error messages generated by a program or command that fails at some point in its execution. It writes the error message to the terminal. Like standard output, the default destination of standard error is the terminal display. The output redirection operator can also be used to redirect the error to a file. But it needs to prefix the file descriptor. A **file descriptor** is a number assigned to a standard file that identifies the standard file. The file descriptors 0,1 and 2 represent the standard input, standard output and standard error respectively.

Example,

```
$ cat   testfile.txt   >  newfile.txt
```

cat: testfile: No such file or directory

```
$ cat   testfile   2>  newfile
```

Here, standard error has been redirected by using file descriptor (2).

```
$ cat   newfile
```

cat: testfile: No such file or directory

4.8 PIPING

Pipes are used to redirect a standard input or standard output from one command or program to another. The shell enables the standard output of one command to be connected as a standard input to another. This is called **piping.** Piping is done by using a special operator "**|**" called **pipe** that acts as the connector of two commands.

```
$ ls
```

file1.txt file2.txt file3.txt file4.txt

```
$ ls | wc
```

4 4 40

Here, the **ls** command passes its output directly to the input of **wc** then the **wc** command gives the output as above.

```
$ ls  file1.txt  |  wc
```

1 1 10

```
$ cat  file1.txt  |  wc
```

1 5 19

Here, the **cat** command passes its output (i.e. the content of the file **file1.txt**) to the input of **wc** command and **wc** command gives the final output which is displayed on the screen.



CHECK YOUR PROGRESS

Q.6: What is the use of redirection?

.....

.....

.....

.....

Q.7: What do the standard input, standard output and standard error actually represent?

.....

.....

.....

.....

.....

.....

Q.8: What is a pipe?

.....

.....



4.9 LET US SUM UP

- Kernel is the core component of Linux system that interact with the hardware.
- A device is simply a file.
- A device driver is a routine of codes specific to a particular device that performs the control operation for that device.
- In Linux, the system start up process goes through a series of stages.
- Shutdown command is used with several options for shutting down a system.
- Redirection is used to redirect the output to some other stream other than the standard inputs and outputs.
- The standard inputs, standard outputs and the standard errors are actually character streams.
- Piping is the technique of redirecting a standard input or output from one command to another.



4.10 FURTHER READING

1) Das, S. (2000). *UNIX*: *Concepts and Applications*. Tata McGraw-Hill.
2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



4.11 ANSWER TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: Kernel is a small and special code that interacts with the hardware of the system. It is the core part of the Linux system.

Ans. to Q. No. 2: kernel space.

Ans. to Q. No. 3: A block device is one that stores or transfers a block of data at a time. On the other hand, a character device is one that stores, processes or displays one character data at a time.

Ans. to Q. No. 4: For a device, the major numbers represents the type of the device and the minor number represents the special characteristics of the device.

Ans. to Q. No. 5: A device driver performs all the control operations of the device associated to it.

Ans. to Q. No. 6: The redirection is used with the some commands either to take input from a source other than standard input device (i.e. the keyboard) or to give out to a destination other than standard output device (i.e. the terminal) or for both.

Ans. to Q. No. 7: The standard input is the character stream given to a command that comes either from keyboard or from other source.

The standard output is the output character stream of a command which is passed either to the terminal or to another command.

The standard error is the character stream that contains the error message generated by a command that fails at some point in its execution which is passed either to the terminal or to another command.

Ans. to Q. No. 8: A pipe is an operator used to redirect a standard input or standard output from one command or program to another.



4.12 MODEL QUESTIONS

Q.1: What is Kernel? Describe the functions and the architecture of the kernel.

Q.2: What is a device driver? How can you identify whether a device file is block or character?

Q.3: Describe the stages performed by the start up process of Linux system.

Q.4: Describe the options used with the shutdown command in Linux.

Q.5: What are redirection and piping? Explain the functionality of these two technique.

Q.6: Differentiate between standard input, standard output and the standard error.

*** ***** ***

UNIT 5: BASICS OF LINUX FILE SYSTEM

UNIT STRUCTURE

5.1 Learning Objectives

5.2 Introduction

5.3 Basics of Linux File System

- 5.3.1 Ext
- 5.3.2 Ext2
- 5.3.3 Ext3
- 5.3.4 Ext4
- 5.3.5 XFS
- 5.3.6 JFS
- 5.3.7 Reiser FS
- 5.3.8 ISO 9660

5.4 Basic Types of Files

- 5.4.1 Regular Files
- 5.4.2 Special Files
- 5.4.3 Directories
- 5.4.4 Other Common Types of Files

5.5 Inodes

5.6 File Attributes

5.7 Let Us Sum Up

5.8 Further Reading

5.9 Answer to Check Your Progress

5.10 Model Questions

5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define file and file system
- describe different file system types
- explain the features of different file system
- compare different extended file systems

- describe the journaling concept in file system
- discuss different file types used in Linux system
- describe Linux command to access those file types
- describe different file attributes.

5.2 INTRODUCTION

We are already familiar with the features of Linux Operating Systems and the working of Linux kernel. We have also understood the redirection and piping concepts in Linux system. In this unit we will learn about what a file system is and the properties of different Linux file systems. We will learn here about the functions of different commonly used file systems in Linux. We will study the different types of files used in Linux. We will study about I-node and file attributes. In the next unit we will learn about different paths, mounting and un-mounting of file system and organization of file tree.

5.3 BASICS OF LINUX FILE SYSTEM

In Linux system, all data and programs are stored in file. There is not any restriction on the structure of a file. It is a collection of related data. The size of a Linux file is the exact size of the contents that stored inside the file. That means it does not need its own size for storage.

A Linux file or a file system is said to be the heart of it. The files in a file system are connected logically. A file system manages how the files are stored and can be retrieved easily. It is not possible for an operating system to know the positioning of a file i.e., where a file starts and where ends, without a file system. It is a method through which operating system keeps track of how files are organised in a disk partition so that it can be found easily. There are different file systems present for Linux OS. They are different from each other based on the structure, logic and other important properties like flexibility, speed, size etc. We will study some of the file systems that are being commonly used for better understanding of the functionalities of a file system.

File System Types: Any Linux distribution has different types of file system choices to partition a disk. Some of them are given below:

- EXT
- EXT2
- EXT3
- EXT4
- XFS
- IFS
- REISER FS
- ISO9660 etc.

5.3.1 EXT

Ext stands for extended file system. It is one of the first file system used in Linux. It was created in April, 1992 especially for Linux operating system by Remy Card. It is the first version in the series of extended file systems. It was replaced by ext2 in 1993.

5.3.2 EXT2

Ext2 or the second extended file system is created for Linux OS. It is an improved version of extended file system. It was also designed by Remy Card. It is one of the most popular and commonly used file systems in Linux and it has been used for many years.

Here, the data stored in files are kept in data blocks. All the data blocks have same length but the length is not fixed in all ext2 file systems. The size of the file system is depended on the block size.

The block size may vary from 1 KB to 8 KB. The maximum size of the ext2 file system is from 2 TB to 32 TB whereas the maximum file size is from 16 GB to 2TB. The filename size can be of maximum of 255 characters.

The limitation of ext2 file system is that it takes very long time to recover after an unclean shutdown which may occur due to situations like power failure or system crash. This problem leads to development of the improved version of ext2.

5.3.3 EXT3

Ext3 is the third extended file system in Linux. It is an extended version of ext2 file system introduced in November 2001 with Linux. It was developed by Stephen Tweedie. It overcomes the limitation ext2 file system by introducing the concept of **journaling**.

In addition to this, ext3 also adds two more features viz. growth of online file system and HTree indexing for larger directories. An ext3 file system works similar to ext2 file and an ext2 file system can easily be upgraded to ext3. Ext3 file system is used as default file system in many standard Linux distributions.

A **journaling file** system handles a special file known as a journal. It is used to keep track of changes that not committed to the file system completely. It is used for correcting the inconsistencies that may occur at the time of system crash or power failure. It may keep track of both stored data and metadata. This file system keeps the information in a log area. After the updating of log, the system wrote the actual data in the appropriate areas of the file system. After a file system crash, it can be repaired very quickly back online using the journal log. So, there is a very little chance of data loss in journaling file system.

Ext2 file system runs a program called ***e2fsck*** *(ext2 file system check)* at the time of rebooting after an unclean shutdown. It checks whether any problem occurs during the improper shutdown and tries to fix the problems before the file system can be mounted. Thus ext2 tries to maintain the data integrity. But the problem is that e2fsck checks the entire file system which may take a long time to recover.

The main advantage of ext3 file system is that it supports journaling. Ext3 file system does not need any checking in the file system at the time of rebooting after an unclean shutdown unlike ext2 file system since it guarantees the internal consistency of the system.

The size of the file system and file size is similar to file system of ext2. The maximum size of the ext3 file system is from 4 TB to 32 TB whereas the maximum size of a file is 16 GB to 2 TB. The filename size is also same as ext2 i.e. maximum of 255 characters.

The disadvantage of the ext3 file system is that it occupies more system memory to store the information and can make the I/O system little slower. In spite of this, many of the distributions use ext3 file system for its faster recovery.

