PG CA S2-04 (DSE)

TWO YEAR MASTER'S DEGREE COURSE

(CBCS in Accordance with NEP 2020)

SEMESTER - II

LINUX SYSTEM ADMINISTRATION

Hiranya Chandra Bhuyan School of Science and Technology
KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Anjana Kakoti Mahanta, Gauhati University

Prof.Hiren Kumar Deva Sarma, Gauhati University

Dr. Lipi B Mahanta, IASST, Guwahati

Dr. Sanjib Kr. Kalita,Gauhati University

Dr. Tapashi Kashyap Das, KKHSOU

Dr. Sruti Sruba Bharali, KKHSOU

Course Co-ordinators: **Dr. Ridip Dev Choudhury,** KKHSOU

SLM Preparation Team

Units	**Authors**
1,2,3,4,5	Dr. Gautam Chakraborty, NERIM
6	Mr. SaurabhSutradhar, Royal Global University
7,8	Dr. Ridip Dev Choudhury, KKHSOU
9,10,11,15	Dr. Naba Jyoti Sarmah, Nalbari Commerce College
12,13,14	Mr. Saurabh Sutradhar, Royal Global University

Editors

Content: Dr. KhurshidAlamBorbora, KKHSOU (Unit 1,2,3,4,5,6,7,8)
Dr. Gautam Chakraborty, NERIM (Unit 9,10,11,12,13,14,15)

Structure,Format&Graphics: Dr. Ridip Dev Choudhury, KKHSOU
January, 2024

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State Open University is made available under a Creative Commons Attribution-Non Commercial-Share Alike 4.0 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 Campus: NH 37, Khanapara, Guwahati-781 022; Web: www.kkhsou.in

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

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

DETAILED SYLLABUS

CONTENTS

UNIT 1: **Introduction to System Administration** **7-18**

Introduction to SystemAdministration, Role of System Administrator, Qualities of good System Administrator, System Administration Common administrative tasks, identifying administrative files-configuration and log files

UNIT 2: **Introduction to LINUX Operating System** **19-35**

Basic idea on Proprietary, Open Source, Free Software etc, Basic Features of the Linux operating system, Introduction of Various Linux Distribution (Red Hat Enterprise Linux, Cent OS, Fedora Projects, Debian Linux, Ubuntu, SUSE etc.)

UNIT 3 : **Installation of LINUX Operating System** **36-65**

Installation Requirements, Partitioning the Hard drive in Linux, Installing the Linux system: Using Live CD, Virtual Machine, Direct Installation, Installing and Configuring software in Linux

UNIT 4 : **LINUX Kernel** **66-82**

Linux kernel and device drivers, System Start up and Shutdown, StandardI/O, Standard error, Redirection and Piping

UNIT 5: **Basics of LINUX File System** **83-106**

Basics of Linux file system- File system types (ext3, ext4, xfs, jfs, Reiser FS, iso 9660etc.), Bootblock, Superblock, I node table, Data blocks, How Linux access files, storage files, Linux standard directories, LILO, GRUB Boot Loader, three basic types of files (ordinary or regular, special or device and directory), I-nodes and file at tributes, init and run levels.

UNIT 6: **File organization** **107-123**

Files and File System (File Types and Permissions, Links, Size and Space, Date and Time); Working with Files: Reading Files, Searching for files, Copying, Moving, Renaming, Deleting, Linking, and Editing Files; Absolute and Relative path names, File system Mounting and Unmounting, Organization of the file tree, Standard directories and the ircontents, Disk related commands, checking disk free spaces.

UNIT 7: **LINUX File Handling Commands** **124-136**

Files and Directory handling Commands ls, cd, cp, mv, rm, mkdir, rmdir, pwd, Commands for Creating and Viewing ordinary files-cat, more, less

UNIT 8: **LINUX Filter Commands** **137-154**

Filter Commands-wc, head, tail, cut, tr, top, grep (with regular expressions),. Other commands-tar, unzip, nice, kill, netstat. Setting user and group ownership of files and Access permissions-chmod, chown, chgrp commands.

UNIT 9: **LINUX Shells** **155-193**

Various types of Shell available in Linux, Comparisons between various Shells, Study of different Linux Shells (sh, bash, csh, zsh), Environment variables, Shell script basics (examples of some simple shell programming), Shell programming in bash, read command, conditional and looping statements, case statements, parameter passing and arguments, Shell variables, system shell variables, shell keywords, creating Shell programs for performing various tasks.

UNIT 10: **Process** **194-217**

Services and Process, Basic commands for starting and stopping processes, Basic process attributes and their role in Access control. Examining the list of running processes on the system and understand the data presented there. Background process, Job control, Crontab file format, Backup and Restore procedure

UNIT 11: **Printing Jobs** **218-234**

Submit a print job, check the status of a print job, cancel a print job, configuring the Print Queue, Selecting the Print Driver, Editing the Printer configuration

UNIT 12: **System Administration Tasks** **235-266**

Getting Started: Login process, Creating Users Account and Group, Getting Help, Understanding the “root” account, Managing user accounts-adding, modifying and removing User accounts, changing permissions and ownerships, Creating and managing groups, modifying group attributes, becoming super user using su; limited su (sudo), Changing Password, System monitoring and logging, Monitoring memory usage, disk space usage and I/O activity, Temporary disable user’s accounts, creating and mounting file system, checking and monitoring system performance file

security & Permissions, Getting system information with uname, host name, disk partitions & sizes, users, kernel. Backup and restore files, reconfiguration hardware with kudzu, installing and removing packages in Linux. Configure X-windows starting & using X desktop. KDE & Gnome graphical interfaces, changing X windows settings

UNIT 13: **Networking in LINUX** **267-307**

Installation and configuration of: Proxy server(Squid), DNS server(BIND), Mail server, Web server(Apache), File server(Samba), DHCP server; Installation and configuration of a SSH server and client; Installation and configuration of FTP server and client, rules governing IP address classes and netmasks, Network Address, Netmask and Gateway configuring Interface with ifconfig, ping, netstat, traceroute, telnet, understanding the significance of the /etc/ services file and well known port numbers

UNIT 14: **Network Protocols** **308-342**

Basics of configuring NFS, NIS, DNS, FTP, Squid Proxy, DHCP server, ip tables and firewall.

UNIT 15: **Basic Network Security Issues** **343-360**

Introduction to Basic Network Security Issues, Packet Snifflers, DOS Attacks, Linux Kernel Firewalling, Virtual Private Networks

COURSE INTRODUCTION

This is the value added course of PGJMC second semester Programme. This course "Advertising Strategy" will focus on the different aspects of promotions in Marketing. In this course we have briefly discussed all the promotional mixes.

The 2 credits course has 7 units which deals with the introductory concepts of advertising, integrated marketing communications, developing promotional strategies, creating advertising messages and media strategies etc.

The first unit introduces us to Meaning and Concept of Advertising
The second unit gives us a broad idea of Integratd Marketing Communication
Third unit gives us an idea on how to develop promotional strategies.
The fourth and fifth units are about managing advertising.
The sixth unit gives us a broad idea about media strategies.
The seventh and the last unit of introduces us to evaluating advertising effectiveness

Each unit of these blocks includes some along-side boxes to help you know some of the difficult, unseen terms. 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 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. This will help you in making your learning more active and efficient. And, at the end of each section, you will get "CHECK YOUR PROGRESS" questions. These have been designed to self-check your understanding.

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 Common Administrative Tasks

1.6 Identifying, Configuration and Log Files

1.7 Let Us Sum Up

1.8 Further Readings

1.9 Answer to Check Your Progress

1.10 Model Questions

1.1 LEARNING OBJECTIVES

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

- describe the need of the system administrator
- explain the usage of su command
- describe the roles and privileges of a system administrator
- describe the skills required for being a system administrator
- learn some system administrative tasks
- learn about the LINUX configuration and log files

1.2 INTRODUCTION

System administration, required in a multiuser system, is the task of administering and monitoring the system when the system is accessed by multiple users. It is usually handled by a trusted person who is responsible for the administration of the whole system. The main job 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 super user mode which is called root. The super user has the privilege to

issue any command, access any file, and perform all the functions in the system. root is the user-id for the super user to log on to the multiuser system. The use of root command must be limited due to the following reasons:

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

The screen shot of super user login into the system is depicted in figure 1.1.

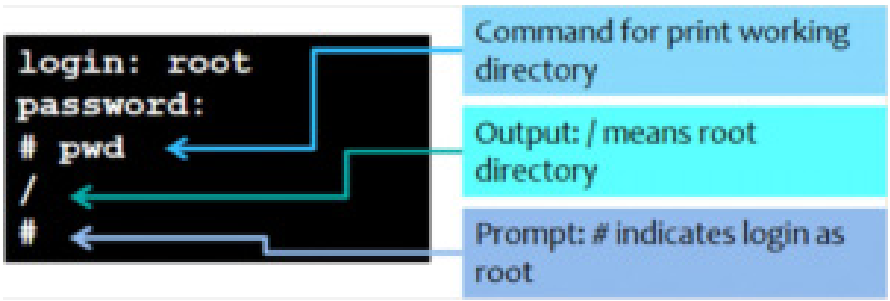


Fig. 1.1 Super user login

The fundamental tasks of the super user includes starting-up & shutting-down the system, creating accounts for new users with necessary environments. Administrative 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. The su command is used for becoming super user. It switches from non-privileged account to the super user account. The who am i command is used to view the login name.

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 user's current environment.

By using the command shutdown, super user can shut-down or restart the system. The option -h with shutdown command is used for a complete shutdown and the -r option is for restarting the system.

1.3 ROLE AND POWER OF THE SYSTEM ADMINISTRATOR

The 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 has to accomplish the task of managing the disk space as well as the backup. Also, the job of system administrator is changing the contents of the file or its attributes like its ownership or permission etc. He/she 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:

- To setup and maintain user account,
- To monitor system performance,
- To verify that peripherals are working properly,
- To quickly arrange to repair for hardware in occasion of hardware failure,
- To create and manage file systems,
- To install software and patching firmware and software,
- To create a backup and recovery policy,
- To monitor network communication,
- To update system as soon as new version of OS and application software,
- To Setup security policies for users,
- To manage network administration and Database administration

Further, Linux system administrator 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 users by answering their technical queries. Many-a-time, they also provide the customer support through e-mail, telephone, knowledge-based programs and other resources.

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.

1.4 QUALITIES OF GOOD SYSTEM ADMINISTRATOR

To become a successful Linux system administrator, one should be a bootstrapper who has high motivation and detail orientation capacity, and also be flexible and ready to follow the instructions given by their superior. The person should be prepared to acquire recent information/ updates and apply the up-to-date technologies that would help the organization to grow. He/she should hold excellent interpersonal skills and should be comfortable working in a team. Further, the quality of identifying the issues related to system and resolving them in an effective way should also be with the individual. He/she must be able to work well under pressure and able to respond and prioritize work according to the stringent deadlines. In concise, a good system administrator should have the following skills:

- Problem-solving skill:
  - Taken up the matter when a computer system malfunctions.
  - Quickly and correctly diagnose the crisis and produce the best solution.
- Analysing skill:
  - Understand the behavior of software.
  - Deployment of software.
  - Troubleshoot the Problems if occur.
- Programming skill:
  - Have the knowledge of different types of programming languages.
  - Scripting or automation of routine tasks.

- Soft-skills:
  - Users to the system should feel comfortable and know that their problems will be fixed in time.
  - Good and healthy interaction among colleagues/team members etc.

1.5 COMMON ADMINISTRATIVE TASKS

User account management: The system administrator can add accounts for new users and removes the accounts of users those are no longer active. The linux commands used for this are:

- useradd
- usermod
- userdel

useradd: Using this command, the administrator can create an account to represent a user who will be working on the system. Each user must authenticate to Linux with an identity that can be used to control their resource access and consumption. User accounts' detail are stored in the **/etc/passwd** file. It is used in the following way:

$ sudo useradd <accountname>

Also, we should not forget to set a password for the created account by using the passwd command.

$ sudo passwd <accountname>

Few common options are used with useradd command:

- --create -home (-m): Adds a home directory.
- --shell (-s): Sets the user's preferred shell if it's different from /bin/bash
- --uid (-u): Specifies a particular user ID (UID)
- --comment (-c): Populates the comment field (usually with the user's full name enclosed in quotes)

usermod:

This command is used to modify the settings of an existing user account. Modifications are necessary when users change their names, request different shells, or need updated password information.

Standard options for usermod includes:

- --comment (-c): Modifies the comment field.
- --home (-d): Modifies home directory information.
- --expiredate (-d): Changes account-expiration settings.
- --login (-l): Modifies the username.
- --lock (-L): Locks a user account.
- --unlock (-U): Unlocks a user account.

Syntax to use this command is:

$ sudo usermod -c <newName> <oldName>

userdel:

We can remove/delete an account representing a user whose role has changed or is no longer with the organization.

To delete a particular account, type:

$ sudo userdel <accountname>

However, before deleting an account, we also have to consider the resources such as the home directory or system mail related to that account.

Few common options associated with userdel that address these resources are:

- --force (-f): Deletes the account (including mail and home directory), even if the user is still logged in
- --remove (-r): Deletes the account (including mail and home directory), but the user must be logged out

Performing backups: The term backup refers to making copies of data that may be used to restore the original if data loss happens. In Linux, tar command is used to backup/restore files. It can also be used to archive a group of files into another file. The following options are used with tar command.

-c to create an archive

-t to display files on archive

-x to extract from archive

-v verbose -display filenames

-z to compress files

-f device name or filename to write to

There are some examples of using tar command as given below:

tar -cvf /dev/st0 backup the entire system to tape

tar -tvf /dev/st0 view the contents of the tape

tar -xvf /dev/st0 restore the entire contents of the file

1.6 IDENTIFYING, CONFIGURATION AND LOG FILES

A Configuration File, also known as a config file, is a unique type of plain text file that is used to control the operations of a program, utility and process. Linux configuration files contain the settings and instructions for different systems, utilities, applications and processes. As the name infers, configuration files contain information about the structure or arrangement of specific parts of the system. They are kept under the /etc directory.

These files are not essentially of any direct use to users. However, they do affect the way a system functions. For example, the fstab file is the filesystem configuration file that includes all the disk partitions and the purpose of each of the partitions. For regular partitions, the conventional mount point may also be listed. The /etc/fstab file is used by mount and other such commands. It is what allows file systems to be mounted automatically at boot time rather than having to mount them all by hand, one at a time.

As a part of system administration tasks, administrator has to frequently modify these files. For instance, every time a new user is added, the /etc/passwd and /etc/group configuration files must be modified.

/etc/passwd is a password file in which each line holds all the information of a single user. A line consists of seven fields, separated by a colon.

- Login name
- Encrypted password
- UID
- Default GID
- GCOS field including full name, office location, extension number, and home phone.

- Home directory
- Login shell

User-id's (UID's) are mapped to the names in this file. Every time a new user is added their information must be added to this file.

/etc/group contains the names of valid groups and the usernames of their members. This file is owned by root and only root may modify it. When a new user is added information on what groups they are a member of must be added here. Group-id's (GID's) are mapped to the group names kept in this file.

Other configuration files include:

/etc/inetd.config, /etc/hosts, /etc/inittab, /etc/hosts.equiv, /etc/export, and /etc/mtab.

/etc/inetd.config is the Internet daemon file. It holds the instructions used by /etc/inetd to manage the network service daemons.

/etc/hosts is the TCP/IP network configuration file. It is accessed by all commands that use the network. The file contains a list of IP addresses, hostnames and hostname aliases.

/etc/inittab is a system configuration file that helps regulate the operation of serial interfaces.

/etc/hosts.equiv contains a list of TCP/IP "trusted" hosts. If machine A is a trusted host of machine B, then all users on machine A may rlogin to machine B without supplying a password.

/etc/export lists all the file systems that are available to other machines via NFS.

/etc/mtab holds information on all mounted file systems.

Linux Log Files

Log files are the set of records that Linux system keeps so that the system administrator keep a track of important events. They contain messages about the server, including the kernel, services and applications running on it. These files can be helpful for troubleshooting system issues.

Most Linux log files are stored in a plain ASCII text file and are kept in the /var/log directory and subdirectory. Logs are generated by the Linux system daemon log, syslogd or rsyslogd.

Viewing Logs:

- First, open the Linux terminal as a root user. This will enable root privileges.
- Use the following command to see the log files:

  $ cd /var/log
- To view the logs, type the following command:

  $ ls

This command displays all Linux log files, such as kern.log and boot.log. These files contain the necessary information for the proper function of the operating system.

Log files are accessed using root privileges. By definition, root is the default account that can access to all Linux files.

Use the following command to access the respective file:

$ sudo less [logname].log

This command displays a timeline of all information related to that operation.

It is be noted that log files are stored in plain text and hence, they can be viewed by using the following standard commands:

zcat - Displays all the contents of logfile.gz

zmore - lists the file in pages, without decompressing the files

zgrep - Searches inside a compressed file

grep - Finds all occurrences of a search term in a file or filter a log file

tail - produces the last few lines of files

Log files can provide a lot of information about the operations of a system. Most directories can be grouped into one of four categories:

- System Logs
- Event Logs
- Application Logs
- Service Logs

Many of these logs can be located in the var/log subdirectory.