5.3.4 EXT4

Ext4 or fourth extended file system is an improved version of ext3. Stable version of ext4 was introduced with Linux in October, 2018. It was developed as an extended version of ext3 in order to increase the size of the file and entire file system to improve the performances. The features of ext4 file system are as follows:

- **Compatibility:** Any ext3 file system can be upgraded to ext4 easily. So, we can easily increase the size of the file system and improve the performance without doing any formatting or re-installation of the operating system. Ext4 uses the new data structures for the new data only. The old data can be modified whenever required.
- **Extents:** Ext2 and ext3 file systems use an indirect block mapping scheme to keep track of blocks used for particular data. It is not efficient for the larger files. An extent is a collection of contiguous physical blocks. A single extent can have a contiguous block size of 128 MB where size of a single block is 4 KB.
- **Multiblock Allocation:** In ext file system there is a block allocator to decide which free block will be used at the time of writing a new data. But it allows only one block at a time. Therefore it is not efficient technique for writing a file of large data size. To improve the performance, ext4 uses multiblock allocator which can allow more than one block at a single call.

- **Journal Checksumming:** Ext4 file system checksums the journal data as it is one of the most used files in the disk. It is used to check whether any journal block has corrupted or not.
- **Delayed Allocation:** In ext4 blocks are not allocated to data immediately. There is a delay in allocating the block unlike ext3 and the files are kept in cache. For this, block allocator gets the opportunity to optimize the situation.
- **No Journal Mode:** A journaling file system keep a track of information of the current changes. For this, it occupies little storage. Some programs may have special requirement where it can run without journal. Then, journaling option can be disabled in ext4 file system.
- **Larger File System and File Sizes:** Ext4 file system has significantly larger sizes than ext3 file system. A file in ext4 can have a maximum size of 16 TB. The file system has maximum size of 1 EiB or 1 **ExbiBytes [1 EiB = 2^{60} bytes].**

The comparison of the different features of ext2, ext3 and ext4 file system is shown in Table 5.1.

Table 5.1: Comparison of Three Extended File System

Criteria	Ext2	Ext3	Ext4
Developed in	It was developed in 1993	It was introduced with Linux in 2001	It was introduced with Linux in 2008
Length of Filename size	255 bytes	255 bytes	255 bytes
Maximum file size	16 GB – 2 TB	16 GB – 2 TB	16 TB
Maximum size of	2 TB – 32 TB	2 TB - 32 TB	1 EiB
File system Feature	No journaling	Journaling file system	Journaling file system

5.3.5 XFS

XFS file system is a high-performance and highly scalable 64 bit system. It supports journaling. It was created by Silicon

Graphics and introduced in 1994 with IRIX. It was used in Linux kernel in 2001. This file system is supported by almost all Linux distributions. It also ensures the consistency of the data using the concept of journaling. For space allocation extents with stored data structure in B+ tree is used. One of the unique features of XFS file system is pre-allocation of input-output bandwidth at previously determined rate which may be beneficial for real time applications. XFS file system has block size of 512 bytes to 64 KB.

5.3.6 JFS

JFS stands for journaled file system. It is a 64 bit file system created by IBM. Its first version was developed for AIX operating system of IBM in 1990 known as jfs1. JFS2 was introduced in 2001 with Linux OS.

The term “journaled” depicts that the file system keeps the record of ongoing modification of files in a log file. This log file is used to recover the loss that may occur during an improper shutdown. It uses B+ tree to search in directories. As ext4 has better performance compare to JFS, so it is not widely used in Linux.

5.3.7 Reiser FS

Reiser file system was introduced in 2001 with Linux. It was developed by a team named Namesys, which was led by Hans Reiser. It is general purpose file system supports journaling. It is a default file system in many of the Linux distribution like Elive, Xandros, and Linspire etc. Along with journaling, it supports a scheme known as tail packing to reduce internal fragmentation. The performance of ReiserFS is comparatively faster.

5.3.8 ISO 9660

ISO 9660 is a file system created for optical media. It is published by ISO. The main aim of this file system is that it can support

in any other standard operating system like Linux, Windows and Mac etc. for exchanging the data between the media and the system.



CHECK YOUR PROGRESS

Q.1: Which among the following file system has introduced the concept of journaling first?

a) EXT b) EXT2 c) EXT3 d) EXT4

Q.2: ExbiBytes is equal to ___ bytes?

a) 2^{18} b) 2^{32} c) 2^{52} d) 2^{60}

Q.3: Which of the following file system is created for optical media?

a) Reiser FS b) ISO 9660 c) XFS d) EXT2

Q.4: Which of the following file system was created by IBM?

a) JFS b) IFS c) ISO 9660 d) EXT

Q.5: What is the maximum size of ext3 file system?

a) 2 TB b) 8 TB c) 16 TB d) 32 TB

Q.6: A program called e2fsck is run in .....?

a) EXT2 b) EXT3 c) EXT4 d) Reiser FS

5.4 BASIC TYPES OF FILES

Linux operating systems have different type of files. Most of the commonly used file types are regular files, directory files and device files.

5.4.1 Regular Files

Regular files or ordinary files are the mostly used files in Linux system. All the text files, binary files, executable program files, image files, shared libraries etc. are known as regular files. Regular files are denoted by a dash ('–') symbol.

Regular files can be created using touch, vi, gedit commands. For listing the regular files following Linux command is used:

```
ls - l | grep ^-
```

Where grep is a filter which will find all the regular files which are denoted by '–' symbol.

5.4.2 Directory Files

Linux stores all the data and programs in files. These files are contained in directories. A directory contains all the regular files, special files, other folders stored in a physical device. These files are denoted by symbol 'd'.

Directories can be listed using the following Linux command:

```
ls – l | grep ^d
```

Some of the important commands related with directories are given in Table 5.2.

Table 5.2: Commands Related to Directories with Example

Command Name	Purpose	Syntax
mkdir	For creating a new directory	**mkdir <directory name>** **e.g. mkdir dir111**
rmdir	For removing or deleting any empty directory from the system	**rmdir <directory name>** **e.g. rmdir dir111**
cd	For changing the directory in which an user is working currently	**cd <directory name>** **e.g. cd dir111** It will take the current working directory to dir111

*We will study these commands with their different attributes in detail in Unit 7.

5.4.3 Special or Device File

Device files or special files in Linux are the files that work as an interface to device drivers. It allows an application to interact with a device via any device driver. Most of the device files are stored

in /dev directory in Linux. Each device in the system should have an entry in /dev. These files are created during installing the system and device drivers. There are two different types of device files present in Linux operating systems. These are: **character special files** and **block special files.**

- **Character Special Files:** They are also known as character devices. Their behaviour is like serial ports, pipes etc. They provide a direct access to hardware devices. They do not allow any program to perform read or write operation. It is up to the device. Sometimes character devices are known as raw devices also. Character devices are symbolically denoted by 'c'.

  To list all the character files locate in /dev directory, following Linux command can be used:

  ```
  ls  -l /dev | grep ^c
  ```

- **Block Special Files:** They are also known as block devices. These files differ from the character devices in the fact that a block device allows a program to read or write a block of any size. These are buffered access to hardware devices whereas character devices are unbuffered access. Block devices are denoted by 'b'.

  To list all the symbolic link files locate in /dev directory, following Linux command can be used:

  ```
  ls  -l /dev | grep ^b
  ```

5.4.4 Other Common Types of Files

Besides these types of files, Linux system uses some other type of files also:

- **Symbolic Links:** They are actually either the regular files or directories. A symbolic link is a reference to another file located in a disk. This file is stored as referenced file's path. These files are denoted by **'l**'. It is also known as **soft link**. Each soft link contains a different inode value which directs to the original file.

A **hard link** has the same inode value as the original file. For this, it remains linked even the original file is moved to another location unlike a soft link file.

To list all the block files located in /dev directory, following Linux command can be used:

```
ls  -l /dev | grep ^b
```

- **Named Pipes:** It performs communication between two or more processes locally with each other using file. This file acts as a pipe among the processes. These are also known as FIFO. Named pipes are denoted using 'p'.

  To list all the named pipe files located in /dev directory, following Linux command can be used:

```
ls  -l /dev | grep ^p
```

- **Socket Files:** Socket files are used for inter process communication. It is used to pass the information between applications. The main difference between socket and named pipe is that a socket can perform communications between processes which are running on two different systems. But named pipes perform communication locally. A socket file can be created using a system call named *socket ()*. Socket files are denoted by's'.
- To list all the named pipe files locate in /dev directory, following Linux command can be used:

```
ls  -l /dev | grep ^s
```



CHECK YOUR PROGRESS

Q.7: Which of the following command is used to create a directory?

a) rmdir b) chdir c) mkdir d) crdir

Q.8: Ordinary files in Linux are denoted by which symbol?

a) l b) – c) r d) o

Q.9: Device files are located in directory called .....

a) /dev b) /device c) /etc d) /drivers

Q.10: Linux command rmdir is used to?

a) Create a new directory b) Delete a new directory

c) Rename a new directory

d) Change the current directory to this directory

Q.11: Which of the following is true?

a) A hard link file contains the same inode as that of referenced file

b) A soft link file contains the same inode as that of referenced file

c) Socket file cannot be used for inter process communication

d) Socket files are also known as FIFO

Q.12: What is a directory file?

a) A directory contains data.

b) A directory contains only images

c) A directory contains details of files and subdirectories it contains

d) A directory contains only the special files

Q.13: The most common type of file is:

a) Regular files b) Directories

c) Device file d) Socket files

Q.14: Which of the following Linux command can be used to list the detail of device files?

a) ls –l| grep ^– b) ls –l| grep ^dev

c) ls –l| grep ^d d) ls –l| grep ^–b

5.5 INODES

Inode or index node is a data structure that contains all the information regarding a file in Linux file system. It does not contain the actual content. Each file has an inode and they are identified by a unique inode number. Every file is assigned a filename and an inode number at the time of its creation. Each inode has an entry in inode table.