- System Logs

Systems log files contain the most significant amount of information

about system functionality. The most common log files include:

- /var/log/boot.log: System Boot log (the boot log stores all information related to booting operations)
- /var/log/auth.log: Auth logs (the authentication log stores all authentication logs, including successful and failed attempts)
- /var/log/httpd/: Apache access and error logs
- /var/log/mysqld.log: MySQL database server log file
- /var/log/debug: Debug logs (the debug log stores detailed messages related to debugging and is useful for troubleshooting specific system operations)
- /var/log/daemon.log: Daemon logs (the daemon log contains information about events related to running the Linux operation)
- /var/log/maillog: Mail server logs (the mail log stores information related to mail servers and archiving emails)
- /var/log/kern.log: Kernel logs (the kernel log stores information from the Ubuntu Linux kernel)
- /var/log/yum.log: Yum command logs
- Event logs

An event log refers to a "log book" that is analyzed and monitored for higher level network intelligence. It can capture many different types of information. For example, it can capture all logon sessions to a network, along with account lockouts, failed password attempts, etc.

- Application Logs

Application logs store information relevant to any application that is executed. These includes error messages, signs of system compromise, and browser identification string. Log files that fall into this category include CUPS Print System logs, Rootkit Hunter log, Apache HTTP server logs, Samba SMB server logs, and X11 server log.

- Service logs

Service logs provides diagnostic information about the resources in your occupancy. When you enable logging on resources, you receive information about the resource in a log file. This information allows you to analyze, optimize, and troubleshoot your resources.



CHECK YOUR PROGRESS

Q 1: Choose the correct options from the followings:

i) The process of starting up a computer is known as

a) Boot Loading b) Boot Record c) Boot Strapping

d) Booting

ii) The shell used for Single user mode shell is:

a) bash b) csh c) ksh d) sh

iii) Which is the only partition mounted in Single user mode

a) boot b) usr c) root d) tmp

iv) Which command is used to restart the Linux system?

a) shutdown -h b) shutdown -r c) restart

d) none of these

v) Which Linux command is used for becoming a super user?

a) su b) man c) who d) None of these



1.7 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.
- Administrator can create a backup and recovery policy.
- The system administrator also monitors 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 and soft skills.
- A configuration file, also known as a config file, is a unique type of plain text file that is used to control the operations of a program, utility or process.
- Log files are a set of records that Linux keeps for the system administrators to keep track of important events.



1.8 FURTHER READINGS

1) S. Das (2012). Your UNIX/Linux: The Ultimate Guide. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
2) Christopher Negus (2012). Linux Bible. (8th Edition). Wiley.



1.9 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: i) c ii) d iii) c iv) b v) a



1.10 MODEL QUESTIONS

Q 1: What is called System administration and why is it required for multiuser system?

Q 2: Explain the responsibility of a system administrator in detail.

Q 3: Mention the skills required for a good administrator.

Q 4: Write the application of sudo and who am I commands.

Q 5: What is the purpose of configuration file and log file?

Q 6: What are the Linux commands used for user management?

*** ***** ***

UNIT 2:INTRODUCTION TO LINUX OPERATING SYSTEM

UNIT STRUCTURE

2.1 Learning Objectives

2.2 Introduction

2.3 Basic idea on Proprietary, Open Source, Free Software

2.4 Linux Operating System

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

2.5 Basic Features of Linux Operating System

2.6 Popular Linux Distributions

- 2.6.1 Red Hat Enterprise Linux/CentOS
- 2.6.2 Ubuntu
- 2.6.3 Debian
- 2.6.4 Fedora
- 2.6.5 SUSE

2.7 Let Us Sum Up

2.8 Further Readings

2.9 Answer to Check Your Progress

2.10 Model Questions

2.1 LEARNING OBJECTIVES

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

- learn about proprietary, open source and free software
- define an operating system
- describe the Linux Operating System
- explain the internal architecture of Linux
- list the basic features of the Linux Operating System
- get brief overview of various Linux distributions like Fedora, Debian, SUSE, etc.

2.2 INTRODUCTION

In the first unit, we have learnt about the need of a system administrator in multiuser operating system. The role and power of a system administrator are also discussed in that unit. In this unit, we concentrate on about LINUX operating system. An operating system is a vital part of a computer system. In general, a computer system works in the presence of three general components: the hardware, the operating system and the applications. The hardware refers to various electronic as well as electro-mechanical parts such as a central processing unit (CPU), a keyboard, a hard drive, a mouse, a printer, etc. Applications are those which are used to operate the computer in order to perform the desired task. The operating system is the component that manages and controls the hardware on one side and manages the applications on other side.

2.3 BASIC IDEA ON PROPRIETARY, OPEN SOURCE AND FREE SOFTWARE

Proprietary Software

Proprietary Software, also known as Closed Source Software, is the software product that legally remains the property of an organisation, a group, or an individual who developed it. The organisation that owns the rights to the product usually does not release the source code, and may insist that only those who have purchased a special licence key can use it. This software is under restrictive copyright licensing and the source code is usually hidden from the users. The products developed and marketed by Microsoft such as Windows 2010, Oracle DBMS, etc. are some of the examples of proprietary software.

Free Software

Free Software, also called Freeware, is licensed at no cost or for an optional fee. It is usually closed source. According to Richard Stallman's definition and adopted by the Free Software Foundation (FSF), free software can be defined as a matter of liberty not price, and it upholds the Four

Essential Freedoms which include the following:

- The freedom to run the program as you wish, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish.
- The freedom to redistribute copies so you can help others.
- The freedom to distribute copies of your modified versions to others.

Open-Source Software

Open-source software is free and openly available to everyone. People who create open-source products, publish the code and allow others to use and modify it. Communities of programmers often work together to develop the software and to support users. Open-source products are usually tested in public by online contributors.

Large companies such as Twitter, Facebook and the BBC make use of open-source technology. For example, the BBC makes use of MySQL and it creates open-source software, such as the program to improve the compatibility of iPlayer on smart TVs.

2.4 LINUX OPERATING SYSTEM

Linux Operating System is one of the UNIX-like operating system, which is designed to offer a free or low-cost operating system for personal computer users. It has gained the status of a fast performing and very efficient operating system. This is a remarkably considered a complete operating system, which includes a GUI (Graphical User Interface), TCP/IP, the Emacs editor, X Window System, etc. Linux is based on the Linux kernel, an operating system kernel first released on September 17, 1991 developed by Linus Torvalds, a then-student at the University of Helsinki. Linux is typically packaged in a Linux distribution which includes 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. Due to its open-

source licensing, Linux is freely available to anyone. However, the trademark on the name "Linux" rests with its creator, Linus Torvalds. The source code for Linux is under copyright by its many individual authors, and licensed under the GPLv2 license.

Popular Linux distributions include Debian, Fedora, CentOS 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 include a solution stack such as LAMP. Linux was originally developed for personal computers based on the Intel x 86 architecture, but later it has been ported to more platforms. 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.

Further, 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 opensource software collaboration which offers the source code 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.

2.4.1 Components of Linux System

Linux Operating System is comprised of the following three components as shown in figure 2.1:

- **Kernel:** Kernel is the central part of Linux OS. All major activities of Linux OS are executed under the control of Kernel. It consists of various modules and by using these modules kernel interacts

directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

- **System Library:** System libraries are special functions or programs by using which application programs or system utilities can access features of the Kernel. These libraries implement most of the functionalities of the operating system.
- **System Utility:** System Utility refers to specialized programs which are responsible for doing specialized individual level tasks.

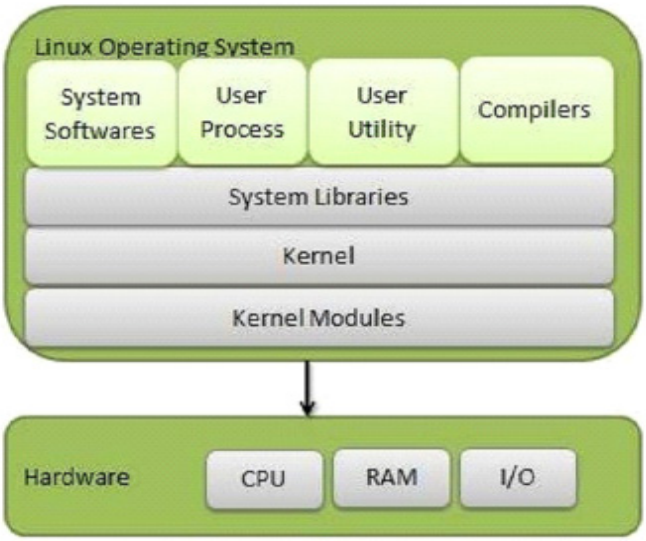


Fig. 2.1 Components of Linux OS

Kernel Mode vs User Mode

Kernel 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 does not require any context switch. Kernel runs each process and provides system services to processes. It also provides protected access to hardware for processes.

in User Mode which does not have 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 Linux operating System

The architecture of the Linux operating system, depicted in figure 2.2, consists of four layers.

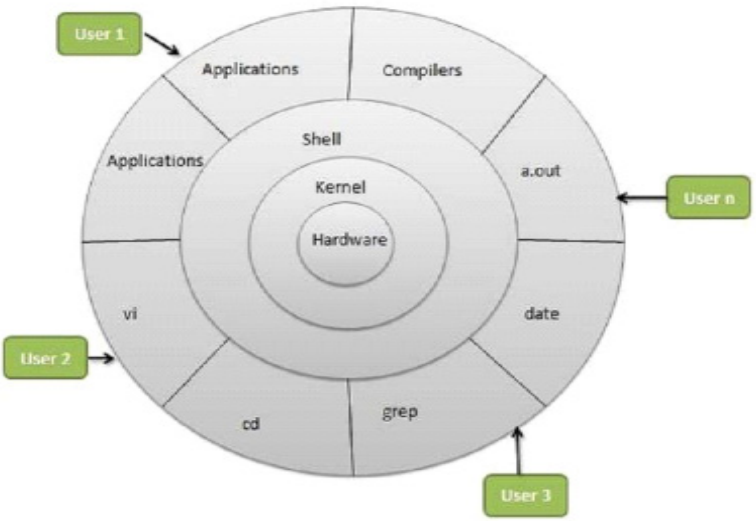


Fig. 2.2 Architecture of Linux Operating system

- Hardware layer - It consists of all peripheral devices such as RAM, HDD, CPU, etc.
- Kernel - It is the core component of Operating System, interacts directly with hardware, and provides low level services to upper layer components.
- Shell - It acts as an interface to kernel, hiding complexity of kernel's functions from users. The shell accepts commands from the user and sends to the kernel to execute.
- Utilities - Utility programs provide the user most of the functionalities of an operating systems.

2.5 BASIC FEATURES OF THE LINUX OPERATING SYSTEM

Linux Operating System provides the following basic features:

- **Portability:** This feature allows software to work on different types of hardware 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 provides a multiuser environment which means multiple users can access system resources like memory / RAM / application programs at the same time.
- **Multiprogramming:** This feature allows multiple applications to run at the same time.
- **Hierarchical File System:** Linux provides a standard tree form file structure in which system files/ user files are arranged.
- **Security:** It provides security in three ways namely authenticating (by assigning password and login ID), authorization (by assigning permission to read, write and execute) and encryption (converts file into an unreadable format).
- **Live CD/USB:** Almost all Linux distributions have Live CD/USB feature by which user can run the OS even without installing it on the system.
- **Graphical user interface (X Window System):** Though Linux is usually considered as a command-based OS, but Linux has packages which can be installed to make the whole OS graphic based.
- **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.

2.6 POPULAR LINUX DISTRIBUTIONS

Technically, Linux is an Operating System Kernel, the core part of an Operating System. The Linux kernel is commonly bundled with many other components to form a complete Linux Distribution. When most people refer to Linux, they are talking about this complete set of programs needed to run a computer, not just the kernel. Because Linux is an open-source operating system, anyone can extend and distribute the system. Several organizations put together the Linux kernel along with a host of programs

and their own special setup utility and distribute the whole package. These are called Linux distributions. 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 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 one 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, which allow 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 defined as a particular collection of application and utility software (various GNU tools and libraries, for example), packaged together with the Linux kernel in such a way that its functionalities 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.

Some commercially backed Linux distributions include Fedora Linux, Red Hat Enterprise Linux, openSUSE (SUSE) and Ubuntu, etc. and entirely community-driven distributions include Debian, Slackware, Gentoo and Arch Linux, etc.

2.6.1 Red Hat Enterprise Linux/CentOS

Red Hat Enterprise Linux (RHEL) is a commercial Linux distribution intended for servers and workstations. 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 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 prevent their official Red Hat Enterprise Linux software from being redistributed, 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. CentOS is a community project that uses the Red Hat Enterprise Linux code, removes all Red Hat's trademarks, and makes it available for free use and distribution. It's a free version of RHEL, so it's good if you want a stable platform that will be supported for a long time. Nowadays, CentOS is now part of Red Hat itself. A screen shot of Desktop window of RHEL is shown in Figure 2.3.

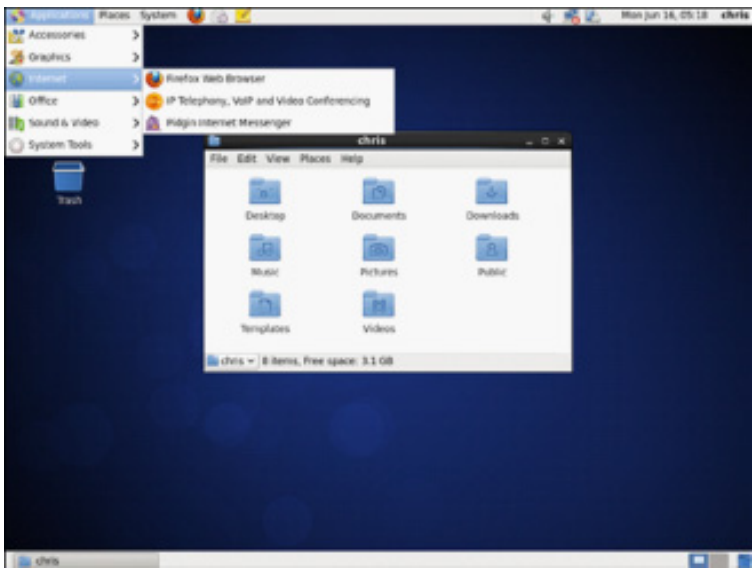


Fig. 2.3 Desktop window of RHEL

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". It came into existence in 2004 by Canonical and quickly became popular for it's a graphical Linux desktop without the use of command line. It is the most well-known Linux distribution. Ubuntu is a next version of Debian and easy to use for beginners. It is comprised of lots of pre-installed apps and easy to use repositories libraries. A screenshot of Desktop window of Ubuntu is depicted in Figure 2.4.

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 default file manager is GNOME Files, formerly called Nautilus.



Fig. 2.4 Desktop window of Ubuntu

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. To increase 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 Debian

Debian is a Unix-like operating system having 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 marketed 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 work of Debian 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. Desktop window of Debian Linux is presented in Figure 2.5.

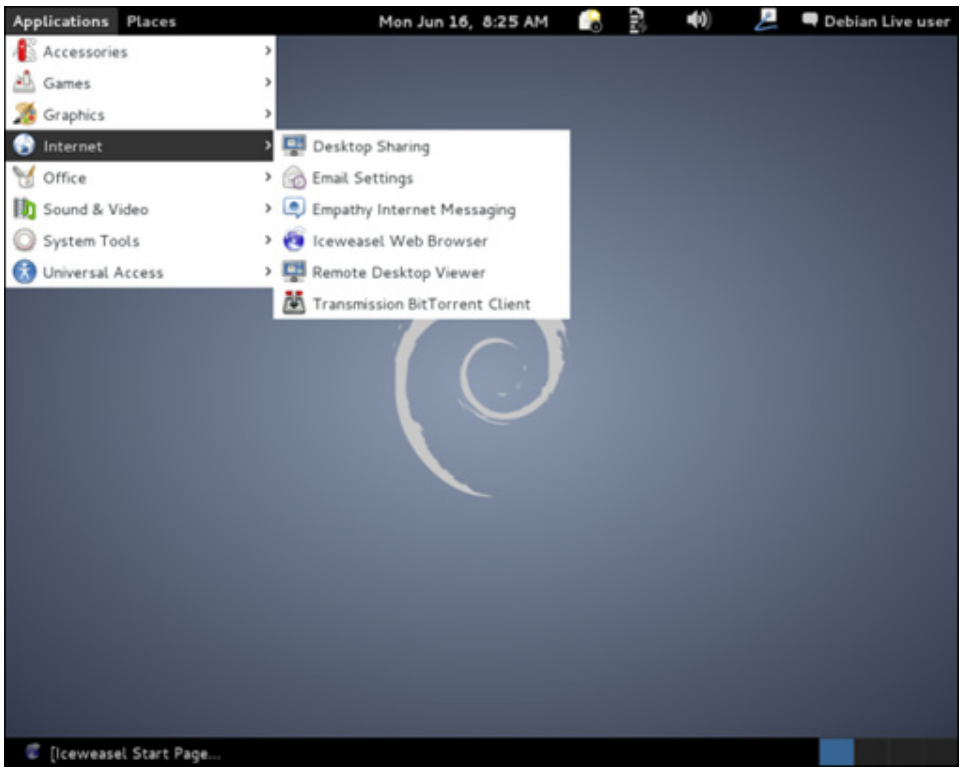


Fig. 2.5 Desktop window of Debian Linux

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 considered the upstream source of the commercial Red Hat Enterprise Linux distribution.

Since the release of Fedora 21, different editions of Fedora 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 upstream Linux 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. The other desktop environments include KDE Plasma, Xfce, LXDE, MATE and Cinnamon.

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 which 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 was released 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 the 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 the commercial version of SUSE Linux, which enables 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 1: Linux is free and open-source operating system. (True/False)

.....

Q 2: Which type of software product has the legal right to use?

.....

Q 3: Shell acts as an interface between the .....and

.....

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

.....

Q 5: How many layers are there in Linux OS architecture and what are these?

.....



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 was first released on September 17, 1991 by Linus Torvalds. Linux is typically packaged

in a Linux distribution.

- 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 Enterprise Linux, Ubuntu, Fedora, SUSE and Debian.



2.8 FURTHER READINGS

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: Proprietary software

Ans to Q No 3: Kernel, user

Ans to Q No 4: Multi user operating system

Ans to Q No 5: Four layers. Kernel, shell, Hardware and Utilities



2.10 MODEL QUESTIONS

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

Q 2: Distinguish between Proprietary, free and open-source software.

Q 3: Discuss about different layers of Linux architecture.

Q 4: What are the basic features of the Linux Operating System?

Q 5: What is a Linux distribution? Name some of the popular Linux distributions.

Q 6: What is Ubuntu? What are the features of Ubuntu?

Q 7: What is Fedora? What are the features of Fedora?

Q 8: Discuss about RHEL system with various features.

*** ***** ***

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 (Red Hat Enterprise Linux 6.0)

3.6 Installing and Configuring Software in Linux

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

3.7 Installing the Ubuntu 20.04

3.8 Let Us Sum Up

3.9 Further Readings

3.10 Answer to Check Your Progress

3.11 Model Questions

3.1 LEARNING OBJECTIVES

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

- learn the hardware and software requirements for installing Linux operating system
- know about hard disk partitions
- describe logical partitioning
- 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 Operating System. In this unit, step-by-step installation of Linux OS will be discussed.

The Linux OS is a freely distributable; cross-platform operating system based on Unix and can be installed on PCs, laptops, mobile and tablet devices, video game consoles, servers, supercomputers and more. Certain hardware requirements are to be met before installing Linux Operating System on a computer. Once this is met, the hard drive requires to be partitioned for installing the operating system. The installation steps for Ubuntu 20.04 LTS will also be discussed.

3.3 INSTALLATION REQUIREMENTS FOR RED HAT ENTERPRISE LINUX

Both of the Red Hat Enterprise Linux versions for 32-bit and 64-bit platforms have the same system requirements as listed below (Table 3.1).

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	1-2 GB minimum
Hard Disk	4 to 5 GB minimum

3.4 PARTITIONING THE HARD DRIVE IN LINUX

3.4.1 Partitions

A hard disk can be divided into several segments called partitions. Each partition operates as if it were a separate hard disk. The idea behind this is that if we want to install two operating systems in one hard disk, we can divide the disk into two partitions. Each operating system uses its own partition area without interfering 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 called the master boot record (MBR) of the disk. BIOS reads in MBR and

starts when the machine is first booted. The MBR contains a small program that reads the partition table, and checks which partition is active or bootable, and then reads the first boot sector of that partition (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, and then it starts it.

There are three types of partitions available namely primary partition, extended partition and logical partition.

Primary partition is a partition which is used to save and boot an operating system. Usually, the operating system is installed in a primary partition which can be set "active" so that the computer can locate the operating system that needs starting. If there are more than one primary partition on the hard disk, only one of them can be visible and active at a time. This active partition is the partition from which the operating system is booted when computer starts. There are at most four primary partitions or three primary partitions and an extended partition on a hard disk.

The partitioning scheme is not built into the hardware, or even into the BIOS. It is only a convention that many operating systems use. 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 like Linux exists with other operating systems and does not require any special measures. 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

Extended partition: Extended partition is a kind of special primary partition which can be subdivided into logical partitions. It can be seen as a container for logical partitions. A hard disk can contain only one extended partition and it is not formatted or assigned a

drive letter, for it is used to create logical partitions.

Logical partition: Logical partitions are partitions created in extended partition. Logical partitions are similar to primary partitions. However, the number of logical partitions that can exist on a hard disk is unlimited.

An extended partition does not directly hold data and logical partitions are mainly used for storing data. Once created, logical partitions must be formatted, and each may be assigned a drive letter. 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.

A sample partitioned 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.

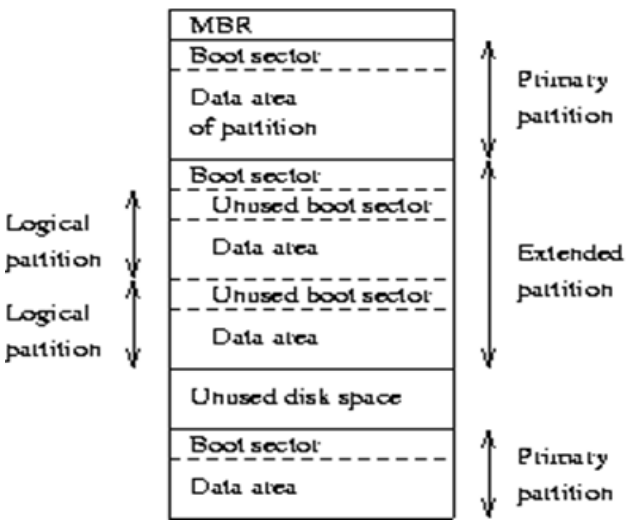


Fig. 3.1: A sample hard disk partitioning.

3.4.3 Partitioning the Hard Drive in Linux

Partitioning disks (Hard Drive) is generally the first step before installing Linux operating system. Before we can create any files, a filesystem must exist. Similarly, we can't make a file system unless there is a partition on the disk.

Partitions are used to divide raw storage space into chunks (slices). This provides the means for organization and hosting of filesystems. Besides that, partitions also help to isolate the filesystem faults.

For a proper Linux system 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 space on the hard drive 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 the 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 we install with Linux. We must create a partition for the root directory. The size of the root partition will vary depending on what to install or plan to install. It is to check 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 we plan to download and try out lots of software, more space is required. If hard drive is in small size, we can trim back on our installed packages to save space. In general, root partition is kept in between 2GB and 8GB.

/home: The third and final partition is created to hold /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 we do not, it will reside on the root partition like everything else.



CHECK YOUR PROGRESS

Q 1: What is the minimum RAM and Hard disk space required for Red Hat Enterprise Linux installation?

.....

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

.....

Q 3: Mention the type of 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 (Red Hat Enterprise Linux 6.0)

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

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

Then Select Install or upgrade an existing system option on Grub Menu (Fig. 3.2)



Fig. 3.2

2. Select a language from the language option.



Fig. 3.3

Generally, English option is used in India (Fig. 3.3).

3. Choose a keyboard type (Fig. 3.4).



Fig. 3.4

Note: To change the keyboard layout type after the completion the installation, use the Keyboard Configuration Tool. Type the system-config-keyboard command in a shell prompt to launch the Keyboard Configuration Tool. If the user is not admin, it prompts him for the root password to continue.

4. Choose an installation media from the screen (Fig. 3.5).

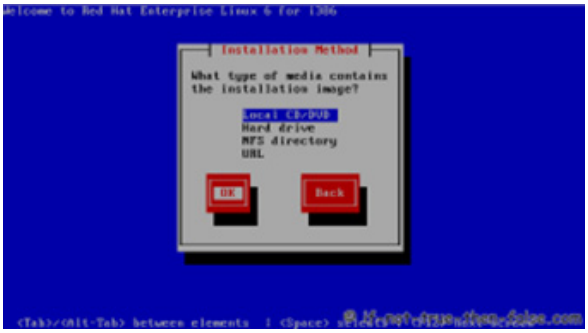


Fig. 3.5

5. Skip DVD media test (or select media test, if you want to test installation media before installation) (Fig. 3.6).



Fig. 3.6

6. Red Hat 6 graphical installer starts, select next (Fig. 3.7).



Fig. 3.7

7. Accept Pre-Release Installation (Fig. 3.8).

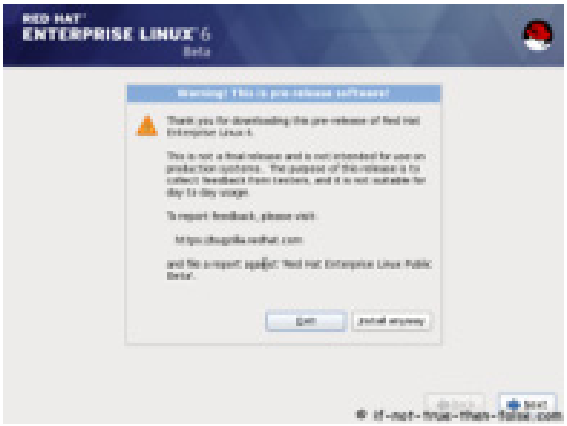


Fig. 3.8

8. Select storage devices.

We 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 wherein Specialized Storage Devices include Storage area networks (SANs), Direct access storage devices (DASDs), Firmware RAID devices and Multipath devices (Fig. 3.9)



Fig. 3.9

9. Insert computer name (Fig. 3.10).



Fig. 3.10

10. Select time zone (Fig. 3.11).



Fig. 3.11

11. Enter a password for root user (Fig. 3.12).



Fig. 3.12

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

12. Select type of installation (Fig. 3.13).

Read every options info carefully. And select encrypting if needed and option to review and modify partition layout.



Fig. 3.13

Partitioning allows us to divide the hard drive into isolated sections, where each section behaves as if it is a separate hard drive. Partitioning is particularly useful if multiple operating systems are to be installed. In the above screen we can choose to create the default partition layout using one of four different ways or choose to partition storage devices manually to create a custom layout. The first four options allow us to perform an automated installation without having to partition our storage devices. If we do not feel comfortable with partitioning our system, we can choose one of these options and let the installation program partition the storage devices for us. Depending on the option we select, we can still control what data (if any) is removed from the system.

Type of installation can be chosen from:

Use All Space: Select this option to remove all partitions on our 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 we may have on our hard drives (such

as VFAT or FAT32 partitions).

Shrink Current System: Select this option to resize our 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 the 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 we select this option.

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

13. Review partition layout (Fig. 3.14).

Modify if needed. Default setup with ext4 and LVM looks good for desktop machine.



Fig. 3.14

14. Accept write changes to disc (Fig. 3.15).



Fig. 3.15

15. Writing changes (creating partitions) to disc (Fig. 3.16).



Fig. 3.16

16. Configure boot loader options (Fig. 3.17).

Select device to install bootloader and check/create boot loader operating system list.



Fig. 3.17

17. Select software to install and enable repositories.

In this case, we have to select Software Development Workstation and enable Red Hat Enterprise Linux 6.0 Beta Repository and select Customize now (Fig. 3.18).



Fig. 3.18

18. Customize package 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.

Select PHP and Web Server to installation (Fig. 3.19).

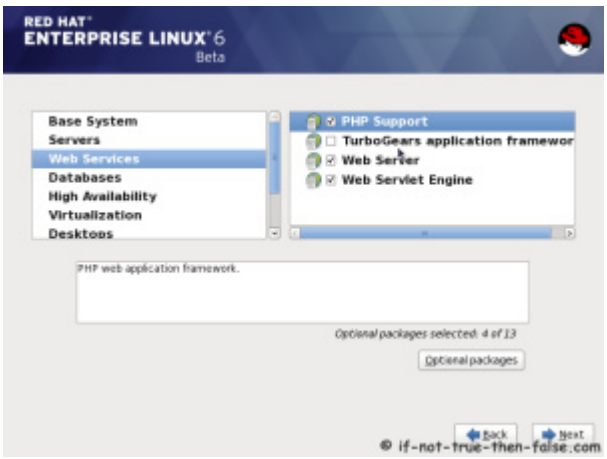


Fig. 3.19

Select MySQL and PostgreSQL Databases (Fig. 3.20).

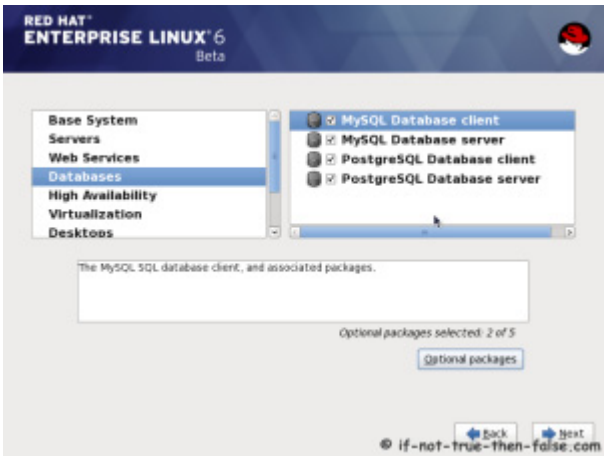


Fig. 3.20

Select set of Development tools like Eclipse IDE (Fig. 3.21).

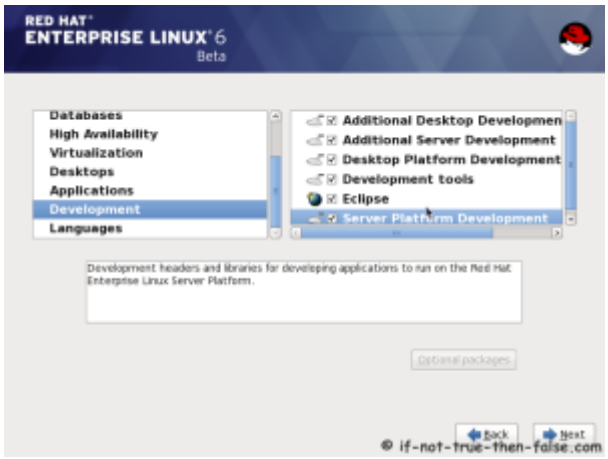


Fig. 3.21

19. Checking dependencies for installation (Fig. 3.22).

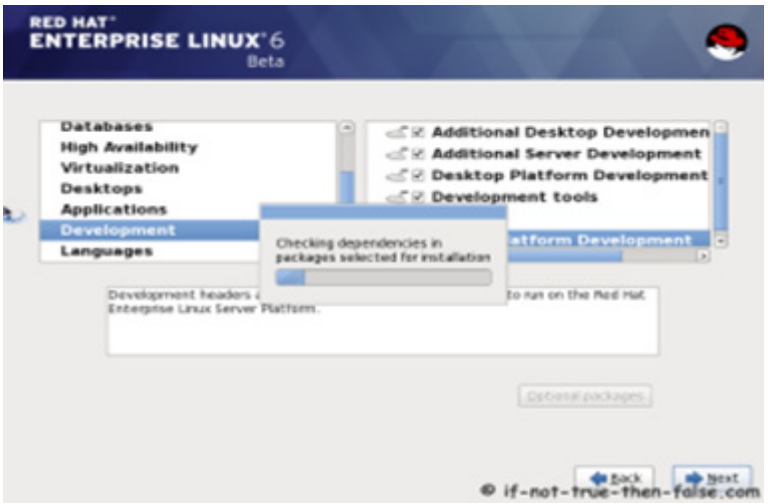


Fig. 3.22

20. Starting installation process (Fig. 3.23).



Fig. 3.23

21. Installing packages (Fig. 3.24 & 3.25).



Fig. 3.24



Fig. 3.25

22. Installation is complete (Fig. 3.26).

Click reboot computer and remove installation media.

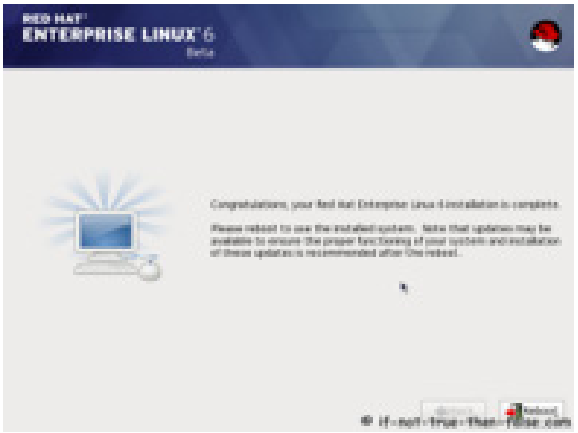


Fig. 3.26

3.6 INSTALLING AND CONFIGURING SOFTWARE IN LINUX

The standard Linux package format is called RPM. RPM stands for Red Hat Packet Manager. It is originally developed by Red Hat and widely used in the Linux community Distributions which 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) have introduced several advanced features such as automatic dependency resolution and signed packages. Debian packages are used by Debian GNU/Linux, and distributions based on it, including Ubuntu, Knoppix, and Mepis. A Debian package file is normally named something like program-version-other.deb. It is to be noted that to install software packages, user needs to become SuperUser.

3.6.1 Debian, Ubuntu: apt

There is an array of tools for working with DEB packages, but the one the commonly used is apt-get, arguably the easiest of Linux package management tools. apt-get is easy to use because it not only keeps track of what packages are installed, but also what other packages are available. We can install software package by the following way.