An inode provides following information about a file:

- File Types (i.e., regular, directory, device file or other)
- File Permission or access modes (i.e., read, write or execute to different users)
- User ID of the owner
- Group ID of the group
- Last access, last modification detail
- Number of soft links and hard links
- Size of the file
- Location or path of the file
- Other related data

We can find the inode number of a particular file using the following Linux command:

```
ls –i <filename>
```

When a program or a user is referred by its name, the operating system maps the filename with its corresponding inode. Then through the inode further operations in the file is performed.

5.6 FILE ATTRIBUTES

File attributes are used along with the computer files as settings. They are used to either allow or restrict certain operations for a user or for the operating system. Some of the attributes and its functions are described in Table 5.3.

Table 5.3: Different File Attributes and Their Functions

Attribute	Meaning	Functions
a	append only	This attribute permits a file to be added or appended only. It does not allow the removal of file.
c	Compressed	This attribute helps kernel to compress any data automatically written in a file and uncompress it when require.

d	no dump	It ensures that file is not backed up in backups where dump utility is used.
e	extent format	This attribute depicts that extent is used by file for mapping the blocks
i	Immutable	An immutable file cannot be removed, cannot be renamed and link can't be created
j	Journaling	Journaling option guarantees that a file will be written in journal first and then to the disk
s	source deletion	It tells that a particular file cannot be recovered once it has deleted.
t	no tail merging	It is a process in which small part of data which do not feel a complete block are merged with similar field of data from other files.
u	Undeletable	The contents of a deleted file is saved which allows a utility to recover the deleted file
A	no time updates	Access timestamp will not update at a time accessing the file
D	synchronous directory updates	It ensures that the changes made to the file are written immediately in the disk
S	synchronous updates	It writes the changes made in the file synchronously in the disk
T	top of directory hierarchy	A directory is deemed to be the top of directory hierarchies



CHECK YOUR PROGRESS

Q.15: Each entry of a directory files must have .....

a) A filename b) A inode number

c) Both filename and inode number

d) Either filename or inode numbercrdir

Q.16: File attribute 'a' stands for?

a) Add only b) Append only

c) Asynchronous mode d) No time updates

Q.17: An inode represents only .....

a) Data and files b) Only files

c) All processes d) Files and directories

Q.18: Which of the following file attribute option means the file is 'immutable'?

a) i b) m c) T d) U

Q.19: Any file's attribute information is stored in which structure on the disk?

a) inode b) Data blocks

c) File blocks d) Directory file

Q.20: Deleting a soft-link does which of the following?

a) Deletes the destination file.

b) Deletes both the softlink and the destination file

c) Deletes only the softlink

d) backup of the destination is automatically created



5.7 LET US SUM UP

- A file is a collection of related data stored in a computer.
- A file system is used to store and organize all the files in a Linux system.
- Ext2 file system runs e2fsck program for recover the file after any unclean shutdown.
- Ext3 is an extended version of ext2 file system.
- Journaling concept is introduced in ext3.
- A journal is used to correct the inconsistencies which can occur in case improper or unclean shutdown.
- Ext4 file system has some improvised features such as multiblock allocation, delayed allocation, no journaling mode, extents etc.
- ISO 9660 is a file system created for optical media.

- A regular file contains the entire text, binary, image, program and other files.
- Directory contains all the files and folders, sub folders.
- We can create, delete directories using *mkdir* and *rmdir* command respectively.
- *cd* command is used to change the current working directory of the system
- A special file or device file is located in */dev* directory of the file system
- Character device file and block device file are two types of special files.
- A socket and named pipe file is used for inter process communication.
- Symbolic link file is a reference to a file located in the disk.
- An inode or index node contains all the details about a file
- An inode of a file can be found using ***ls –i*** command
- File attributes are some setting option which allow or restrict specific operations on a file.



5.8 FURTHER READING

1) Das, S. (2000). *UNIX*: *Concepts and Applications*. Tata McGraw-Hill.
2) Bovet, D. P. & Cesati, M. (2005). *Understanding the Linux Kernel: from I/O ports to process management*. "O'Reilly Media, Inc."
3) Burgess, M. (2004). *Principles of Network & System Admin*. John Wiley & Sons.



5.9 ANSWER TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (c) **Ans. to Q. No. 2:** (d)
Ans. to Q. No. 3: (b) **Ans. to Q. No. 4:** (a)
Ans. to Q. No. 5: (d) **Ans. to Q. No. 6:** (a)
Ans. to Q. No. 7: (c) **Ans. to Q. No. 8:** (b)
Ans. to Q. No. 9: (a) **Ans. to Q. No. 10:** (b)

Ans. to Q. No. 11: (a) **Ans. to Q. No. 12:** (c)

Ans. to Q. No. 13: (a) **Ans. to Q. No. 14:** (d)

Ans. to Q. No. 15: (c) **Ans. to Q. No. 16:** (b)

Ans. to Q. No. 17: (d) **Ans. to Q. No. 18:** (a)

Ans. to Q. No. 19: (a) **Ans. to Q. No. 20:** (c)



5.10 MODEL QUESTIONS

Q.1: What is purpose of using file system in Linux?

Q.2: Name the different series of extended file system.

Q.3: Compare the features of ext2, ext3 and ext4 file system.

Q.4: What is the limitation of ext2 file system? How it is overcome by ext4 file system?

Q.5: What is journaling file system?

Q.6: Explain the features of ext4 file system.

Q.7: Explain any three commonly used file types in Linux.

Q.8: Differentiate between following:

a) Hard link vs. Soft link

b) Named pipe file vs. Socket files

c) Character files and Block files

Q.9: Explain the functions of inode in Linux file system.

Q.10: Define file attributes. List the different type of attributes used in Linux with their functions.

Q.11: Describe different types of file systems used in Linux.

Q.12: Write short notes on following file systems:

a) XFS

b) IFS

c) ReiserFS.

*** ***** ***

UNIT 6: FILE ORGANIZATION

UNIT STRUCTURE

6.1 Learning Objectives

6.2 Introduction

6.3 Absolute Pathnames

6.4 Relative Pathnames

6.5 Organization of File Tree

6.6 File System Mounting and Unmounting

6.6.1 *mount* Command

6.6.2 *umount* Command

6.7 Standard Directories and their Contents

6.8 Let Us Sum Up

6.9 Further Reading

6.10 Answers to Check Your Progress

6.11 Model Questions

6.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe how the files and directories are organized in Linux system
- move from one directory to another directory
- attach a device or a file system to the Linux system
- detach a device or a file system to the Linux system
- describe the standard directories and their functionalities.

6.2 INTRODUCTION

In the previous unit, you have already learnt about the Linux file system, and the different types of file systems used in Linux and their functions. You have also been acquainted with the basic types of files and the uses of those files. The previous unit has also provided the knowledge of the different file attributes. You also have got the concept of the directory in Linux system.

In this unit you will learn about the organization of the file tree in Linux system. You will know about the absolute and relative pathnames of the files and directories and how to move from one directory to another directory by using these pathnames. Through this unit, you will also learn how to attach and detach a device or a file system.

6.3 ABSOLUTE PATHNAMES

Each directory and file in Linux is indicated by its pathname begining from the *root* directory. i.e., the pathname specifies the location of a file or directory where it is found. *Absolute pathname* is the complete path of a directory or file starting from the root. For example, */usr/abc* is the absolute pathname of the directory *abc*. The figure *6.1* shows it. The first forward slash(/) indicates the root directory. Thus the pathname */usr/abc* means that the directory *abc* is found under the *usr* directory which is in turn found under the *root* directory. Similarly, the absolute pathname of the directory *abc3* is */usr/abc/abc3*. To move directly to a particularly directory or file you must have to use the absolute pathname. The following commands show the use of absolute pathname.

```
$ pwd
/usr
$ cd   /usr/abc/abc3/test1
$ pwd
/usr/abc/abc3/test1
$ cd   /usr/abc
$ pwd
/usr/abc
```

The *pwd* command is used to give the present working directory and the *cd* command is used to move from present working directory to another directory. The *pwd* command gives the absolute path of the current working directory.