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

To remove any software package, we can use the following.

```
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 which is done in the following way.

apt-get update

Moreover, upgradation of all the packages that have patches or security updates is also required. The following command will do this all at once.

apt-get update

apt-get upgrade

3.6.2 Fedora, Red Hat: yum

Like what apt-get does for Debian packages, yum does for RPM packages. yum can download and install software packages from a configured repository by the following way.

yum install ${packagename}

Software can be removed by using the following command.

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 the following command:

yum update

We can also explicitly update a single package in the following way:

yum update ${packagename}



CHECK YOUR PROGRESS

Q 5: Linux cannot be installed on Direct access storage devices. (State TRUE or 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 Debian?

.....

3.7 INSTALLING THE UBUNTU 20.04

Ubuntu 20.04 was released on April 23, 2020 and it is known as a Long-Term Support (LTS) release, meaning that it can receive support and updates until April 2025. Ubuntu 20.4 offers some new features which include the following.

- **Linux Kernel 5.4 -** This version of the Linux kernel adds support for a wider range of processors. It also includes improvements for boot speed, power-saving, and USB-C, plus several security features.
- **Gnome 3.36 -** Gnome has been updated and improved. System animations should be smoother and have a lower impact on CPU usage.
- **ZFS 0.8.3 file system -** Improved performance and native encryption.
- **Dark theme**
- **Redesigned login screen**
- **Newer versions of popular programming languages -** Supports the languages like Python 3.8, OpenJDK 11, PHP 7.4, Glibc 2.31, GCC 9.3, Ruby 2.7.0, Perl 5.30, Golang 1.13.

To install Ubuntu 20.04 in a computer system, the following requirements are essential:

- 2 GHz dual-core processor
- 4 GB memory
- 25 GB available disk space for storage
- DVD drive or USB port
- At least a 4GB USB drive

Steps involved to install Ubuntu 20.04 are as follows.

Step 1: Download the Installation Media

1. In a web browser, visit the Ubuntu download page and pick the Ubuntu version suitable for your machine. The most popular versions

include:

- Ubuntu Desktop
- Ubuntu Server
- Ubuntu Derivatives

2. Once you find the version you need, click the green Download button (Fig. 3.27). It will take back to a thank-you page, and download will start. The downloaded file will be an ".iso" file. You can use it to create a bootable USB drive.

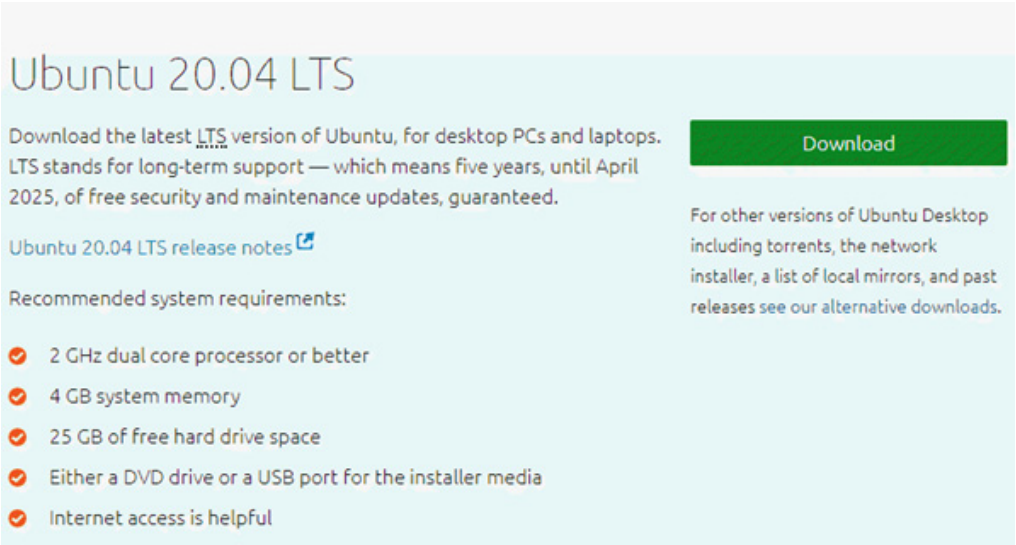


Fig. 3.27

3. Save the file to a location of your choice.

Step 2: Create Bootable USB

You will need a USB drive with 4GB or more. This process will delete all data on the USB drive. Make sure to backup any existing data on the USB drive.

Option 1: Create a Bootable USB Drive on Ubuntu

Use the Create Start-up disk tool:

- Open a search dialog, and type create startup.
- If it's not installed, the Software Center will offer the option to install it - choose the option for USB drive, then open the utility.
- In the top pane, click Other, then browse and select the Ubuntu 20.04 .iso file you downloaded.
- In the bottom pane, select your USB drive.

- Click Make startup disk.

Option 2: Create Bootable USB Drive on Windows

You'll need to install a third-party utility called Rufus to create a USB bootable drive.

1. Download the Rufus utility. Scroll down to the download section and click the link to download the latest version of Rufus (Fig. 3.28).

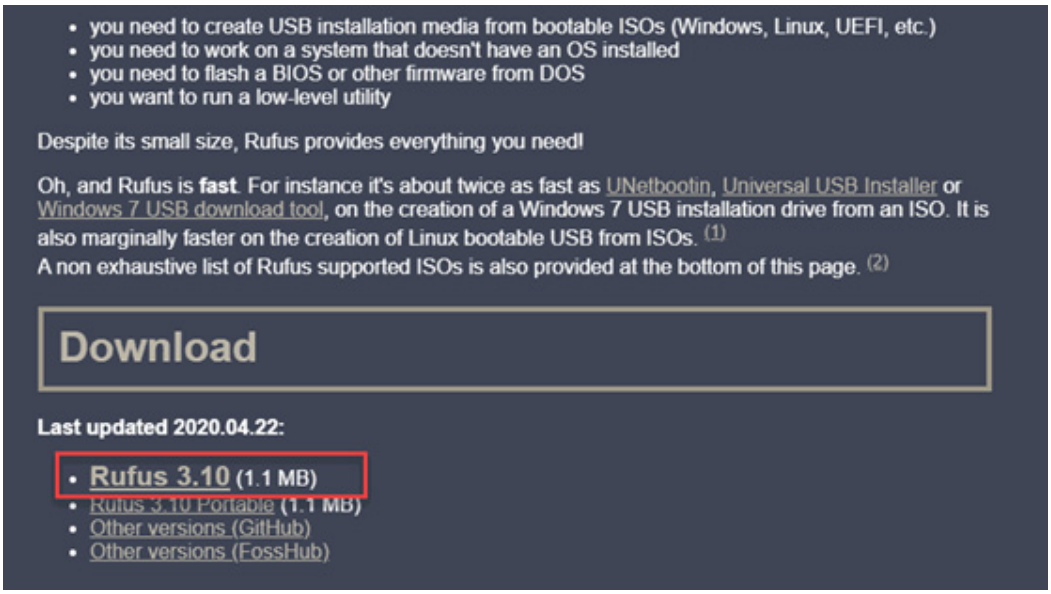


Fig. 3.28

2. Run the file once downloaded.

3. A pop-up dialog opens. You will be prompted whether you want to check for online updates. Select No. See fig. 3.29.

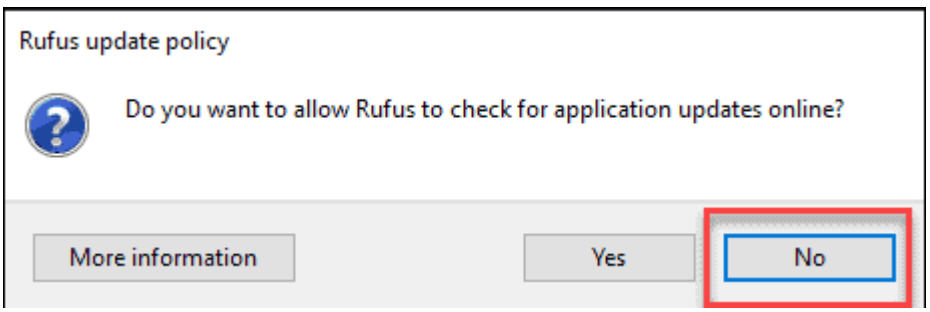


Fig. 3.29

4. The Rufus utility launches. Plug in the USB drive - you should see the drive pop up in the device field (Fig. 3.30).

- Set the USB as the device you wish to write to.
- In the Boot Selection drop-down, click Disk or ISO Image.

- Click the Select button to the right.
- Browse and select the .iso Ubuntu file you downloaded earlier.

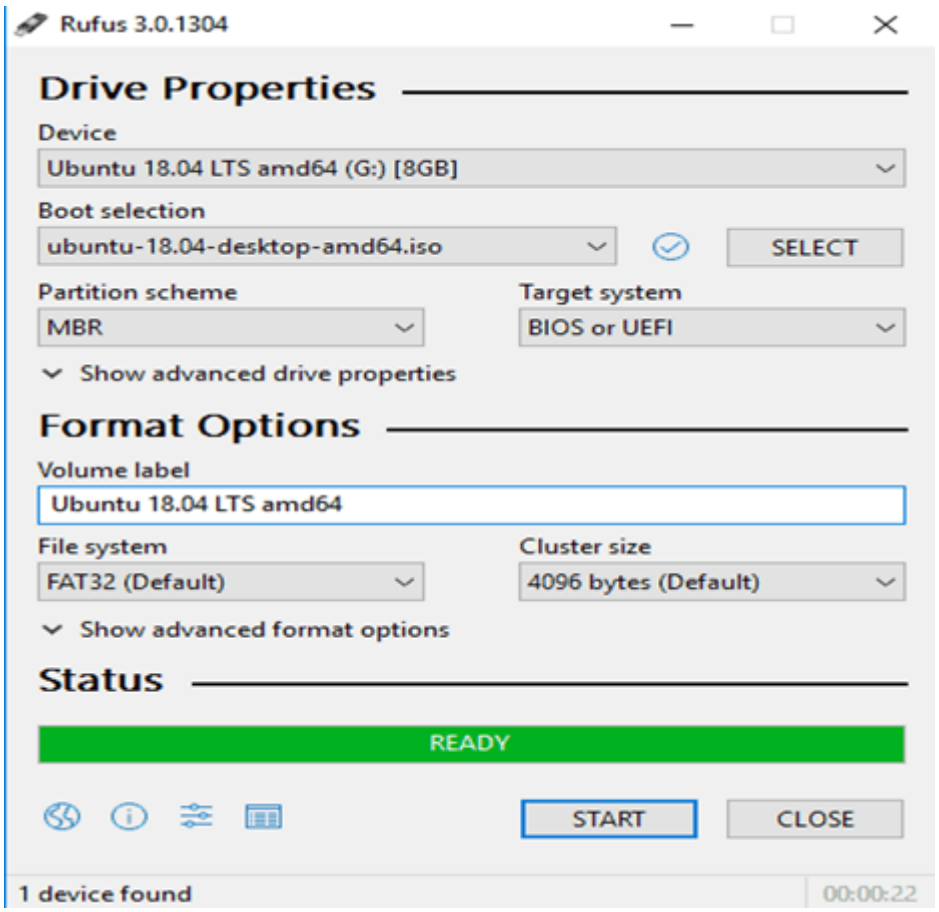


Fig. 3.30

5. Click Start.

Step 3: Boot up Ubuntu from USB

1. Turn off your system. Make sure you remove all other USB devices, such as printers, memory cards, etc.

2. Insert the Ubuntu USB drive into the system and turn on your machine.

There are two possible scenarios:

- The computer boots the USB drive automatically.
- You need to manually configure USB booting in the Boot Menu or BIOS/UEFI.

3. To manually configure the boot order, tap the boot menu key about once or twice per second as soon as the computer powers on.

The boot menu key may be different depending on your computer manufacturer. Below is a list of common boot keys associated to a brand (Table 3.2):

Asus	F8 or Esc
Acer	F12, F9 or Esc
Compaq	F9 or Esc
Dell	F12
Fujitsu	F12
HP	F9 or Esc
Lenovo	F8, F10 or F12
Samsung	F2, F12 or Esc
Toshiba	F12

4. Once you see your boot menu, use the arrows to pick the Ubuntu media to boot from. For a DVD, the entry will usually have DVD or Optical in the name. USB is usually labelled USB.

Your system should start loading the Ubuntu live disc menu.

Step 4: Run Ubuntu

You can test Ubuntu 20.04 before you commit to installing it. The .iso includes a live mode that only runs in memory.

Launch this mode by clicking Try Ubuntu. See fig. 3.31.

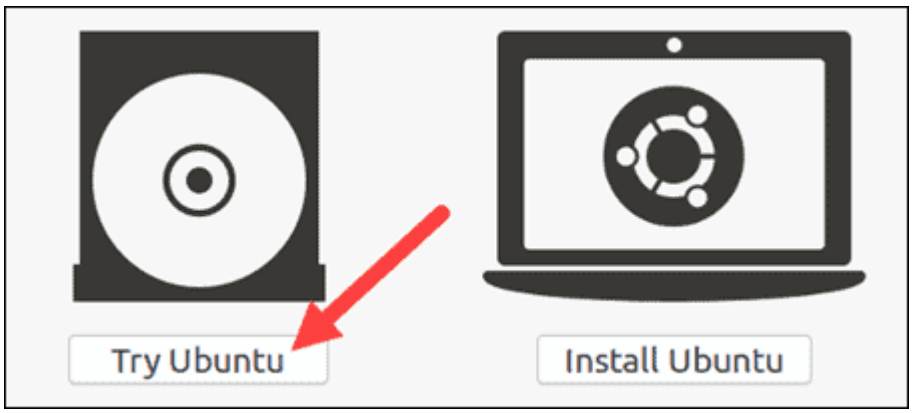


Fig. 3.31

Step 5: Install Ubuntu 20.04 LTS Desktop

To begin the installation, click Install Ubuntu (Fig. 3.32).

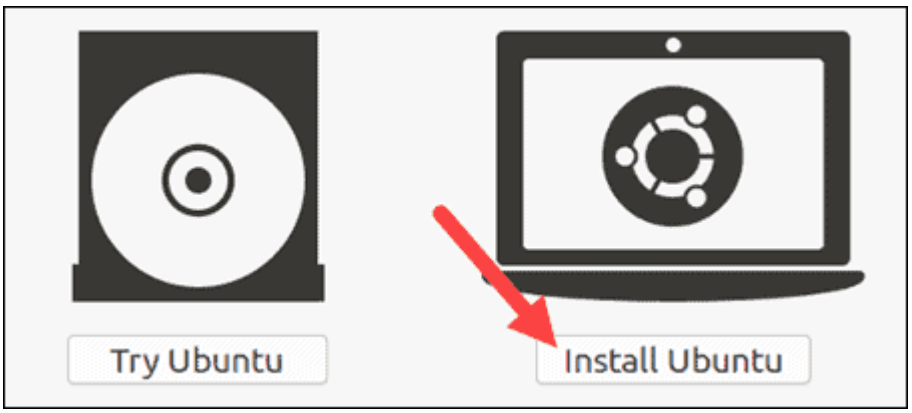


Fig. 3.32

Choose Keyboard Layout

By default, the system will select English (US).

If you have a non-standard keyboard, you can select it in the list. Alternately, click Detect Keyboard Layout and the system will automatically choose your keyboard. If you need to test your keyboard, use the labelled field.

When you're ready, click Continue. See fig. 3.33.

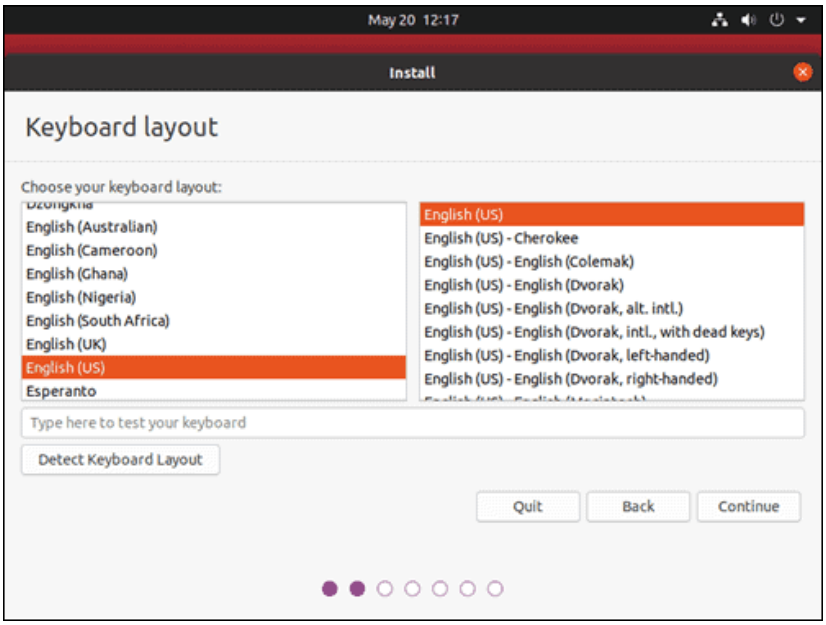


Fig. 3.33

Choose Starting Applications

- **Normal Installation -** This is the full Ubuntu Desktop experience, with office software, games, and media players.

- **Minimal Installation -** Choose this to save disk space, especially if you won't be using media players or productivity software.

  You will also be asked to confirm other options:
- **Download updates while installing Ubuntu -** This does the work of downloading large package files during the installation. Once the installation finishes, the packages will be ready to apply as updates.
- **Install third-party software for graphics and Wi-Fi hardware and additional media formats -** Some hardware, like graphics cards and wi-fi cards, do not have open-source driver support. Also, some media formats, such as ".wmv", do not fall under the GPL license. If you need support for these, you'll need to agree to additional terms of use.

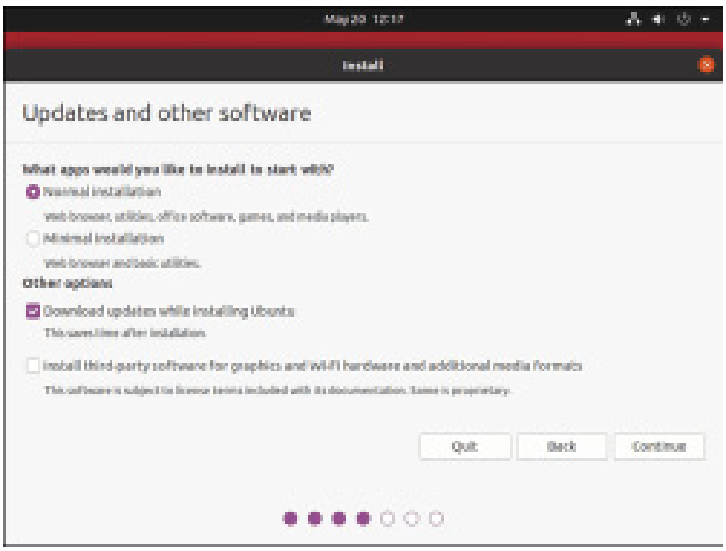


Fig. 3.34

Disk Partitioning

Next, an Installation Type dialog will appear. You can wipe the hard drive clean prior to installing Ubuntu by clicking Erase disk and install Ubuntu. If you go way, skip ahead to the next step.

Advanced users may want to edit Advanced Features. Use this to specify your own disk partitions or set other advanced options:

- **Use LVM with the new Ubuntu installation:** LVM stands for Logical Volume Management. This is a tool for dynamically managing different virtual drives on your system. It's much like an enhanced version of the gparted tool.

- **Encrypt the new Ubuntu installation for security:** This will encrypt the drive's contents. You'll choose a security key, which will be required to decrypt and use the drive.
- **Experimental: Erase disk and use ZFS:** ZFS refers to Zettabyte File System, but it has grown into a hybrid file system and volume manager. Since it's still being tested, avoid this setting on mission-critical production systems.

If you'd rather create your own hard drive partitions, click Something Else.

The next screen will allow you to create your own partition table and logical drives. This lets you divide a physical hard drive into different partitions. The operating system sees partitions as individual drives.

Note: Some users create their /home directory on a separate partition. If the operating system needs to be reinstalled, the partition with the /home directory is unaffected.

Click Continue to apply your changes to the drive partitions.

You'll be asked to Write changes to disks? None of the options you've selected are permanent until you click Continue on this screen. Click Continue to proceed.

Select Time Zone

Once the system formats the disk partitions, the installer will ask Where are you?

Type the nearest large city into the box, and the system will set your local time zone.

Click Continue.

Create User Account

Next, you'll need to configure a user account. Fill in the following fields:

- Name: Your actual name.
- Computer name: This is the hostname or network name.
- Username: The user account name you want to use.
- Password: Enter and confirm a strong password - the installer will automatically evaluate your password strength.

- **Log in automatically:** This is not recommended for publicly accessible servers.
- **Require my password to log in:** This is recommended for publicly accessible servers.

Click Continue to install Ubuntu.

Once the installer finishes, remove the Ubuntu installation media.

You'll be prompted to Restart Now (Fig. 3.35).

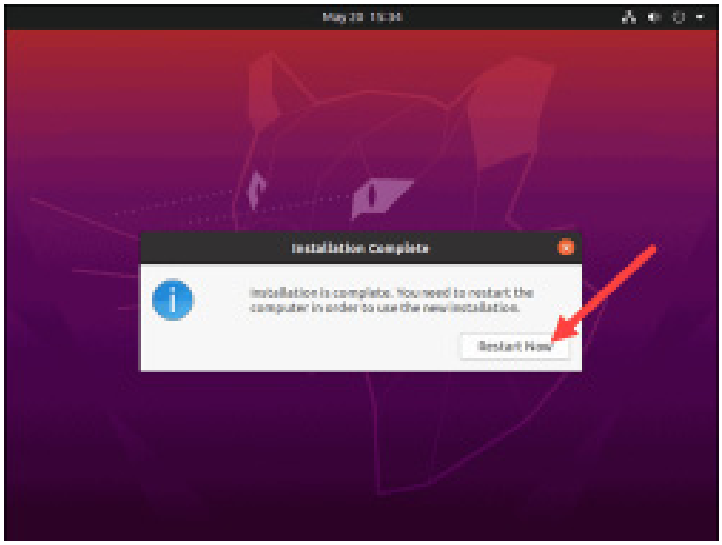


Fig. 3.35

The system should boot into your fresh install of Ubuntu 20.04.



3.8 LET US SUM UP

- Both Red Hat Enterprise Linux versions on both 32-bit and 64-bit platforms need 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 is partitioned is stored in its first sector which is named as 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 recommended: 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 are installed 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.
- We can install Red Hat Enterprise Linux on a large variety of storage devices- either basic or specialized storage devices. Basic Storage Devices include 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 us 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, we 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 Debian and Ubuntu Systems the command apt-get install ${packagename} is used. To remove software, we must use apt-get remove ${packagename}.
- To install packages on Fedora and Red Hat Linux systems, the command yum install ${packagename} is used.

3.9 FURTHER READINGS

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.

3.10 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: minimum 1 GB RAM and minimum 4 GB hard disk.

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}

3.11 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?

Q 7: What are the system requirements for installing Ubuntu 20.04?

Q 8: Mention the steps of installing Ubuntu 20.04?

*** ***** ***

UNIT 4: LINUX KERNEL

UNIT STRUCTURE

4.1 Learning Objectives

4.2 Introduction

4.3 Linux Kernel Concepts

- 4.3.1 Major 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 Readings

4.11 Answer to Check Your Progress

4.12 Model Questions

4.1 LEARNING OBJECTIVES

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

- learn about Linux kernel
- describe the architecture of the Linux kernel
- describe the role of the kernel in the Linux system
- define device driver and its types
- explain how a device is represented in Linux system
- know about redirection and piping

4.2 INTRODUCTION

From the previous units, we have already learned the basic concepts along with its architecture and the basic features of the Linux operating system. We are also accustomed with the basic ideas about the various Linux Distributions. In the last unit, the concept of the linux installation process & requirements for the installation, how to partition the hard drive in linux, how to install the linux etc. were described. Besides these, we have also learned to install and configure different software or application in linux in the previous units.

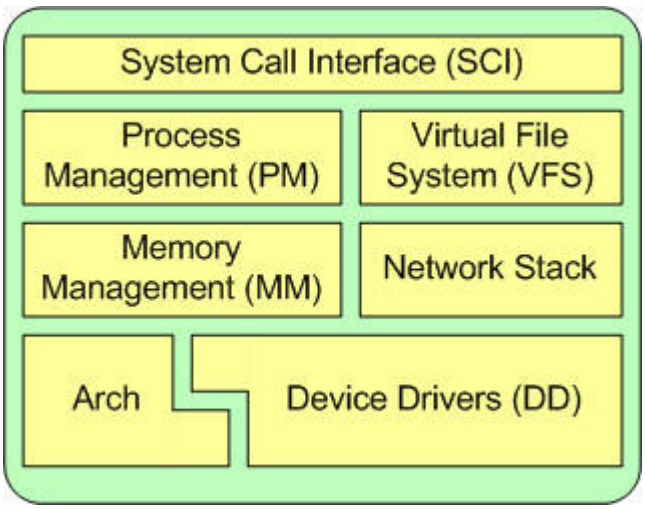
In this unit, we are going to discuss about the Linux Kernel, the role of the kernel in the linux system and the architecture of the Linux kernel. By going through this unit, we will be able to know about the concepts of device drivers in linux system in addition to booting and shutting down process of the system. The important features such as Redirection & Piping and how they work will also be learnt in this unit besides knowing about the standard input, standard output and the standard error in linux 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 accumulates all of the basic services. The linux kernel is layered into a number of distinct subsystems. The kernel is conceptually composed of six major subsystems: System Call Interface, the Process Management, the Memory Management, the Virtual File System, the Network Interface, and the Device Driver.

The major subsystems of the kernel are presented in the figure 4.1.



4.3.1 Major Functions of Linux Kernel

The major functions performed by kernel are described below.

- **Process Management:** The kernel handles the creation and destruction of the processes and their connection to the input and output systems. The inter process communication, i.e., the communication among different processes is also handled by the kernel. Process management can provide the information about what is happening with a process, file allocated to process, state of process (like running, waiting, stop), etc.
- **Memory Management:** The memory management is the most important function of kernel. Kernel assigns the 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).
- **File System Management:** In Linux, almost everything is considered as a file. The Linux file system is a hierarchically data structured tree starting from root followed by other subdirectories which organizes the file system. A directory is a group of files. Two types of directories available in Linux system include: 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.

- **System Call Interface:** The system call interface accomplishes function calls from user space into the kernel. It can be architecture dependent, even within the same processor family.
- **Device controlling:** This subsystem which maps to the 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 into the kernel.
- **Network Management:** This function is responsible for 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 is divided into two levels - User Space and Kernel Space. The architecture is depicted in the figure 4.2.

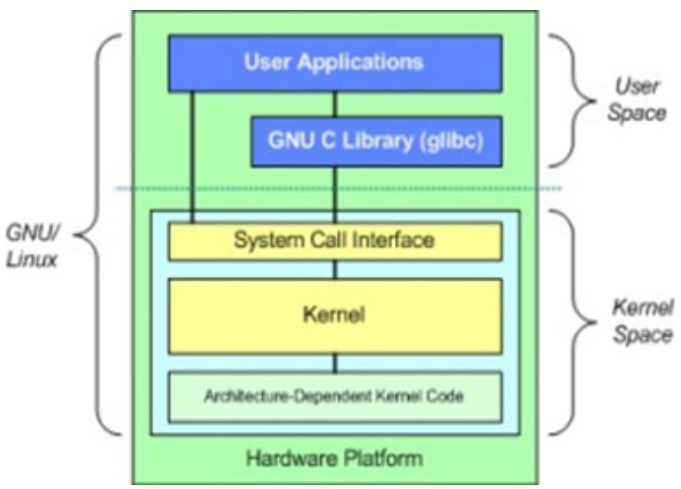


Fig. 4.2: Architecture of Linux Kernel

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

Kernel Space: Kernel Space is for executing kernel programs. It can access full part of memory and directly interact with the hardware. The kernel space is further subdivided into three levels - System Call Interface, Kernel and Architecture-Dependent Code. System Call Interface sits in between the user space and kernel space and it executes 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.

To summarise, the kernel space is the memory area that is reserved to the kernel while user space is the memory area reserved to a particular user process. The kernel space is accessed protected so that user applications cannot access it directly, while user space can be directly accessed from code running in kernel mode.

4.3.3 Features of Linux Kernel

As discussed in previous units, 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 by employing dynamically loadable kernel modules. And they can be inserted at booting time when they are required. Though much of Linux is independent of the architecture on which it runs, but there are some elements which are considered for normal operation and for efficiency reason. Low-level Scheduling of Processes, Inter-

process Communication, Process Synchronization and Context Switching are some other features of Linux kernel.

4.4 DEVICE DRIVERS

The software that handles or manages a hardware controller is known as a device driver. The Linux kernel device drivers are, essentially, a shared library of privileged, memory resident, low-level hardware handling routines. It is Linux's device driver that handles the peculiarities of the devices they are managing. The device drivers have a special role in the kernel of the Linux system. All the devices are represented by files. Nearly 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 referred to as device driver. The kernel calls the appropriate device driver, whenever a particular device is accessed and then the kernel passes some parameter 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 following Linux command lists device names connected to the system.

```
$ 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

The Linux system classifies most of the hardware devices into two forms: block or character devices. These devices are indicated by the first character (either b or c) in the permission field. The character b indicates that it is a block device file and if it is c then it means that it is a character device file. A block device can store or transfer a block of data at a time, wherein 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. 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) which is presented below.

crw-w-- 1 root tty 4, 0 Feb 3 18:11 tty0

crw-w-- 1 root tty 4, 1 Feb 3 18:11 tty1

4.5 SYSTEM START UP

To start-up the system, we have to power on the machine such that the power is supplied to the peripherals those are immediately required. When the kernel is loaded, these peripherals are also to be loaded. The start-up or booting process follows through a number of stages.

At first, the kernel is loaded into the memory. It is represented by the file /vmlinuz. BIOS (Basic Input/Output System) then performs some system integrity checks. It searches, loads, and executes the boot loader program from 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. 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. This file loads and executes kernel and then the kernel starts spawning further processes, among which the most important is init. The kernel executes init process 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 which is called the 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

If we look at the /etc/inittab file to decide the Linux run level, it may observe that 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.

o Run level 0 - /etc/rc.d/rc0.d/

o Run level 1 - /etc/rc.d/rc1.d/

o Run level 2 - /etc/rc.d/rc2.d/

o Run level 3 - /etc/rc.d/rc3.d/

o Run level 4 - /etc/rc.d/rc4.d/

o Run level 5 - /etc/rc.d/rc5.d/

o Run level 6 - /etc/rc.d/rc6.d/

4.6 SYSTEM SHUTDOWN

The shutdown command is used to bring the system down in a secured way. The /sbin/shutdown command controls the shutdown sequence. This shutdown job is executed by signalling the init process, asking it to change the run level. The activities performed by shutdown are-

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

Different options used with shutdown command can be explained with following example.

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 in Linux include >, < and >> where < means the input redirection, > specifies the output redirection operator and >> operator 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. Suppose the thisfile.txt contains the following texts.

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 wc command will count and display the number of lines, words and characters in the file thisfile.txt.

$ 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.

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 another file thatfile.txt.

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

When a command begins running, it usually expects that three files are already open: standard input, standard output, and standard error. A number, called a file descriptor, is associated with each of these files, as follows:

File descriptor 0 means Standard input

File descriptor 1 means Standard output

File descriptor 2 means Standard error (diagnostic) output

A child process normally inherits these files from its parent. All these three files are initially assigned to the peripherals like 0 to the keyboard, 1 and 2 to the display. The shell permits them to be redirected elsewhere before control is passed to a command.

When you enter a command and no file name is given then your keyboard is the standard input, denoted as stdin. When a command finishes, the results are displayed on your screen. This is called the standard output, denoted as stdout. By default, commands take input from the standard input and send the results to standard output. Standard Error, denoted as stderr, is where error messages go. By default, this refers to monitor screen.

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 a character stream without knowing the source of the stream. If we 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, 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 commands 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 monitor screen. Again, the standard output can be redirected.

For example,

$ wc thisfile.txt > thatfile.txt

This command redirects 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 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.

For 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.

$ cat newfile

cat: testfile: No such file or directory

4.8 PIPING

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux. Pipes are used to redirect a standard input or standard output from one command or program to another for further processing. 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 (the vertical

slash symbol on your key board, not to be confused with the small 'L') that acts as the connector of two commands. Here, the output of one command acts as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes. Consider, there are four text files which are listed as:

$ 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.

Pipes are unidirectional i.e., data flows from left to right through the pipeline. The syntax is as follows:

command_1 | command_2 | command_3 | .... | command_N

For example, to list all files and directories and give it as input to more command, the following command with pipe displays the output as given below.

$ ls -l | more

Output is (Fig. 4.3):

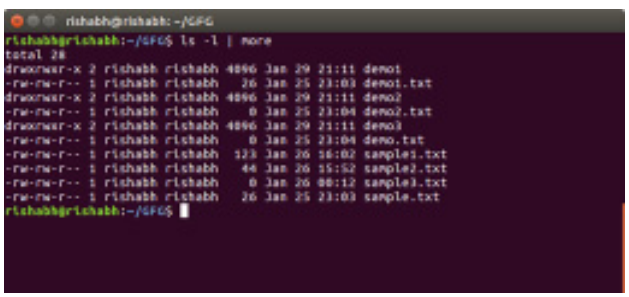


Fig. 4.3: Output of the command "ls -l | more"



CHECK YOUR PROGRESS

Choose the correct option and fill up the blanks:

Q 1: Redirection is a process of switching of the standard stream of data.

a) True b) False

Q 2: How many files are used for representing different standard streams?

a) 1 b) 2 c) 4 d) 3

Q 3: Which stream is connected to the display?

a) standard input b) standard output

c) standard error d) None of these

Q 4: The command wc < sample.txt will count data from the file sample.txt.

a) True b) False

Q 5: Which of the following symbol(s) can be used to redirect the output to a file or another program?

a) | b) > c) >> d) |, > and >>

Q 6: Device Drivers reside in which environment?

a) Kernel space b) Device space

c) Application space d) User space

Q 7: Which command is used to display the operating system name

a) os b) unix c) kernel d) uname

Q 8: A ___ is a special program that can control a specific type of hardware. (Pick the right option)

a) Shell b) Device driver c) Complier d) Database

Q 9: _____ directory has system configuration files, network configuration files, application configuration files and other installed software. (Pick the right option)

a) /var b) /mnt c) /opt d) /etc

Q 10: In Linux, devices are placed in _____ directory. (Fill up the blank)



4.9 LET US SUM UP

- Kernel is the core component of Linux system that interact with the hardware.
- Each device is considered 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. At first, the kernel is loaded into the memory. BIOS (Basic Input/Output System) then performs some system integrity checks. It searches, loads, and executes the boot loader program from floppy, CD-ROM, or hard drive.
- 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 the character streams.
- Piping is the technique of redirecting a standard input or output from one command to another. Piping is done by using a special operator "|" called pipe that acts as the connector of two commands.



4.10 FURTHER READINGS

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



4.11 ANSWERS TO CHECK YOUR PROGRESS

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



4.12 MODEL QUESTIONS

Q 1: Define Kernel? Describe the functions and the architecture of the Linux kernel.

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

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

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

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

Q 6: What do you mean by file descriptor? State the difference 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 Linux File System Internal Structure

5.5 How Linux Access Files

5.6 Storage of Files

5.7 Linux Directory Structure

5.8 LILO, GRUB Boot Loader

5.9 Basic Types of Files

- 5.9.1 Regular Files
- 5.9.2 Derectory Files
- 5.9.3 Special or Device Files

5.10 Other Common Types of Files

5.11 File Attributes

5.12 Let Us Sum Up

5.13 Further Readings

5.14 Answer to Check Your Progress

5.15 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 and boot loader
- learn about boot block, super block, inode and data block
- describe the journaling concept in file system
- know about various file types used in Linux system
- describe different file attributes.

5.2 INTRODUCTION

In the previous units, the features of Linux Operating System and the working of Linux Kernel were described. Also, the redirection and piping concepts in Linux Operating System were discussed. In this unit, we will study the Linux File System and the related properties. Subsequently, the functions of different file systems in Linux will be explained here in this unit. Further, we will study the different types of files, I-node and the file attributes.