6.4 RELATIVE PATHNAMES

Relative pathname represents the location of a directory or file with respect to the current working directory. For example, if your current directory is */usr/abc,* then you can move to a directory say *abc3* under the *abc* directory by using only the directory name *abc3* with *cd* command. It is performed in command line as–

```
$ pwd
/usr/abc
$ cd   abc3
$ pwd
/usr/abc/abc3
```

Here, in the command line *"cd abc3", abc3* has been used as a relative pathname. It can be understood using the figure 6.1. Similarly, if the current directory is */usr/abc/abc3*, then by using the relative pathname of a directory (i.e., only the directory name) resides under the directory *abc3* to move to that particular directory.

```
$ pwd
/usr/abc/abc3
$ cd   test1
$ pwd
/usr/abc/abc3/test1
```

Again, **" . ."** can be used as an argument to the *cd* command to the parent directory of the current working directory. Following the figure 6.1, the following command lines work as–

```
$ pwd
/usr/abc/abc3/test1
$ cd   . .
$ pwd
/usr/abc/abc3
$ pwd
```

```
/usr/abc/abc3
$ cd   . .
$ pwd
/usr/abc
```

The relative pathnames are useful, especially when the absolute pathnames are quite long. A relative pathname does not start with a forward slash(/). i.e., if a pathname does not contain a forward slash, then it means that it's a relative pathname.

6.5 ORGANIZATION OF FILE TREE

Linux file system is organized as a tree or hierarchy of directories and files. i.e., all files and sub-directories are stored within directories. The tree or hierarchy starts with the root directory. There are several system directories and the home directories are found under the root directory. The forward slash is used as a separator of directories and files, where the first forward slash represents the root directory. For example, a sub directory **abc** of the directory **usr** is shown as **/usr/abc.** Here, the first forward slash represents the root directory.

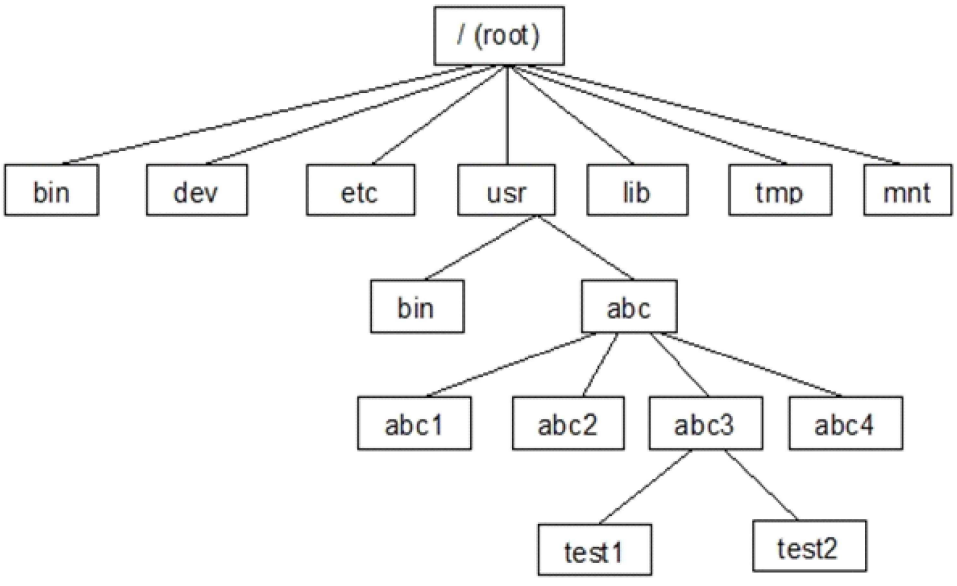


Figure 6.1: A Sample Structure of File System in Linux

In Linux, the **ls** command is used to display the files and sub directories under the current working directory. For example, from the figure 6.1.

```
$ pwd
/usr/abc

$ ls

abc1   abc2   abc3   abc4
```



CHECK YOUR PROGRESS

Q.1: What are the uses of absolute and relative pathnames.

.....

.....

.....

Q.2: The hierarchy of the Linux file system starts from ..... directory.

Q.3: From the figure 6.1, write the absolute paths to move to the directories **dev**, **abc**, **abc4** and **test1**

.....

.....

.....

.....

6.6 FILE SYSTEM MOUNTING AND UNMOUNTING

In Linux, once a file system is created, it is logically separated with its own tree structure and root directory. But during the time of booting, all these file systems are required to be united to become a single file system. Then the root file system become the main file system to which all the secondary file systems are attached.

File system mounting is the process of attaching all the secondary file systems (i.e., the file systems other than the root or main file system) to the main file system at different points. Here, the secondary files attach themselves to the file system. File system mounting is done by using the **mount** command. The file systems are mounted during the system start up.

File system *unmounting* is the process of detaching the secondary file systems from the main file system. The file systems are unmounted when the system is shut down. File system unmounting is not possible if there is a file open in it.

6.6.1 *mount* Command

The **mount** command is used to mount (or attach) a file system. The mount command usually has two arguments to achieve the mounting- one is the name of the file system and the other one is the directory name under which it is to be mounted.

To mount a file system at a specified location (mount point), the mount command in the following form is used.

```
$ mount    [OPTION...]    DEVICE_NAME    DIRECTORY
```

To mount a newly created file system, an empty directory, say **/new** must be first made available in the main file system (the **root** file system is the main file system). Then the *root* directory of the newly created file system, say **newroot** has to be mounted on the */new* directory. It is done by using the mount command as follows-

```
$ mount     -t    ext2    /dev/newroot       /new
```

Here, **ext2** is the type of the file system which is specified with the **-t** option.

The point (i.e. the location) at which the attachment is done is called the **mount point**. In the above example, */new* is the mount point. The mount is generally an empty directory. But sometimes it may contain files also. If the mount point contains files, then these files will not be seen when a new file system is mounted at the mount point. After unmounting the file system the files are seen again.

To mount a **CD-ROM,** the command will be as follows:

In Red Hat Linux,

```
$  mount  -t  iso9660  /dev/cdrom  /mnt/cdrom
```

In UnitedLinux:

```
$ mount  -t  iso9660  /dev/cdrom  /media/cdrom
```

In these examples, **/mnt/cdrom** and **/media/cdrom** are the mount point directories for the CD-ROM drive.

6.6.2 *umount* Command

Unmounting of file system is achieved by the **umount** command. The ***umount*** command needs either the file system name or the mount point as argument. For example, to unmount CD-ROM in Red Hat Linux, the command is as follows-

```
$  unmount  /dev/cdrom
```

or

```
$  unmount  /mnt/cdrom
```



CHECK YOUR PROGRESS

Q.4: What is file system mounting? Why is it needed in Linux system?

.....

.....

.....

.....

Q.5: What is mount point?

.....

.....

Q.6: Write the two ways of unmounting a file system.

.....

.....

.....

6.7 STANDARD DIRECTORIES AND THEIR CONTENTS

The standard directories are located directly under the root directory. These directories are essential to the startup and continuous operation of the system.

The following are the standard directories of the Linux file system.

/bin : It contains the programs needed for using and managing the system.

The commands found in this directory are date (displays today's date), ls (lists the contents of a directory), and cp (makes a copy of a file).

/boot : It contains the static files of the boot loader.

/dev : It contains system device files. A device file provides an interface to a particular device. Examples of devices having device files in /dev are disk drives, tape drives, or CDROM drives.

/etc : It contains system specific configuration files, and files essential for system startup.

/home : This is the directory where the home directories for all users of the system are stored.

/lib : Shared libraries and kernel modules are under this directory.

/mnt : This is the directory where temporary file systems are mounted. It may contain subdirectories like cdrom, floppy, and disk.

/opt : It contains software files that are not installed when the operating system is installed. This directory usually contains products provided by third-party software vendors.

/sbin : Programs for administering a system are located in this directory. The commands fdisk (used to partition a disk), fsck (used to check the integrity of a file system), and shutdown (used for stopping a system) are found under this directory.

/tmp : This directory is used to hold temporary files. This directory is generally referred to as a scratch directory, and can be used by all system users.

/usr : It contains programs and files related to the users of a system. The data in /usr is typically read-only, and may be shared with other computer systems on a network.

/var : Files with varying content are stored in this directory. It includes system log files, mail system files, and print spooling system files.

/srv : It contains data for services provided by the system.



6.8 LET US SUM UP

- Absolute pathname is the complete path starting from the root to a particular directory.
- Relative pathname is the reduced pathname which is relative to the current working directory.
- The file system of the Linux system is organized in a tree structure where the tree starts with the root directory
- File system mounting attaches the all the secondary file systems (i.e. the file systems other than the root or main file system) to the main file system.
- Unmounting og the file system detaches the file systems which are attached to the main file system at different points.
- The different standard directories of the Linux system are needed to star up and continuous operation of the system.



6.9 FURTHER READING

1) Das, S. (2000). *UNIX*: *Concepts and Applications*. Tata McGraw-Hill.

2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



6.10 ANSWER TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: The absolute pathname is used to move directly to a particular directory. But the relative pathname is used to move to a directory under the current working directory.

Ans. to Q. No. 2: root

Ans. to Q. No. 3: The absolute path to move to the **dev** directory is: /dev
The absolute path to move to the **abc** directory is: /usr/abc
The absolute path to move to the **abc4** directory is: /usr/abc/abc4
The absolute path to move to the **test1** directory is: /usr/abc/abc3/test1

Ans. to Q. No. 4: File system mounting is the process of attaching all the secondary file systems to the main file system at different points called mount point. File system mounting is needed because during the time of booting, all the file systems are need to be united to become a single file system.