5.3 BASICS OF LINUX FILE SYSTEM

In Linux, all data and programs are stored in files. No restriction is imposed on the structure of a file. The size of a linux file is the exact size of the contents that stored inside the file. The files in a file system are connected logically. A file system manages how the files are stored and can be retrieved easily. Without a file system, it is not possible for an operating system to know the positioning of a file i.e., where a file starts and where ends and thus the 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 available in linux operating system. They are different from each other based on the structure, logic and other important properties like flexibility, speed, size etc. Some of the file systems that are being commonly used for better understanding of the functionalities of a file system are described as follows.

Any Linux distribution has different kinds of file system to partition a disk. Some of the commonly used file systems include the following:

- ext

- ext2
- ext3
- ext4
- xfs
- JFS
- ReiserFS
- ISO9660 etc.

5.3.1 ext

ext stands for extended file system. It is considered the first file system used in Linux. It was developed by Remy Card in April, 1992 especially for Linux Operating System. It is the first version in the series of extended file systems. It was replaced by ext2 in 1993. The ext file-system uses a system called inodes to track information about the files stored in the virtual directory. The inode system creates a separate table on each physical device, called the inode table, to store the file information. Each stored file in the virtual directory has an entry in the inode table. Linux maintains each inode in the inode table with a unique number (called the inode number), assigned by the file system.

5.3.2 ext2

ext2 is meant for second extended file system created for Linux OS and it is an improved version of ext. It was also designed by Remy Card. It is one of the most popular and commonly used file systems in Linux and has been used for many years. Here, the data stored in files are kept in data blocks. All the data blocks have same length. However, the length is not fixed in all ext2 file systems. The size of the file system is depended on the block size which 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 size of the filename may 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 which was introduced by Stephen Tweedie in November 2001 with Linux. It overcomes the limitation of ext2 file system by introducing the concept of journaling. Moreover, ext3 also adds two more features which are the growth of online file system and Tree indexing for larger directories. An ext3 file system works similar to ext2 file and an ext2 file system can easily be upgraded to ext3. It uses the same inode table structure as the ext2 filesystem, but adds a journal file to each storage device to journal the data written to the storage device. This 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 are 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. If a file system is crashed out, 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 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. This 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. 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

The result of expanding the ext3 file system is the ext4 file system. The ext4 file system was officially supported in the Linux kernel in 2008, and is now the default file system used in most popular Linux distributions, such as Fedora and Ubuntu. It was developed over ext3 in order to increase the size of the file and entire file system to improve the performances. The various 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 reinstallation 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, a program block allocator decides which free block will be used at the time of writing a new data. But it allows only one block at a time. So, it is not efficient technique for writing a file of large data size. To improve the performance, ext4 uses multiblock allocator which allows to use more than one free 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. In this case, block allocator can 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. Accordingly, 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 and on the other side, the file system has maximum size of 1 EiB or 1 ExbiBytes [1 EiB = 260 bytes].

5.3.5 xfs

xfs file system is a high-performance and highly scalable 64 bit system. It supports journaling and thereby it ensures the consistency of the data. It was created by Silicon Graphics and introduced in 1994 for its commercial IRIX Unix system. It was used in Linux

kernel in 2001. This file system is supported by almost all Linux distributions. The xfs file system uses the writeback mode of journaling, which provides high performance and also allows online resizing of the file system. 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 is considered one of the oldest journaling file systems. It is a 64-bit file system created by IBM. It's 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. JFS uses the ordered journaling method, storing only the inode table data in the journal, but not removing it until the actual file data is written to the storage device. In addition, JFS file system uses extent-based file allocation, allocating a group of blocks for each file written to the storage device. This method provides for less fragmentation on the storage device. It uses B+ tree to search in directories. Since ext4 shows better performance compared to JFS, so it is not widely used in Linux.

5.3.7 ReiserFS

In 2001, Hans Reiser created the first journaling file system for Linux, called ReiserFS. The ReiserFS file system only supports writeback journaling mode, writing only the inode table data to the journal file. Due to this feature, the ReiserFS file system is considered one of the fastest journaling file systems in Linux. Two interesting features available with the ReiserFS file system refer

the following.

- We can resize an existing file system while it's still active, and
- It uses a technique called tail packing, which stuffs data from one file into empty space in a data block from another file and thereby reduces internal fragmentation.

5.3.8 ISO 9660

ISO 9660 is a file system created for optical media. It is published by International Standard Organization (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. This file system standard evolved from the High Sierra Format file system. It has three different levels for files:

- **Level 1** - supports only 8.3 filenames (eight characters with a three-character extension).
- **Level 2** - supports longer 32-character filenames.
- **Level 3** - supports longer 32-character filenames.

5.4 LINUX FILE SYSTEM INTERNAL STRUCTURE

Linux files have disk partitions or blocks containing file data. Linux File System is divided mainly into 4 blocks.

- Boot block
- Super block
- inode
- Data Block

Partition is divided into data blocks with of fixed size which are used to store content of file. In case if file is of greater size than of block then it is stored in multiple blocks.



Boot Block: It is located in the first few sectors of a file system. The boot block contains the initial bootstrap program used to load the

operating system. Typically, the first sector contains a bootstrap program that reads in a larger bootstrap program from the next few sectors, and so forth. The boot block is usually a part of the disk label, a special set of blocks containing information on the disk layout. The boot block holds the loader to boot the operating system.

Super Block: Each filesystem has one super block. The superblock contains the basic information about the entire file system. This includes the size of the file system, the list of free and allocated blocks, the name of the partition, and the modification time of the filesystem.

Inodes: Information about each file in the filesystem is kept in a special kernel structure called an inode. The inode contains a pointer to the disk blocks containing the data in the file, as well as other information such as the type of file, the permission bits, the owner and group, the file size, the file modification time, and so on. The inode does not, however, contain the name of the file. The name of each file is listed in the directory the file is associated with. Every file has one inode and vice versa. In linux, every file is recognized with integer number known as inode number. 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.

Inodes include pointers to the data blocks. Each inode contains 15 pointers:

- the first 12 pointers point directly to data blocks
- the 13th pointer points to an indirect block, a block containing pointers to data blocks
- the 14th pointer points to a doubly-indirect block, a block containing 128 addresses of singly indirect blocks
- the 15th pointer points to a triply indirect block (which contains pointers to doubly indirect blocks, etc.)

Data Blocks: Data blocks start at the end of the inode list and contain file data and directory blocks. An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be of the same kind. The three types of data blocks are:

- Plain data blocks
- Symbolic-link data blocks
- Directory data blocks

Plain data blocks contain the information stored in a file. Symbolic-link data blocks contain the path name stored in a symbolic link. Directory data blocks contain directory entries. fsck can check the validity only of directory data blocks.

Directories are distinguished from regular files by an entry in the mode field of the inode. Data blocks associated with a directory contain the directory entries. Directory data blocks are checked for inconsistencies involving:

- Directory inode numbers pointing to unallocated inodes
- Directory inode numbers greater than the number of inodes in the file system
- Incorrect directory inode numbers for "." and ".." directories
- Directories disconnected from the file system

5.5 HOW LINUX ACCESS FILES

Linux file system is generally a built-in layer of Linux operating system used to handle the data management of the storage. It helps to arrange the file on the disk storage. It manages the file name, file size, creation date, and much more information about a file. A file system is designed in a way so that it can manage and provide space for non-volatile storage data. All file systems required a namespace that is a naming and organizational methodology.

In Linux, the file system creates a tree structure. All the files are arranged as a tree and its branches. The topmost directory called the root (/) directory. All other directories in Linux can be accessed from the root directory.

Some key features of Linux file system are as following:

- **Specifying paths:** Linux does not use the backslash (\) to separate the components; it uses forward slash (/) as an alternative. For example, as in Windows, the data may be stored in C:\ My Documents\ Work, whereas, in Linux, it would be stored in /home/ My Document/ Work.
- **Partition, Directories, and Drives:** Linux does not use drive letters to organize the drive as Windows does. In Linux, we cannot tell whether we are addressing a partition, a network device, or an "ordinary" directory and a Drive.
- **Case Sensitivity:** Linux file system is case sensitive. It distinguishes between lowercase and uppercase file names. Such as, there is a difference between test.txt and Test.txt in Linux. This rule is also applied for directories and Linux commands.
- **File Extensions:** In Linux, a file may have the extension '.txt,' but it is not necessary that a file should have a file extension. While working with Shell, it creates some problems for the beginners to differentiate between files and directories. If we use the graphical file manager, it symbolizes the files and folders.
- **Hidden files:** Linux distinguishes between standard files and hidden files, mostly the configuration files are hidden in Linux OS. Usually, we

don't need to access or read the hidden files. The hidden files in Linux are represented by a dot (.) before the file name. To access the files, we need to change the view in the file manager or need to use a specific command in the shell.

5.6 STORAGE OF FILES

File systems are defined based on where they are used. There are file systems defined for operating systems, networks, databases, and other special-purpose file systems. A filesystem is designed to provide space for non-volatile storage of data. All filesystems need to provide a namespace with a name and organizational methodology. This defines how a file can be named, specifically the length of a filename and the subset of characters that can be used for filenames out of the total set of characters available. It also defines the logical structure of the data on a disk, such as the use of directories for organizing files instead of just lumping them all together in a single, huge conglomeration of files.

Once the namespace has been defined, a metadata structure is necessary to provide the logical foundation for that namespace. This includes the data structures required to support a hierarchical directory structure; structures to determine which blocks of space on the disk are used and which are available; structures that allow for maintaining the names of the files and directories; information about the files such as their size and times they were created, modified or last accessed; and the location or locations of the data belonging to the file on the disk. Other metadata is used to store high-level information about the subdivisions of the disk, such as logical volumes and partitions. This higher-level metadata and the structures, it represents, contain the information describing the filesystem stored on the drive or partition, but is separate from and independent of the filesystem metadata.

File systems also require an Application Programming Interface (API) that provides access to system function calls which manipulate filesystem objects like files and directories. APIs provide for tasks such as

creating, moving, and deleting files. It also provides algorithms that determine things like where a file is placed on a filesystem. Such algorithms may account for objectives such as speed or minimizing disk fragmentation. Modern filesystems also provide a security model, which is a scheme for defining access rights to files and directories. The Linux filesystem security model helps to ensure that users only have access to their own files and not those of others or the operating system itself. The final building block is the software required to implement all of these functions. Linux uses a two-part software implementation as a way to improve both system and programmer efficiency.

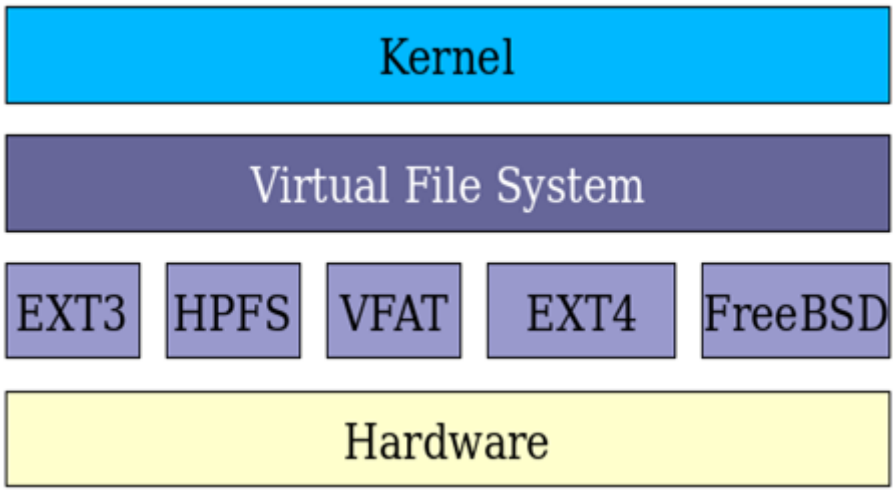


Fig. 5.1: The Linux two-part filesystem software implementation.

The first part of this two-part implementation is the Linux Virtual File System. This virtual file system provides a single set of commands for the kernel, and developers, to access all types of file systems. The virtual file system software calls the specific device driver required to interface to the various types of file systems. The file system-specific device drivers are the second part of the implementation. The device driver interprets the standard set of file system commands to ones specific to the type of file system on the partition or logical volume.

5.7 LINUX DIRECTORY STRUCTURE

The File System Hierarchy Standard (FHS) defines the directory structure and directory contents in Linux distributions. It is maintained by

the Linux Foundation. In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices. Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.

Most of these directories exist in all Unix-like operating systems and are generally used in much the same way. However, the descriptions here are those used specifically for the FHS, and are not considered authoritative for platforms other than Linux. Some standard Linux directories with their description are listed in the following table (Table-5.1).

Directory	Description
/	Primary hierarchy root and root directory of the entire file system hierarchy
/bin	Essential command binaries that need to be available in single user mode; for all users, *e.g.*, cat, ls, cp.
/boot	Boot loader files, *e.g.*, kernels, initrd.
/dev	Device files, *e.g.*, /dev/null , /dev/disk0 , /dev/sda1 , /dev/tty , /dev/random.
/etc	Host-specific system-wide configuration files.
/home	Users' home directories, containing saved files, personal settings, etc.
/sys	Contains information about devices, drivers, and some kernel features.
/usr	contains the majority of (multi-)user utilities and applications
/var	Contains Variable files such as logs, spool files, and temporary e-mail files.

Table 5.1: Few standard Linux directories with description

What the file system looks like is represented graphically in the following figure 5.2. Some of these directories have standard files as well like /bin, /etc and /sbin specifically.

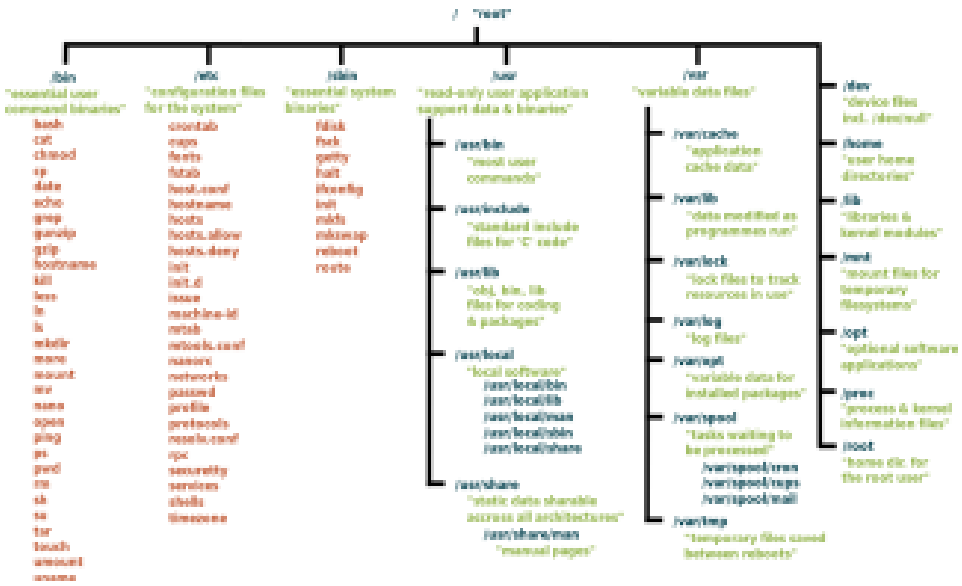


Fig. 5.2 Linux Directory structure

5.8 LILO, GRUB BOOT LOADER

A boot loader refers to a program which is responsible for loading the Linux kernel with optional kernel parameters and the Linux initial RAM disk, known as initrd. Linux kernel is the core of the Linux operating system, and it initiates the init process immediately after being loaded. The Linux initial RAM disk provides a temporary storage space for loading critical files into memory before the real root file system can be mounted. On older computers with BIOS (Basic Input/Output System), a boot loader resides in the MBR (Master Boot Record), which occupies the first 512 bytes on a disk, but newer computers with UEFI (Unified Extensible Firmware Interface) store it in a special partition called EFI System Partition.

A boot loader is loaded by BIOS (Basic Input Output System) after a successful POST (Power-On Self-Test), which is a self-test process performed immediately after a computer or other digital electronic device is powered on. There are several boot loaders in Linux system. Among them, two mostly used Linux loader programs are described here.

LILO (Linux Loader)

LILO once used to be the most popular Linux boot loader, but it has gradually fallen from beauty because it lacked support for multi-boot

environments. The development of LILO was stopped in December 2015, and the limited support for modern file systems and non-existing support for UEFI both reflect this reality.

GRUB (GRand Unified Bootloader)

GRUB is the most popular and the most feature-packed boot loader for the Linux operating system. It is based on the obsolete GRUB Legacy, which was created in 1995 by Erich Boleyn for the operating system GNU/ Hurd. GRUB supports both BIOS and UEFI, and it can handle all popular Linux file systems, including ext4, ReiserFS, VFAT, and XFS. Compared to GRUB Legacy, modern GRUB is cleaner, more powerful, and safer. GRUB has been designed to be compatible with the multiboot specification which allows GRUB to boot many versions of Linux and other free operating systems.

GRUB can also allow the user to choose to boot from among several different kernels for any given Linux distribution. This affords the ability to boot to a previous kernel version if an updated one fails somehow or is incompatible with an important piece of software. GRUB can be configured using the /boot/grub/grub.conf file.

5.9 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.

5.9.1 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.

5.9.2 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 below:

mkdir : used for creating a new directory.

Syntax: mkdir <directory name>

rmdir: used for removing or deleting any empty directory from the system.