Ans. to Q. No. 5: Mount point is a directory or location at which the mounting of a file system is done.

Ans. to Q. No. 6: The two ways of unmounting a file system are–

a) By suing the file system name as argument
b) By using the mount point as argument.



6.11 MODEL QUESTIONS

Q.1: What are absolute and relative pathnames? Differentiate between them through examples.

Q.2: Briefly describe the file tree organization of the Linux file system.

Q.3: How is a file system mounted by using the mount command in Linux system? Explain with example

Q.4: Describe unmounting process to detach a file system in Linux system.

Q.5: Describe the standard directories of the Linux system.

*** ***** ***

UNIT 7: LINUX FILE HANDLING COMMANDS

UNIT STRUCTURE

7.1 Learning Objectives

7.2 Introduction

7.3 Files and Directory Handling Commands

- 7.3.1 pwd Command
- 7.3.2 ls Command
- 7.3.3 cd Command
- 7.3.4 cp Command
- 7.3.5 mv Command
- 7.3.6 rm Command
- 7.3.7 mkdir Command
- 7.3.8 rmdir Command

7.4 Commands for Creating and Viewing Ordinary Files

- 7.4.1 cat Command
- 7.4.2 more Command
- 7.4.3 pg Command

7.5 Let Us Sum Up

7.6 Further Reading

7.7 Answers to Check Your Progress

7.8 Model Questions

7.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe different file handling commands
- copy and move files and directories
- remove files and directories
- create and view files
- use pg command to view files as per requirement.

7.2 INTRODUCTION

In the previous unit, we have learned about absolute and relative path names, how to mount and unmount file systems. We also learned about standard directory and their contents. In this unit, we will explore the various file and directory handling commands used in Linux. It gives details the various commands for navigating the filesystem, creation, deletion of files etc. In the next unit, we will explore the concept of LINUX filter commands like wc, head, tail etc.

7.3 FILE AND DIRECTORY HANDLING COMMANDS

This section describes some of the important file and directory handling commands used in Linux.

7.3.1 pwd Command

The **pwd command** displays the full absolute path to the current location in the file system.

For example, `$ pwd ↵`

/usr/bin

implies that /usr/bin is the current working directory.

7.3.2 ls Command

The **ls command** lists the contents of a directory. If no target directory is specified, then the contents of the current working directory are displayed.

So, if the current working directory is /,

```
$ls ↵
bin     dev   home  mnt   share  user  var
boot    etc   lib   proc  sbin   tmp   vol
```

However, ls command doesn't show you all the entries in the directory. Files and directories that begin with a dot (.) are hidden. The reason for this is that files that begin with a dot (.) usually contain

important configuration information and should not be changed under normal circumstances.

If you want to see all files, **ls** supports the –a option:

```
$ ls –a ↵
```

To see more detailed information, such as the size, type and ownership of the files, use the –l (long listing) option, which can be combined with the –a option as follows:

```
$ ls –a –l ↵
```

Or

```
$ ls –al ↵
```

```
–rwxr-xr-x  1  jsmith  staff  43     Mar 23 18:14  prog1
–rw-r—r—    1  jsmith  staff  10030  Mar 22 20:41  sample.f
drwxr-sr-x  2  jsmith  staff  512    Mar 23 18:07  subdir1
```

Each line of the output looks like this:

type	permissions	links	owner	group	size	date	name
–	rw-r—r—	1	jsmith	staff	10030	Mar 22 20:41	sample.f
d	rwxr-sr-x	2	jsmith	staff	512	Mar 23 18:07	subdir1

Where:

- **type** is a single character which is either:
  d (directory)
  – (ordinary file)
  | (symbolic link)
  b (block-oriented device)
  c (character-oriented device)
- **permissions** is a set of characters describing access rights. There are 9 permission characters, describing 3 access types given to 3 user categories. The three access types are read (‘r’), write (‘w’) and execute (‘x’) and the three user categories are the user who owns the file, users in the group that the file belongs to and

other users (general public). An 'r', 'w', or 'x' character means the corresponding permission is present; a '–' means it is absent.

- **links** refers to the number of the filesystem links pointing to the file or directory.
- **owner** is usually the owner who created the file or directory.
- **group** denotes a collection of users who are allowed to access the file according to the group access rights specified in the permissions field.
- **size** is the length of the file, or the number of bytes used by the operating system to store the list of files in a directory.
- **date** is the date when the file or directory was last modified (written to). The –u option displays the time when the file was last accessed (read).
- **name** is the name of the file or the directory.

ls Command Options: Some main options of ls command are given below:

Option	Description
ls –a	list all files including hidden file starting with '.'
ls –d	list directories – with '*/'
ls –F	add one char of */=>@| to enteries
ls –i	list file's inode index number
ls –l	list with long format - show permissions
ls –la	list long format including hidden files
ls –lh	list long format with readable file size
ls –ls	list with long format with file size
ls –r	list in reverse order
ls –R	list recursively directory tree
ls –s	list file size
ls –S	sort by file size
ls –t	sort by time & date
ls –X	sort by extension name

Example 7.1: ls Command Examples:

(You can press the tab button to auto complete the file or folder names.)

List directory Documents/Books with relative path:

```
$ ls Documents/Books
```

List directory/home/user/Documents/Books with absolute path.

```
$ ls /home/user/Documents/Books
```

List root directory:

```
$ ls /
```

List parent directory:

```
$ ls ..
```

List user's home directory (e.g., /home/user):

```
$ ls ~
```

List with long format:

```
$ ls -l
```

```
Show hidden files:
```

```
$ ls -a
```

List with long format and show hidden files:

```
$ ls -la
```

Sort by date/time:

```
$ ls -t
```

Sort by file size:

```
$ ls -S
```

List all subdirectories:

```
$ ls *
```

Recursive directory tree list:

```
$ ls -R
```

List only text files with wildcard:

```
$ ls *.txt
```

ls redirection to output file:

```
$ ls> out.txt
```

List directories only:

```
$ ls -d */
```

List files and directories with full path:

```
$ ls -d $PWD/*
```

7.3.3 cd Command

The **cd command** changes the current working directory. The syntax is:

```
$ cd<path> ↵
```

where <path> can be absolute or a relative path.

One of the most common relative path is **'..'** (the parent directory of the current directory).

Use without any target directory,

```
$ cd ↵
```

reset your current working directory to your home directory.

If you change into a directory and you subsequently want to return to your original directory, use,

```
$ cd ↵
```

7.3.4 cp Command

The **cp command** is used to make copies of files or entire directories.

To copy files, use:

```
$ cp source-file(s) destination
```

where *source-file(s)* and *destination* specify the source and the destination of the copy operation respectively.

The behavior of cp depends on whether the destination is a file or a directory.

- If the destination is a file, only one source file is allowed and cp makes a new file called *destination* that has the same contents as the source file.
- If the destination is a directory, several source files can be specified, each of which will be copied to the destination directory.

To copy entire directories, including their contents, use a *recursive copy*:

```
$ cp –rd source-directories destination-directory
```

7.3.5 mv Command

The **mv command** is used to rename files/directories and/or move them from one directory into another. Exactly one source and one destination must be specified:

```
$ mv source destination
```

The behavior of the mv command depends on:

- If *destination* is an existing directory, the new name for *source* (whether it is a file or a directory) will be *destination/source.*
- If *source* and *destination* are files, *source*is renamed as *destination.*
- If *destination* is an existing file, it will be destroyed and overwritten by *source*. (You can use the –I option if you want to be asked for confirmation before a file is overwritten in this way.)

7.3.6 rm Command

The **rm command** removes the specified files.

The syntax is:

```
$ rm target-file(s)
```

Unlike other operating systems, it is very difficult to recover a deleted file unless you have a backup, since there is no recycle bin. So use this command with care.

If you would like to be asked before files are deleted, use the –i option

```
$ rm –i myfile↵
rm:  remove 'myfile'?
```

rm can also be used to delete directories (along with all of their contents, including any subdirectories they contain). To do this, use the –r option. To avoid rm from asking any questions or giving errors (e.g. if the file does not exist) you can use the –f (force) option.

7.3.7 mkdir Command

The syntax of the mkdir command is:

```
$ mkdir directory ↵
```

The mkdir command creates a subdirectory called *directory* in the current working directory.

You can only create subdirectories in a directory if you have write permission on that directory.

To create multiple directories, pass the names of the directories to be created to the mkdir command. The names of the directories should be separated by space.

Example 7.2: mkdir Examples

```
$ mkdir dir1 dir2 dir3 ↵
```

It will create three directories dir1, dir2 and dir3 in the current working directory.

The –p option creates parent directories as necessary. When this option is specified, no error is reported if a directory already exists. For example,

```
$ mkdir -p /home/hope/Documents/pdf ↵
```

creates the directory /home/hope/Documents/pdf. If any of the parent directories /home, /home/hope, or /home/hope/Documents do not already exist, they will automatically be created.

7.3.8 rmdir Command

The **rmdir** command removes the specified subdirectory from the current working directory.

```
$ rmdir directory
```

You can only remove subdirectories if they are completely empty.



CHECK YOUR PROGRESS

Q.1: Give a command to copy file 'a' to file 'b'.

.....

Q.2: Give a single command to create a directory /abc/b.

.....

Q.3: How will you list all the files within a directory including hidden files?

.....

Q.4: How will you list the contents of a directory in long format listing along with showing hidden files?

.....

.....

7.4 COMMANDS FOR CREATING AND VIEWING ORDINARY FILES

7.4.1 cat Command

The syntax of the **cat** command is:

```
$ cat target-file(s)
```

The cat command displays the contents of the *target-file(s)* on the screen, one after the other.

You can also use it to create files from the keyboard input as follows:

```
$ cat>hello.txt ↵    (hello.txt file is created)
Hello world ↵        (Contents are entered)
[ctrl-d]             (Save and Exit command)
$ ls hello.txt ↵
Hello.txt
$ cat hello.txt
Hello world
$
```

7.4.2 more Command

The syntax of the **more** command is:

```
$ more target-file(s)
```

The more command displays the contents of the *target-file(s)* on the screen, pausing at the end of each screen full and asking the user to press a key. This is useful for long files.

If also incorporates a searching facility (press '/' and the type a phrase that you want to look for). You can also use more command to break up the output of commands that produce more than one screen full of output as follows:

```
$ ls –l | more ↵
```

7.4.3 pg Command

The **pg** command is a pager. It allows you to view text files one page at a time. The **pg** command displays a text file on a CRT one screen full at once. After each page, a prompt is displayed. The user may then either press the newline key to view the next page or one of the keys described below.

If no file name is given on the command line, **pg** reads from standard input. If standard output is not a terminal, **pg** acts like cat but precedes each file with its name if there is more than one.

If input comes from a pipe, **pg** stores the data in a buffer file while reading to make navigation possible.

The syntax of the pg command is:

```
pg [-number] [-p string] [-cefnrs] [+line] [+/
pattern/] [file...]
```

The options used in pg command are:

–number	The number of lines per page. Usually, this is the number of CRT lines minus one.
–c	Clear the screen before a page is displayed, if the **terminfo** entry for the terminal provides this capability.
–e	Do not pause and display “(EOF)” at the end of a file.
–f	Do not split long lines.
–n	Without this option, commands must be terminated by a newline character. With this option, **pg** advances once a command letter is entered.
–p string	Instead of the normal prompt “**:**”, *string* is displayed. If string contains “**%d**”, its first occurrence is replaced by the number of the current page.
–r	Disallow the shell escape.
–s	Print messages in standout mode, if the **term info** entry for the terminal provides this capability.
+number	Start at the given line.
+/pattern**/**	Start at the line containing the Basic Regular Expression pattern given.

The following commands may be entered at the prompt. Commands preceded by i accept a number as argument, positive or negative. If this argument starts with + or -, it is interpreted relative to the current position in the input file, otherwise relative to the beginning.

i<newline>	Display the next or the indicated page.
*i***d**, **^D**	Display the next half page. If **i** is given, it is always interpreted relative to the current position.
i**l**	Display the next or the indicated line.
i**f**	Skip a page forward, *i* must be a positive number and is always interpreted relative to the current position.
*i***w**, *i***z**	Behave as <newline> except that *i* becomes the new page size.
., ^L	Redraw the screen.
$ i*/*pattern*/*	Advance to the last line of the input file. Search forward until the first or the *i*-th occurrence of the Basic Regular Expression *pattern* is found. The search starts after the current page and stops at the end of the file. No wrap-around is performed, *i* must be a positive number.
i**?**pattern**?**, i**^**pattern**^**	Search backward until the first or the *i*-th occurrence of the Basic Regular Expression *pattern* is found. The search starts before the current page and stops at the beginning of the file. No wrap-around is performed, *i* must be a positive number.

The search commands accept an added letter. If **t** is given, the line containing the pattern is displayed at the top of the screen, which is the default. **m** selects the middle and **b** the bottom of the screen. The selected position is used in following searches, too.

i**n**	Advance to the next file or *i* files forward.
i**p**	Reread the previous file or *i* files backward.
s file name	Save the current file to the given *file name*.
h	Display a command summary.
!command	Execute *command* using the shell.
q, Q	Quit.

If the user presses the interrupt or quit key while pg reads from the input file or writes on the terminal, **pg** will immediately display the prompt. In all other situations these keys will terminate **pg**.



CHECK YOUR PROGRESS

Q.5: What will happen if we use pg command with –c and –n option?

.....

.....

.....

Q.6: ..... command is used for displaying contents of a file.

Q.7: Apart from displaying file contents, cat command is also used for ..... of files.

Q.8: Multiple arguments as file name can be specified in cat command. (TRUE/FALSE)



7.5 LET US SUM UP

- The pwd command displays the full absolute path to the current location in the filesystem.
- The ls lists the contents of a directory. If no target directory is specified, then the contents of the current working directory are displayed. Various options are available with the ls command to list the files as per your requirement.
- The cd command is used to change the current working directory.
- Files and entire directories can be copied using the cp command.
- To rename or move files and directories the mv command is used with one source and one destination.
- Files can be deleted using the rm command.
- The mkdir command creates one or more directories.
- Directories can be removed using rmdir command.
- The cat command can be used to create and display files.
- Longer files can be displayed one screen at a time by using more or pg command.



7.6 FURTHER READING

1) Das, S. (2000). *UNIX*: *Concepts and Applications*. Tata McGraw-Hill.

2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



7.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: $ cp a b ↵

Ans. to Q. No. 2: $ mkdir–p /abc/b ↵

Ans. to Q. No. 3: $ ls –a ↵

Ans. to Q. No. 4: We need to use option –a and –l together with ls command.

$ ls –a –l ↵

Or

$ ls –al ↵

Ans. to Q. No. 5: When –c option is used it clears the screen before a page is displayed.

When –n option is used, pg advances once a command letter is entered. Without this option, a newline character must terminate commands.

Ans. to Q. No. 6: cat

Ans. to Q. No. 7: creation

Ans. to Q. No. 8: TRUE



7.8 MODEL QUESTIONS

Q.1: Explain the use of the various options used with **ls** command.

Q.2: What is the significance of the **cat** command?

Q.3: What is **cd** command?

Q.4: How does the **cp** command works in Linux?

Q.5: Compare between **rm** and **rmdir** command.

Q.6: With proper syntax explain the **mv** command.

Q.7: What is the use of **mkdir** command?

Q.8: Why do we use the **more** command in Linux?

Q.9: Explain the **pg** command with its options.

*** ***** ***

UNIT 8: LINUX FILTER COMMANDS

UNIT STRUCTURE

8.1 Learning Objectives

8.2 Introduction

8.3 Filter Commands

- 8.3.1 wc Command
- 8.3.2 head Command
- 8.3.3 tail Command
- 8.3.4 cut Command
- 8.3.5 tr Command
- 8.3.6 grep Command (with Regular Expressions)

8.4 Setting User and Group Ownership of Files and Access Permissions

- 8.4.1 chmod Command
- 8.4.2 chown Command
- 8.4.3 chgrp Command

8.5 Let Us Sum Up

8.6 Further Reading

8.7 Answers to Check Your Progress

8.8 Model Questions

8.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- describe filters in Linux
- count the number of words, characters and lines in a file
- list the contents of a file from the front or the back
- use the cut command to view only certain fields in a file
- use the various options used with tr command
- define regular expressions and use them with the grep command
- describe the different access permissions and permission categories
- alter the permissions of a file.

8.2 INTRODUCTION

In the previous unit, we have learned how to copy, move and remove files and directories. We have also learnt how to create and view files. We have also explored the various uses of pg command. In this unit, we will explore various filter commands, such as wc, head, tail, cut, tr and grep. We will also learn how to set user and group ownership of files in this unit. This is the last unit of this block.

8.3 FILTER COMMANDS

A filter, in the context of the Linux command line, is a program that accepts textual data and then transforms it in a particular way. Filters are a way to take raw data, either produced by another program, or stored in a file, and manipulate it to be displayed in a way more suited to what we are after.

These filters often have various command line options that will modify their behavior; so it is always good to check out the man page for a filter to see what is available.

8.3.1 wc Command

The **wc** command stands for word count and it does just that (as well as characters and lines). By default, it will give a count of all three but using command line options we may limit it to just what we want.

The syntax is:

```
$ wc [-options] [path]
```

Example of wc command:

```
$ wc myfile.txt
12 36 195 myfile.txt
```

This output shows that the file myfile.txt has 12 lines, 36 words and 195 characters.

Sometimes you just want one of these values. –l will give us lines only, –w will give us words and -m will give us characters.

8.3.2 head Command

The **head** command is a program that prints the first so many lines of its input. By default, it will print the first 10 lines but we may modify this with a command line argument.
The syntax is:

```
$ head [-number of lines to print] [path]
```

8.3.3 tail Command

The **tail** command is the opposite of head. Tail is a program that prints the last so many lines of it's input. By default it will print the last 10 lines but we may modify this with a command line argument.
The syntax is:

```
$ tail [-number of lines to print] [path]
```

8.3.4 cut Command

The **cut** command is a nice little program to use if your content is separated into fields (columns) and you only want certain fields.
The syntax is:

```
$ cut [-options] [path]
```

For example, there is a file that has data in 3 columns, the first is a name, the second is a fruit and the third an amount.