Syntax: rmdir <directory name>

cd: used for changing the directory.

cd <directory name>

5.9.3 Special or Device Files

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 -|/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 located in /dev directory, following Linux

command can be used:

ls -|/dev|grep ^b



CHECK YOUR PROGRESS

Q 1: Choose the correct options from the followings:

i) Which among the following allows fast file system recovery?

a) Ext2 b) Journaling c) Caching d) Sysfs

ii) Filesystem for CDROM is:

a) Ext2 b) Ext3 c) ISOfs d) Procfs

iii) Which file contains the filesystems to be automatically mounted during boot?

a) /etc/mount b) /etc/fstab c) /etc/inittab d) /etc/boot

iv) Which command is used to check filesystem usage in a system?

a) mount b) df c) du d) dd

v) In which of the following directory does the configuration files are present?

a) /bin/ b) /root/ c) /etc/ d) /dev/

5.10 OTHER COMMON TYPES OF FILES

Apart from these three types of files, Linux system offers some other type of files which are:

- **Symbolic Links:** 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', 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 located in /dev directory, following Linux command can be used:

ls -l /dev | grep ^s

5.11 FILE ATTRIBUTES

Linux system is a typical multi-user system. Different users in a

different position, with different privileges can access Linux OS. 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. To protect the security of the system, Linux system provides file permissions for different users to do different provisions to access the same files (including the catalogue file). The files and directories in Linux can have following attributes:

- **a - append only:** this attribute allows a file to be added to, but not to be removed. It prevents accidental or malicious changes to files that record data, such as log files.
- **c - compressed:** it causes the kernel to compress data written to the file automatically and uncompress it when it is read back.
- **d - no dump:** it makes sure the file is not backed up in backups where the dump utility is used
- **e - extent format:** it indicates that the file is using extents for mapping the blocks on disk.
- **i - immutable:** it makes a file immutable, which goes a step beyond simply disabling write access to the file. The file can't be deleted, links to it can't be created, and the file can't be renamed.
- **j - data journaling:** it ensures that on an Ext3 file system the file is first written to the journal and only after that to the data blocks on the hard disk.
- **s - secure deletion:** it makes sure that recovery of a file is not possible after it has been deleted.
- **t - no tail-merging:** Tail-merging is a process in which small data pieces at a file's end that don't fill a complete block are merged with similar pieces of data from other files.
- **u - undeletable:** When a file is deleted, its contents are saved which allows a utility to be developed that works with that information to salvage deleted files.
- **A - no atime updates:** Linux won't update the access time stamp when you access a file.

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

Linux provides us the access control by file and directory permissions on three levels which are user, group and other. These file permissions provide the basic level of security and access control. In Linux, we can use the ls -l command to display user and group attributes of a file and the file belongs to, such as:

```
[root@www /]# ls -l
total 64
dr-xr-xr-x   2 root root 4096 Dec 14  2012 bin
dr-xr-xr-x   4 root root 4096 Apr 19  2012 boot
```

Example, the first attribute in bin and boot file with the "d" represents that it is a directory file.

In Linux, the first character represents the file as a directory, the ordinary file or linked files or device file such as

- the [d] is the directory
- the [-] is the ordinary file;
- If it is [l] then it refers to a linked document (link file);
- If it is [b], it indicates that it is a block device;
- the [c] represents the character device such as a keyboard, a mouse (one reading device).

The next character in groups of three, and are a combination of "rwx" of the three parameters. Wherein, [r] represents read permission, [w] represents write and [x] means execute permission. It should be noted that the position of these three rights will not change. If there is no authority, there will be a minus sign [-] only.



CHECK YOUR PROGRESS

Q 2: Choose the correct options from the followings:

i) Which of the following is true about files?

A) All data in Unix is organized into files.

B) All files are organized into directories.

C) The directories are organized into a tree-like structure called the filesystem.

D) All of the above

ii) Each entry of directory file has component(s) namely _____.

A) filename B) inode number C) file size

D) Both A and B

iii) Which of the following symbol represents the current directory?

A) Double dot (..) B) Dollar dot($.) C) Single dot (.)

D) Hash dot(#.)

iv) The directory files have:

A) File names & File location B) File names & Inode Numbers

C) File names & Permissions

D) File names & Address v) Device files have following two types?

A) Character & Block B) Character & Socket

C) Block & FIFO D) Only A



5.12 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 called device file is located in /dev directory of the file system. Character device and block device are two types of special files.
- A socket and named pipe file are 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. The inode files can be displayed by using ls -i command.
- File attributes are some setting options which allow or restrict specific operations on a file.



5.13 FURTHER READINGS

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."



5.14 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: i. b ii. c iii. b iv. b v. c

Ans to Q No 2: i. D ii. D iii. C iv. B v. D



5.15 MODEL QUESTIONS

Q 1: What are the purposes 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:

- Hard link vs. Soft link
- Named pipe file vs. Socket files
- 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:

- xfs
- JFS
- ReiserFS.

Q 13: Define boot loader. Explain various boot loaders used in Linux system.

Q 14: Discuss about Linux directory structure.

*** ***** ***

UNIT 6: FILE ORGANIZATION

UNIT STRUCTURE

6.1 Learning Objectives

6.2 Introduction

6.3 Files and File System

6.4 Working with Files

6.5 File System Mounting and Unmounting,

6.6 Organization of the File Tree

6.7 Disk Related Commands

6.8 Let Us Sum Up

6.9 Further Readings

6.10 Answer to Check Your Progress

6.11 Model Questions

6.1 LEARNING OBJECTIVES

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

- to distinguish different files system,
- to cope up with different commands to work on file,
- understand basic file organization of Linux,
- to learn different disk related commands.

6.2 INTRODUCTION

In the previous unit, we have discussed the Linux File System and the related properties. The functions of different file systems in Linux are explained. We have also discussed the different types of files, I-node and the file attributes. In this unit, we will discuss about the file organisation. File is collection of information, and a file system is used to control how data is stored and retrieved. Without a file system, data placed in a storage medium would be a random assortment of some facts only, without having any specific beginning or ending for a particular data. Unix system treat everything inside it as files. Hence it has thousands of files. Unix file attributes

like name, size and access permission are stored separately rather than putting in the file like in Windows system.

6.3 FILES AND FILE SYSTEM

In general, we can categorize files into 3 categories:

- **Ordinary files:** These are regular files which contain stream of characters as data. This can be further subdivided it into Text file and Binary File
- **Directory files:** Unix system treat directories are as files. A directory file contains an entry for every file and subdirectory it has. Each entry consists of two entry the file and unique identification number (inode number) for the file or directory
- **Device files:** In Unix all devices and peripherals are also treat as files. To perform any operation on devices we must invoke associated with files. Device fil name are generally store under single directory structure called, /dev.

A filename in Unix can consist of up to 255 characters. Filename can be any ASCII character except the / and NULL character. But we should avoid using control characters in naming files.

All files in UNIX are related to one another and organized in hierarchical inverted tree structure. Every UNIX file system has a reference point at top of the tree structure which is called root and represented as / (forward slash). Under root we have a directory called home directory. Whenever we log on to UNIX system it automatically places us at home directory. for example, if you log into your system and open Unix Terminal (ctrl+Alt+T for Ubuntu users) and type pwd after the dollar symbol you will get a result like this:

/home/host_name

Here in the above, first slash "/" is synonym for root directory, inside the root directory we have home directory inside it we have our own directory here directory name is Ubuntu, Home directory and ubuntu directory is separated by the delimiter "/". The pwd (print working directory) command shows you the current working directory.

```
ubuntu@ubuntu:~$ pwd
/home/ubuntu
```

Fig 6.1: Snapshot of pwd command

We can use ls (listing command) to list out all the content present in the current directory. It will show a complete list of filenames in the current directory in ASCII collating sequence (number first, uppercase then lower case). It shows directories also.

```
ubuntu@ubuntu:~$ ls
Desktop    Downloads  Pictures  Templates  test
Documents  Music      Public    Videos     test0.txt
ubuntu@ubuntu:~$
```

Fig 6.2: Snapshot of ls command

At above picture it is showing contents of ubuntu (user directory) where all are directories except test0.txt which is text file

ls Options

ls command has different options which syntax is like ls -Options. We will present some important them.

Type ls -F in a terminal window to add "/" at the end of each directory. This command will help you to distinguish directories from files. See image below for further details:

```
ubuntu@ubuntu:~$ ls -F
Desktop/    Downloads/  Pictures/  Templates/  test/
Documents/  Music/      Public/    Videos/     test0.txt
ubuntu@ubuntu:~$
```

Fig 6.2: Snapshot of ls-F command

When using the basic ls command, you can't see hidden files and files starting with ".". But the ls -a command to display them as seen in the image below:

```
ubuntu@ubuntu:~$ ls -a
.              .cache    .profile                    Downloads  Templates
..             .config   .sudo_as_admin_successful    Music      Videos
.bash_logout   .gnupg    Desktop                     Pictures   test
.bashrc        .local    Documents                   Public     test0.txt
ubuntu@ubuntu:~$
```

Fig 6.3: Snapshot of ls -a command

The get information about the user, size of the file, and date and time of modification. We can use ls -l command. In the picture bellow.

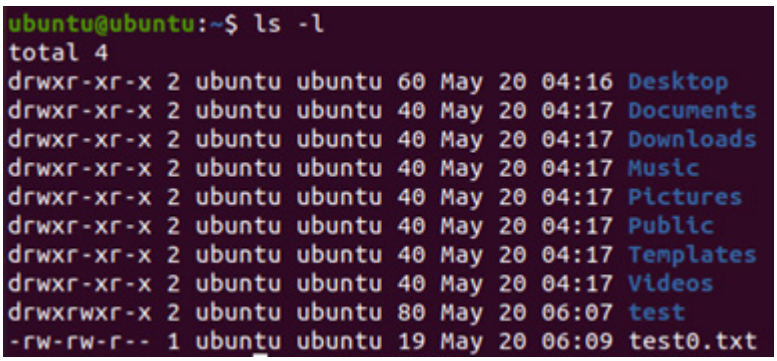


Fig 6.4: Snapshot of ls-l command

Let's us discuss this output little bit more elaborately. If we take the last line of the output, it looks like this:

-rw-rw-r-1 ubuntu ubuntu 19 May 20 06:09 test0.txt

1st Character - File Type: First character specifies the type of the file. In the example above the hyphen (-) in the 1st character indicates that this is a normal file. Following are the possible file type options in the 1st character of the ls -l output.

Field Explanation

- - normal file
- d : directory
- s : socket file
- l : link file
- **Field 1 - File Permissions:** Next 9 character specifies the files permission. The every 3 characters specifies read, write, execute permissions for user(root), group and others respectively in order. Taking above example, -rw-rw-r- indicates read-write permission for user(root) , read permission for group, and no permission for others respectively. If all three permissions are given to user(root), group and others, the format looks like -rwxrwxrwx
- **Field 2 - Number of links:** Second field specifies the number of links for that file. In this example, 1 indicates only one link to this file.
- **Field 3 - Owner:** Third field specifies owner of the file. In this example, this file is owned by username 'ubuntu'.
- **Field 4 - Group:** Fourth field specifies the group of the file. In this example, this file belongs to" ubuntu' group.

- **Field 5 - Size:** Fifth field specifies the size of file in bytes. In this example, '19' indicates the file size in bytes.
- **Field 6 - Last modified date and time:** Sixth field specifies the date and time of the last modification of the file. In this example, 'May 20 06:09' specifies the last modification time of the file.
- **Field 7 - File name:** The last field is the name of the file. In this example, the file name is test0.txt



CHECK YOUR PROGRESS

Q 1: What are the 3 types of file present in Linux?

.....

Q 2: Why ls command is used?

.....

6.4 WORKING WITH FILES

In this section we will try to explore different operations like creating, reading, searching, copying, moving, renaming, deleting, linking, and editing Files. First, we will go through operations of directory then we will go files as directory operations are little bit different.

Changing Directories

The pwd (print working directory) shows current directory we are in, and cd (change directory) is used switch your working directory from one to another. For more illustration look at the picture bellow. As per pwd command shows we are at first home directory "ubuntu", now we can use: cd pathname to move to different directory. Here we use cd test to move to directory test. We can use cd .. command to move out of the directory. After using this command, we can we have again come to "ubuntu" directory. Actually, the "." (dot) symbol refers to the working directory itself and ".." the (dot dot) symbol refers to the parent directory of the working directory

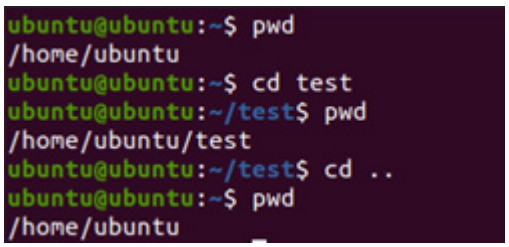


Fig 6.4: Snapshot of cd command

Creating and Deleting Directories

In order to create a directory, we use the mkdir (make directory) command, syntax for this command is mkdir directory_name. It will create a new directory in current directory. To remove a directory, we use the rmdir command. Remember that the rmdir command can only be used to remove empty directories. We can use rm (remove) command to delete directory, but it can also delete empty directories but with -rf option i.e.rm -rf directory_name We can force to delete the directory.



Fig 6.5: Snapshot of mkdir & rm -rf command

File Operations

we can create delete edit files using Linux GUI itself as like in windows. There is a general-purpose text editor for genome desktop called "gedit "with which we can create, edit and delete files. The general syntax is:

$gedit filename

For more details we can look into Linux manual i.e., $man gedit. Next, we have discussed some other means of file operations. All the operations have lots of different options, which can be used to do extra functionalities of the commands. But here we have only use straight forward utility of the commands only. We urged users to explore further possibilities with these commands

The touch command can be used to create a new file. It will create and open a new blank file if the file with a filename does not exist. And in

case the file already exists then the file will not be affected.

$touch filename

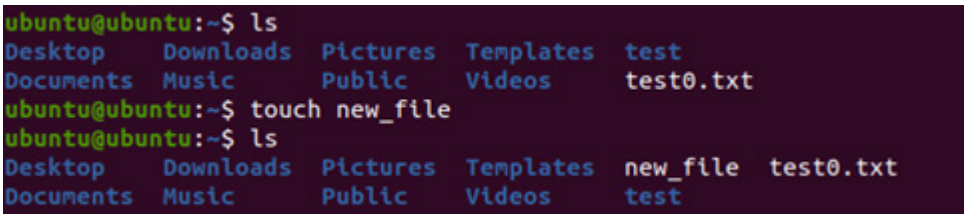


Fig 6.6: Snapshot of touch command

In above picture with ls command, we can see there is no "new_file" in directory, but with touch command we have created the file.

cat command can be used to display the contents of a file. This command will display the contents of the 'filename' file. And if the output is very large then we could use more or less to fit the output on the terminal screen otherwise the content of the whole file is displayed at once.

Syntax for cat command is:

$cat [Option] filename

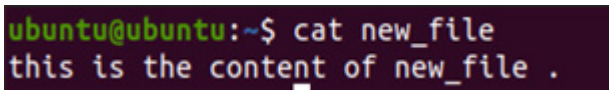


Fig 6.7: Use of cat command

The cat is a very powerful tool user can look for its more uses. For example:

$cat > filename creates a new file

$cat filename1 filename2>filename3 joins two files (1 and 2) and stores the output of them in a new file (3)

Copying a File

cp command could be used to create the copy of a file. It will create the new file in destination with the same name and content as that of the file 'filename'. It remains in the same working directory. General syntax for cp is:

$cp [Option] source/filename destination/filename

First image is copy without path name and second one is copy with a pathname.

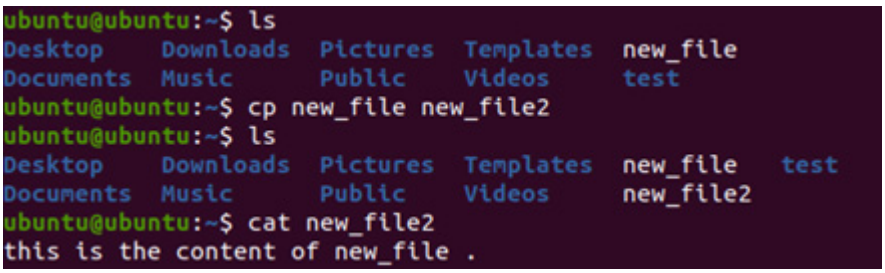


Fig 6.8: Snapshot of copy without path name

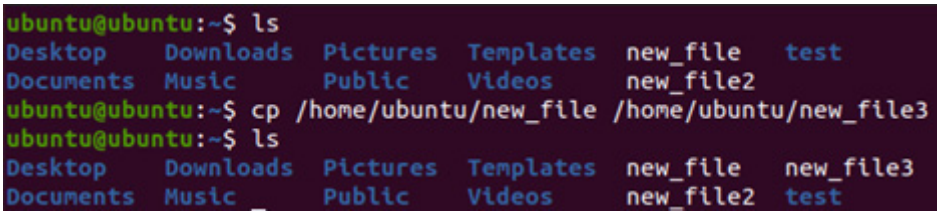


Fig 6.9: Snapshot of copy with path name

mv command could be used to move a file from source to destination. It will remove the file filename from the source folder and would be creating a file with the same name and content in the destination folder.

$mv [Option] source/filename destination/

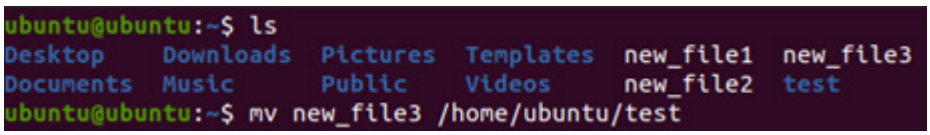


Fig 6.10: Snapshot of moving a file

In this screen shot we have move "new_file3" to "test" directory

rm command could be used to delete a file. It will remove the filename file from the directory. The general syntax for it is

$rm [Option] filename

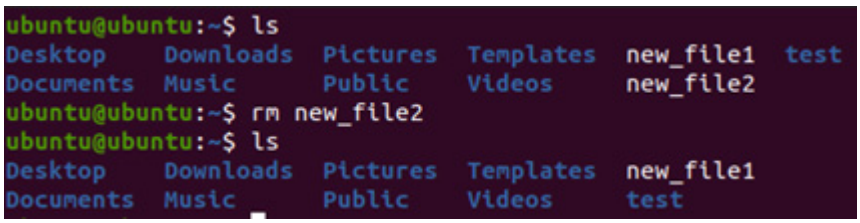


Fig 6.11: Snapshot of removing a file

The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the '-

exec' other UNIX commands can be executed on files or folders found.