Let's say we only wanted the first column.

```
$ cut -f 1 -d ' ' mysampledata.txt ↵
```

Fred
Susy
Mark
Robert
Terry
Lisa
Susy

Mark
Anne
Greg
Oliver
Betty

The **cut** command defaults to using the TAB character as a separator to identify fields. In our file, we have used a single space instead, so we need to tell cut to use that instead. The separator character may be anything you like, for instance in a CSV file the separator is typically a comma (,). This is what the -d option does (we include the space within single quotes so it knows this is part of the argument). The -f option allows us to specify which field or fields we would like. If we want two or more fields then we separate them with a comma as below.

```
$ cut -f 1,2 -d ' ' mysampledata.txt ↵
```

Fred apples
Susy oranges
Mark watermellons
Robert pears
Terry oranges
Lisa peaches
Susy oranges
Mark grapes
Anne mangoes
Greg pineapples
Oliver rockmellons
Betty limes

8.3.5 tr Command

The **tr** command is a command line utility for translating or deleting characters. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting

specific characters and basic find and replace. It can be used with pipes to support more complex translation. The **tr** stands for translate. The syntax of this command is:

```
$ tr [OPTION] SET1 [SET2]
```

The available options are:

–c : complements the set of characters in string i.e., operations apply to characters not in the given set

–d : delete characters in the first set from the output.

–s : replaces repeated characters listed in the set1 with single occurrence

–t : truncates set1

Example 8.1: Convert Lower Case to Upper Case

To convert from lower case to upper case the predefined sets in tr can be used.

```
$ cat greekfile
WELCOME TO
GeeksforGeeks

$ cat greekfile | tr “[a-z]” “[A-Z]”
WELCOME TO
GEEKSFORGEEKS

or

$ cat geekfile | tr “[:lower:]” “[:upper:]”
WELCOME TO
GEEKSFORGEEKS
```

Example 8.2: Translate White-Space to Tabs

The following command will translate all the white-space to tabs:

```
$ echo “Welcome To GeeksforGeeks” | tr [:space:] ‘\t’
Welcome    To    GeeksforGeeks
```

Example 8.3: Translate Braces into Parenthesis

You can also translate from and to a file. In this example we will translate braces in a file with parenthesis.

```
$ cat greekfile
{WELCOME TO}
GeeksforGeeks

$ tr '{}' '()'   newfile.txt
(WELCOME TO)
GeeksforGeeks
```

The above command will read each character from "geekfile.txt", translate if it is a brace, and write the output in "newfile.txt".

Example 8.4: Using Squeeze Repetition of Characters Using –s

To squeeze repeat occurrences of characters specified in a set use the -s option. This removes repeated instances of a character.

We can also say that, you can convert multiple continuous spaces with a single space.

```
$ echo "Welcome To GeeksforGeeks" | tr -s [:space:]
''Welcome To GeeksforGeeks''
```

Example 8.5: Delete Specified Characters Using –d Option

To delete specific characters use the -d option.This option deletes characters in the first set specified.

```
$ echo "Welcome ToGeeksforGeeks" | tr -d 'w'
elcome To GeeksforGeeks
```

Example 8.6: Remove all the Digits from the String, Use

```
$ echo "my ID is 73535" | tr -d [:digit:]
my ID is
```

Example 8.7: Complement the Sets Using –c Option

You can complement the SET1 using -c option. For example, to remove all characters except digits, you can use the following.

$ echo “my ID is 73535” | tr -cd [:digit:]

73535

Or

$ echo “unix” | tr –c “u” “a”

Uaaa

8.3.6 grep Command (with Regular Expressions)

Regular expressions are similar to the wildcards. They allow us to create a pattern. They are a bit more powerful however. Regular expressions are typically used to identify and manipulate specific pieces of data. e.g., We may wish to identify every line which contains an email address or a url in a set of data.

grep is a program which will search a given set of data and print every line which contains a given pattern. It has many command line options which modify its behaviour so it is worth checking out its man page.

E.g., the –v option tells grep to instead print every line which does not match the pattern.

The syntax of the grep command is:

```
grep [command line options] <pattern> [path]
```

For example, let us assume we have a sample file ‘mysampledata.txt’.

```
$ cat mysampledata.txt ↵
```

Fred apples 20

Susy oranges 5

Mark watermellons 12

Robert pears 4

Terry oranges 9

Lisa peaches 7

Susy oranges 12

Mark grapes 39

Anne mangoes 7

Greg pineapples 3

Oliver rockmellons 2

Betty limes 14

Let's say we wished to identify every line which contained the string mellon

```
$ grep 'mellon' mysampledata.txt ↵
```

Mark watermellons 12

Oliver rockmellons 2

The basic behavior of grep is that it will print the entire line for every line which contains a string of characters matching the given pattern. This is important to note, we are not searching for a word but a string of characters.

Also, note that we included the pattern within quotes. This is not always required but it is safer to get in the habit of always using them. They are required if your pattern contains characters which have a special meaning on the command line.

Sometimes we want to know not only which lines matched but their line number as well.

```
$ grep -n 'mellon' mysampledata.txt ↵
```

3:Markwatermellons 12

11:Oliverrockmellons 2

Or maybe we are not interested in seeing the matched lines but wish to know how many lines did match.

```
$ grep -c 'mellon' mysampledata.txt ↵
```

2

The basic building blocks of regular expressions are:

- **.** (dot) –a single character.
- **?** –the preceding character matches 0 or 1 times only.
- ***** –the preceding character matches 0 or more times.

- **+** –the preceding character matches 1 or more times.
- **{n}** –the preceding character matches exactly n times.
- **{n,m}** –the preceding character matches at least n times and not more than m times.
- **[agd]** –the character is one of those included within the square brackets.
- **[^agd]** –the character is not one of those included within the square brackets.
- **[c-f]** –the dash within the square brackets operates as a range. In this case it means either the letters c, d, e or f.
- **()** –allows us to group several characters to behave as one.
- **|** (pipe symbol) –the logical OR operation.
- **^** –matches the beginning of the line.
- **$** –matches the end of the line.

Example 8.8: Examples of grep Command with Regular Expressions

Let's say we wish to identify any line with two or more vowels in a row. In the example below the multiplier {2,} applies to the preceding item which is the range.

```
$ grep '[aeiou]{2,}' mysampledata.txt ↵
```

Robert pears 4
Lisa peaches 7
Anne mangoes 7
Greg pineapples 3

How about any line with a 2 on it which is not the end of the line. In this example the multiplier + applies to the "." which is any character.

```
$ grep '2.+' mysampledata.txt ↵
```

Fred apples 20
The number 2 as the last character on the line.

```
$ grep '2$' mysampledata.txt ↵
```

Mark watermellons 12

Susy oranges 12

Oliver rockmellons 2

And now, each line which contains either ‘is’ or ‘go’ or ‘or’.

```
$ grep 'or|is|go' mysampledata.txt ↵
```

Susy oranges 5

Terry oranges 9

Lisa peaches 7

Susy oranges 12

Anne mangoes 7

Maybe we wish to see orders for everyone whose name begins with A - K.

```
$ grep '^[A-K]' mysampledata.txt ↵
```

Fred apples 20

Anne mangoes 7

Greg pineapples 3

Betty limes 14



CHECK YOUR PROGRESS

Q.1: Which of the following commands is used to count number of lines in a file?

a) wc –l b) wc –c

c) wc –w d) None of these.

Q.2: ..... will allow the user to search contents of a file for a particular pattern.

Q.3: ..... command is used for displaying the beginning of a file.

Q.4: To extract specific columns from a file, ... command is used.

Q.5: Give a command to display the last ten lines of myfile on the terminal.

.....

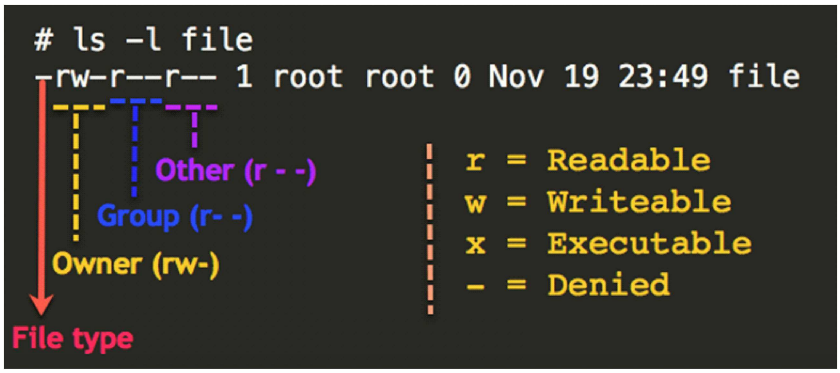
8.4 SETTING USER AND GROUP OWNERSHIP OF FILES AND ACCESS PERMISSIONS

The Linux files access is controlled. There are three types of access (permissions):

- read
- write
- execute

To view the permissions for files and directories, use the ls -l or ls –n commands.

```
$ ls -l file ↵
-rw-r—r— 1 root root 0 Nov 19 23:49 file
```



The first field of information displayed by the ls –l command is the file type. The file type typically specifies whether it is a file or a directory. A file is represented by a hyphen (–). A directory is represented by the letter d.

The remaining fields represent the permission groups: **owner, group,** and **other**.

Permission Groups: There are three permissions groups:

- Owner
- Group
- Other

The table describes the permission groups and their scope:

Permission	Description
Owner	Permissions used by the assigned owner of the file or directory.
Group	Permissions used by members of the group that owns the file or directory.
Other	Permissions used by all users other than the file owner, and members of the group that owns the file or the directory.

Permission Set:

- Each permission group has three permissions, called a permission set.
- Each set consists of **read**, **write**, and **execute** permissions.
- Each file or directory has three permission sets for the three types of permission groups.
- The first permission set represents the owner permissions, the second set represents the group permissions, and the last set represents the other permissions.
- The read, write, and execute permissions are represented by the characters **r**, **w**, and **x**, respectively.
- The presence of any of these characters, such as r, indicates that the particular permission is granted.
- A dash (**–**) symbol in place of a character in a permission set indicates that a particular permission is denied.
- Linux assigns initial permissions automatically when a new file or directory is created.

Permission	Access for a file	Access for a directory
Read (r)	You can display file contents and copy the file.	You can list the directory contents with the ls command.
Write (w)	You can modify the file contents.	You can modify the contents of a directory, such as by deleting a file. You must also have the execute permission for this to happen.
Execute (x)	You can execute the file if it is an executable. You can execute a shell script if you also have read and execute permissions.	You can use the cd command to access the directory. If you also have read access, you can run the ls -l command on the directory to list contents. If you do not have read access, you can run the ls command as long as you know the file name.

File Ownership: Each file belongs to a specific user and group (ownership). Access to the files is controlled by *user, group,* and what is called *other/everyone* permission bits and is usually set using a numerical value.

Each number represents the access level and it can be from 0 to 7.

Different access levels depending on the numbers:

- 0 – no access to the file whatsoever
- 1 – execute permissions only
- 2 – write permissions only
- 3 – write and execute permissions
- 4 – read permissions only
- 5 – read and execute permissions
- 6 – read and write permissions
- 7 – read, write and execute permissions (full permissions)

8.4.1 chmod Command

The **chmod** command is used to alter the permissions of a file. It may be used to add or remove permissions symbolically. For example, to add execute permissions for the owner of a file you would run:

```
$ chmod u+x filename
```

Or, to add read and write permissions for the group that owns the file, you would run:

```
$ chmod g+rw filename
```

Instead of adding permissions, the symbolic syntax of chmod can also be used to subtract or set to some absolute value as shown in these examples:

```
$ chmod o-w filename
$ chmod u=rwx,g=rx,o= filename
```

The chmod command can also explicitly set permissions using a numerical representation. For example, to set permissions on a file to rwxrwxr–, you would run:

```
$ chmod 774 filename
```

In addition to the standard read, write, and execute permissions, chmod can also set special permissions. These are the setuid bit, the setgid bit, and the sticky bit. The following examples show setting each of these special permissions along with brief descriptions of the effect of those permissions.

```
$ chmod u+s filename
```

Adds the setuid bit so that, if executable, this file will execute with the permissions of its owner.

```
$ chmod g+s filename
```

Adds the set gid bit so that, if executable, this file will execute with the permissions of its group. When this is set on a directory, all files created in the directory will have the same group as the directory.

```
$ chmod o+t directory_name
```

Adds the sticky bit so that users can only delete files from this directory that they created.

8.4.2 chown Command

Every file is owned by a specific user (or UID) and a specific group (or GID). The chown command can be used to change just the user, or the user and group of a file. Here is an example of changing the owner of file test to user and its group to user.

```
$ ls -l test
-rw-r—r— 1 root root 0 Nov 20 00:43 test

$ chown user:user test
$ ls -l test
-rw-r—r— 1 user user 0 Nov 20 00:43 test
```

The basic format for the chown command is as follows:

```
$ chown user:group filename
```

A period (.) can be used in place of the colon (:) separator character. Also, either the user or group name can be omitted. If the username is omitted (but the separator character is present), then the chown command behaves like the chgrp command, and only the group ownership is changed. If the group name is omitted but the separator character is present, then the group will be set to the login group of the specified user. If both the group name and the separator character are omitted, then only the user name is changed. For example, to change only the owner you could run the following:

```
$ chown user filename
```

8.4.3 chgrp Command

An alternate command to change only the group of a file is the chgrp command. For example:

```
$ chgrp group filename
```

The chgrp command is commonly used by normal users to change the group ownership of their files. The chown command is normally used only by the root user.

Each user can be a member of many groups (listed in the /etc/group file under several groups). Only one group will be a user's primary group (listed in the user's entry in /etc/password). When a user creates a file, by default the file will be owned by the user's primary group. If they want the file to be owned by one of their other groups, they must use the chgrp command to modify the group membership. For example,

```
$ id
```

uid=1001(user) gid=1001(user) groups=1001(user),10(wheel)

```
$ touch file
$ ls -lrt file
```

-rw-rw-r— 1 user user 0 Nov 20 00:52 file

As shown above the newly created file (file) automatically gets the group as "user".



CHECK YOUR PROGRESS

Q.6: ..... command changes a file's group owner.

Q.7: Which command is used to change permission of files and directories?

.....

Q.8: What is file permission: 764 with chmod command?

.....

.....



8.5 LET US SUM UP

- A filter, in the context of the Linux command line, is a program that accepts textual data and then transforms it in a particular way.
- Filters are a way to take raw data, either produced by another program, or stored in a file, and manipulate it to be displayed in a way more suited to what we are after.
- The wc command gives the word, character and lines count of a file.
- The head and tail command prints the specified number of lines in a file from the top or the bottom respectively.
- The cut command can be used if the data is separated into fields or columns.
- The tr command is a command line utility for translating or deleting characters. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters etc.
- Regular expressions are similar to the wildcards. They allow us to create a pattern.
- There are three types of access permissions in Linux - read, write and execute.
- There are three permission groups in Linux - owner, group and other.
- The chmod command is used to alter the permissions of a file.
- The chown command can be used to change just the user, or the user and group of a file.
- An alternate command to change only the group of a file is the chgrp command.



8.6 FURTHER READING

1) Das, S. (2012). *Your UNIX*/Linux: *The Ultimate Guide*. (Third Edition). (Science/Engineering/Math). McGraw-Hill.

2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



8.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (a) wc -l

Ans. to Q. No. 2: grep

Ans. to Q. No. 3: head

Ans. to Q. No. 4: cut

Ans. to Q. No. 5: $tail –10 myfile↵

Ans. to Q. No. 6: chgrp

Ans. to Q. No. 7: chmod

Ans. to Q. No. 8: Everyone can read, group including owner can write, owner alone can execute that particular file.



8.8 MODEL QUESTIONS

Q.1: What is the use of **chgrp** command?

Q.2: Explain how we use **chmod** command to change permission for files.

Q.3: With proper syntax, explain the use of **chown** command.

Q.4: Describe the various options used with the **cut** command.

Q.5: Explain the significance of **–n, –c** with **grep** command.

Q.6: Describe the various options available with the cut command.

Q.7: How is the **head** and **tail** command used to display various parts of a file?

Q.8: Explain the use of **tr** utilities.

*** ***** ***

Name of the Paper: .....
Course Code: .....

Centre for Internal Quality Assurance (CIQA)
Krishna Kanta Handiqui State Open University
City Office: Housefed Complex, Guwahati-781006

Learner's Feedback on Course

Dear Learner,

Regarding the course as mentioned above, we would like to know your opinions and comments so as to improve the quality of self learning materials in future. Please respond to the following statements by ticking the number you feel most reflect your opinion. After completion of the additional comments, please detach the page and send/mail the same to us at the address given below.

The Director, Centre for Internal Quality Assurance, KKHandiqui State Open University
Housefed Complex, Dispur, Guwahati-781006
(E-mail id: ciqa@kkhsou.in)

1) Approximately how many hours did you spend for studying the units in the course? ☐☐☐

2) Please give your opinions (by ✓ mark) to the following items based on your reading of the block:

Sl. No.	Statements	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
I)	The SLMs of the course fulfil the learning objectives set out in the beginning of each unit					
II)	The units of the SLM could satisfy my academic needs and expectations					
III)	The Unit writers have excellent knowledge about the course contents					
IV)	Language and contents of the units were presented at a level which I could readily understand					
V)	Ample opportunity for participation in the activities provided in the units					
VI)	Used enough Illustrations (Diagrams, tables etc.) for conceptual clarity					
VII)	Quality of content is engaging, relevant, and up-to-date					
VIII)	The self check questions are very helpful					
IX)	The Possible/model questions and the answers to check my progress have benefited me a lot					

Additional Comments: *(Please feel free to provide your open comments)*

1) Which aspects of the SLM, according to you, worked well?

.....

.....

2) What sort of changes/improvements do you feel KKHSOU could implement to improve the overall quality of the SLM?

.....

.....

Thank you for taking the time to complete this form.