$ find /directory where to search [expression determines what to find] [-options] [what to find]

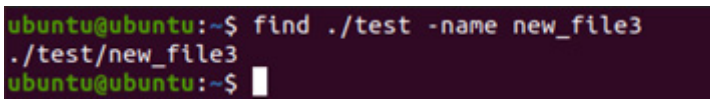


Fig 6.12: Snapshot of find command

Here we search new_file3 from the test directory using name option in find command.

Absolute Path vs Relative Path

A path is a unique location to a file or a folder in a file system of an OS. A path to a file is a combination of / and alpha-numeric characters. An absolute path is defined as the specifying the location of a file or directory from the root directory(/). In other words we can say absolute path is a complete path from start of actual filesystem from / directory. For example

/home/ubuntu/test

/home//boot/grub/grub.conf

all these paths started from / directory which is a root directory for every Linux/Unix machines

Relative path is defined as path related to the present working directory(pwd). Suppose at the moment we are at /home/ubuntu, we want to change it to /home/ubuntu/test, then we can directly change it t test using cd test command. If you observe there is no / before test which indicates it is a relative directory to present working directory. We can use an absolute path from any location where as if you want to use relative path we should be present in a directory where we are going to specify relative to that present working directory.

6.5 FILE SYSTEM MOUNTING AND UNMOUNTING

All files in a Linux filesystem are arranged in form of a big tree rooted at '/'.We will discussed this organization mor thoroughly later in this chapter .These files can be spread out on various devices based on partition table, initially parent directory is mounted(i.e attached) to this tree at '/',

others can be mounted manually using GUI interface(if available) or using mount command. The mount command is used to mount the filesystem found on a device to big tree structure (Linux filesystem) rooted at '/'. Conversely, another command umount can be used to detach these devices from the Tree.

Syntax:

mount -t type device dir

Other forms:

mount [-ll-hl-V]

mount -a [-fFnrsvw] [-t fstype] [-O optlist]

mount [-fnrsvw] [-o options] deviceldir

mount [-fnrsvw] [-t fstype] [-o options] device dir

All these commands tell the Kernel to attach the filesystem found at device to the dir.

Note:

- If you leave the dir part of syntax it looks for a mount point in /etc/fstab.
- You can use -source or -target to avoid ambivalent interpretation. mount --target /mountpoint
- /etc/fstab usually contains information about which device needs to be mounted where.
- Most of the devices are indicated by files like /dev/sda4, etc. But it can be different for certain filesystems. For more information explore man mount

Some Important Options:

- l : Lists all the file systems mounted yet.
- h : Displays options for command.
- V : Displays the version information.
- a : Mounts all devices described at /etc/fstab.
- t : Type of filesystem device uses.
- T : Describes an alternative fstab file.
- r : Read-only mode mounted.

Example: **mount dev/sd4 /home/ubuntu/test**

Here we have mounted device/sd4 file system to test directory.

6.6 ORGANIZATION OF THE FILE TREE

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation. In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices. So it looks like a tree organization where root '/'is the top directory it contain first level subdirectories and each first level sub directories contain second level subdirectories, hence the tree structure propagate. The picture below depicts how directories are organized in tree structure

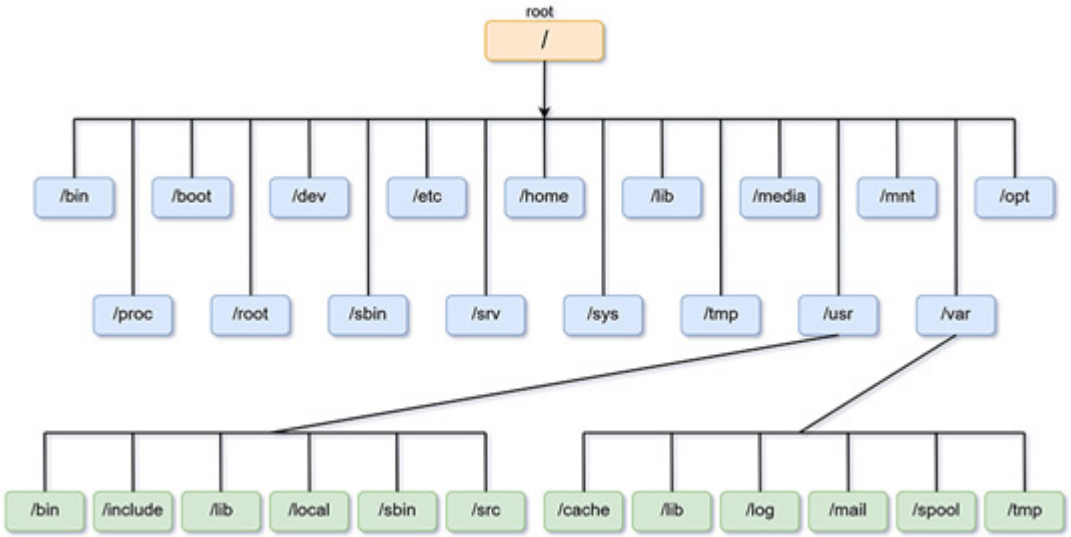


Fig 6.13: Linux File Structure Tree Organization

/ (Root): Primary hierarchy root and root directory of the entire file system hierarchy. Every single file and directory start from the root directory. The only root user has the right to write under this directory. the /root is the root user's home directory, which is not the same as /. Under root we have following subdirectories

- bin - This directory contains the binary or executable files that launch applications, all users have access to these applications. These represent the bare minimum set of programs required for a user to use the system.
- boot - This directory contains the files necessary to boot up your

computer including the boot loader and kernel activation files.

- dev - This directory contains the file pointers for character and block devices, which means the pointers to hard drives, flash drives, multimedia cards, external devices and system ports.
- etc - this directory contains the system configuration files and scripts.
- home - This directory contains the user profiles, desktops and user files, it functionality is similar to the "Documents and Settings" directory in MS-Windows.
- lib - This directory contains library files that are necessary for all programs to work.
- media - This directory is used by hal, the Linux auto-mounter, to load external devices for navigation such as floppy disks, cd-roms, dvd-roms, flash drives, etc…
- mnt - This directory is used to load external devices on system that do not use hal.
- opt - This directory contains optional software packages. Usually the contents of X11 and your window manger and loaded into opt, but you can add whatever programs you wish to this directory.
- proc - This directory is unique because it is not necessarily part of the file system, but a virtual file system that provides access to kernel or system information.
- root - This is the home directory for your root or system administrator. It is kept separate from the other users in case the partition that home is mounted on fails.
- sbin - This directory has a functionality similar to bin, which means it contains application executables, but rather than allowing all users access to these programs it restricts access only to the root or administrator.
- tmp - This is the temporary storage location. All users have read and write rights to the contents of this directory.
- usr - This is the largest directory on a Linux system. Pretty much everything that does not belong in the other directories is placed here. The contents usually include program, documentation and the kernel

source code.

- var - This directory contains most of the frequently changing files such as logging files, cache file and record locks.

If "tree" utility is installed is installed at your machine, we can check tree organization of file system inn UNIX/LINUX systems. It is a recursive directory listing program that produces a depth-indented listing of files. With no arguments, tree lists the files in the current directory. When directory arguments are given, tree lists all the files or directories found in the given directories each in turn. Upon completion of listing all files and directories found, tree returns the total number of files and directories listed. There are options to change the characters used in the output, and to use colour output. The most general way of using the tree command: $tree and the out will be like bellow.

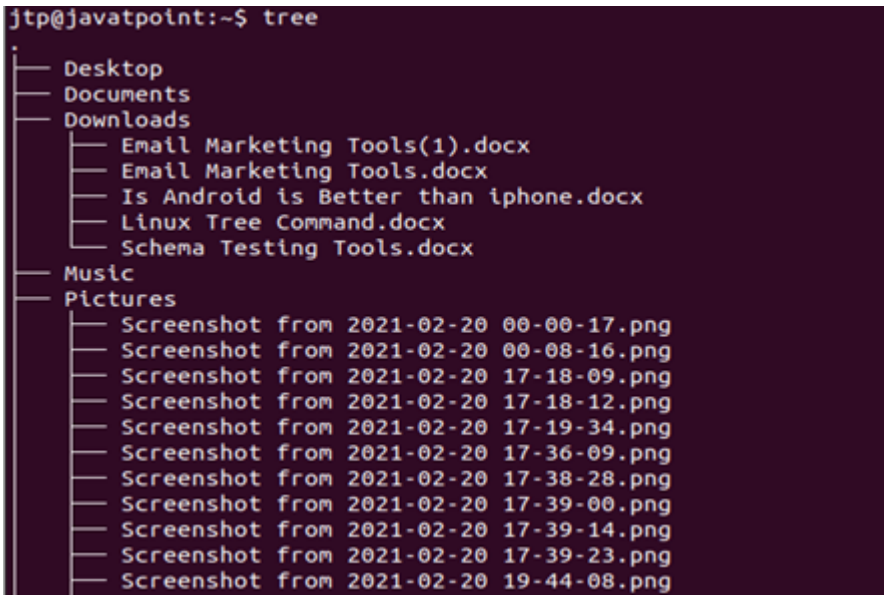


Fig 6.14: Output of Tree Command



CHECK YOUR PROGRESS

Q 3: How to check tree structure of Linux?

.....

Q 4: What is the command for mounting file in Linux system

.....

6.7 DISK RELATED COMMANDS

There are several ways to list all the hard drives present in a system through Linux command lines. A hard drive could be physically connected, virtually connected, or even emulated. Here are some different commands which can list the hard drives, keep in mind there are others, but these are probably the most commonly used and easy to get the job done.

The du (Disk Usage) command shows the usage of disk. The du command will show you the disk usage for specific directories in Linux. The syntax for this command is :

$ du [optios] FILESYSTEM

When we use this command with no parameter, then this command will show you the information related to all the mounted file systems

$du



Fig 6.15: Snapshot of du command

Some other Functionality of du command

- **du -h:** - This will show you the disk usage in a human-readable format for each directory and subdirectory
- **du -a:** - This will show you the disk usage for all the files.
- **du -s:** - This will show you the total disk space used by a specific directory or file.

The df (Disk free) command will show you used and available disk spaces for in Linux system. The syntax for this command is:

$ df [options] FILESYSTEM

When we use this command with no parameter, then this command will show you the information related to all the mounted file systems:

$df

```
ubuntu@ubuntu:~$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev              449040       0    449040   0% /dev
tmpfs             100232    1380     98852   2% /run
/dev/sr0         2999936 2999936         0 100% /cdrom
/dev/loop0       2109440 2109440         0 100% /rofs
/cow               501144  137112    364032  28% /
tmpfs              501144       0    501144   0% /dev/shm
tmpfs                5120       8      5112   1% /run/lock
tmpfs              501144       0    501144   0% /sys/fs/cgroup
tmpfs              501144       0    501144   0% /tmp
/dev/loop1          66688   66688         0 100% /snap/gtk-common-themes/1515
/dev/loop2          56832   56832         0 100% /snap/core18/2128
/dev/loop3          52224   52224         0 100% /snap/snap-store/547
/dev/loop4         224256  224256         0 100% /snap/gnome-3-34-1804/72
/dev/loop5          33152   33152         0 100% /snap/snapd/12704
tmpfs              100228      56    100172   1% /run/user/999
```

Fig 6.16: Snapshot output of df command

The following columns are contained in each line:

"Filesystem" - The filesystem's name.

"1K-blocks"- The filesystem's size in 1K blocks.

"Used"- Used means the used space in 1K blocks.

"Available" - Available means space available in 1K blocks.

"Use%"- The percentage of the used space.

"Mounted on"- The directory on which the filesystem is mounted/

Some options with df command

- 'df -h' option if we want to show information of all file system statistics in Gigabyte (GB).
- df-T option to tell df to display the file system types
- df -h If we want to show the details of disk drives in a human-readable format such as gigabytes, megabytes, kilobytes etc

fdisk can be used to manage disk partitions, you can create a new paritition, delete an existing partition…

the below let us see how to use fdisk command:

$ fdisk [options] FILESYSTEM

$fdisk -l

```
ubuntu@ubuntu:~$ sudo fdisk -l
Disk /dev/loop0: 2.1 GiB, 2160009216 bytes, 4218768 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop1: 65.1 MiB, 68259840 bytes, 133320 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Fig 6.17: Snapshot of fdisk -l command



6.8 LET US SUM UP

- In Linux everything is treated as file and the file system is a hierarchical tree structure where root in the first node
- Command associated to work with file is ls, cd, rm, cp, mv
- Absolute path names are full path name to the file whereas relative pathnames are relative to the working directory
- For mounting a file system in Linux system, we use mount command and for unmounting we use unmount command
- df and du are two commands associated with Linux disk utility
- Leaner must go through to Linux manual to grasp the total functionality of these commands



6.9 FURTHER READINGS

1) Das, S. (2000).UNIX: Concepts and Applications. Tata McGraw-Hill.
2) Das, S. (2012). Your UNIX/Linux: The Ultimate Guide. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
3) Christopher Negus (2012). Linux Bible. (8th Edition). Wiley.



6.10 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: 3 types of file: Ordinary file, Directory File , Device File

Ans to Q No 2: For listing of files.

Ans to Q No 3: $tree command

Ans to Q No 4: $ mount command



6.11 MODEL QUESTIONS

Q 1: Describe tree organization of Linux file system.

Q 2: Create a test file in your home directory write something it display it on monitor then move the file to another directory.

Q 3: Explain how commands for directory is different than command used in case of file system.

Q 4: Explain the output of ls-l command in details.

Q 5: Why we need to mount a filesystem in Linux?

Q 6: Use df and du command to check disk utility.

*** ***** ***

UNIT 7: LINUX FILE HANDLING COMMANDS

UNIT STRUCTURE

7.1 Learning Objectives

7.2 Introduction

7.3 File and Directory Handling Commands

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

7.4 Commands for Creating and Viewing Ordinary Files

- 7.4.1 cat
- 7.4.2 more
- 7.4.3 less

7.5 Let Us Sum Up

7.6 Further Readings

7.7 Answer to Check Your Progress

7.8 Model Questions

7.1 LEARNING OBJECTIVES

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

- learn various file and directory handling commands
- understand commands for creating and viewing ordinary files

7.2 INTRODUCTION

In the previous unit, we have discussed the organisation of the Linux file structure. We have also discussed various disk related commands. In this unit, we will discuss various File and Directory Handling Commands

with examples. We will also discuss the commands for creating and viewing ordinary files in the subsequent sections.

7.3 FILE AND DIRECTORY HANDLING COMMANDS

As you get started out in Linux, you'll spend a great deal of time interacting with files and directories. In the following sections, we will discuss various file and directory management commands that are used mostly by a system administrator or a normal Linux user.

7.3.1 ls

The ls is the list command in Linux. It shows the full list or content of your directory.

```
localhost:~# ls
bench.py    hello.c     hello.js    readme.txt
localhost:~#
```

Various options can be used along with the ls command. Let us discuss the options below:

- **ls -a command**

In Linux, hidden files start with a dot (.) and can't be seen in the regular directory. The ls -a command displays the whole list of a directory including the hidden files also.

```
localhost:~# ls -a
.             .ash_history  .mozilla      bench.py      hello.js
..            .cache        .wine         hello.c       readme.txt
localhost:~#
```

- ls -l command

It displays the list in a long list format.

```
localhost:~# ls -l
total 16
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~#
```

- ls -lh command

Size of the file is very difficult to read when displayed in terms of bytes. This command displays the data in terms of Mb, Gb, Tb, etc.

```
localhost:~# ls -lh
total 16K
-rw-r--r--    1 root     root         114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root          76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root          22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root         151 Jul  5  2020 readme.txt
localhost:~#
```

- ls -lhS command

This command displays the files in descending order (highest at the top) according to their size.

```
localhost:~# ls -lhS
total 16K
-rw-r--r--    1 root     root         151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root         114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root          76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root          22 Jun 26  2020 hello.js
localhost:~#
```

- ls -r

This command displays the list in reverse order.

```
localhost:~# ls -r
readme.txt  hello.js    hello.c     bench.py
localhost:~#
```

- ls -R

This command displays the subdirectories also.

```
localhost:~# ls -R
.:
bench.py    hello.c     hello.js    readme.txt
localhost:~#
```

- ls -lX

This command groups the files with same extensions together in the list.

```
localhost:~# ls -lX
total 16
-rw-r--r--    1 root     root          76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root          22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root         114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root         151 Jul  5  2020 readme.txt
localhost:~#
```

- ls - lt

This command sort the list by displaying recently modified filed at

top.

```
localhost:~# ls -lt
total 16
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
localhost:~#
```

- ls -~

It gives the contents of home directory.

```
localhost:~# ls ~
bench.py    hello.c     hello.js    readme.txt
localhost:~#
```

- ls ../

It displays the contents of parent directory.

```
localhost:~# ls ../
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
localhost:~#
```

7.3.2 cd

This command is used to change the directory. For example, if we want to change the current directory to the directory KKHSOU2, we have to type cd KKHSOU2

- To navigate into the root directory, use "cd /"
- To navigate to your home directory, use "cd" or "cd ~"
- To navigate up one directory level, use "cd .."
- To navigate to the previous directory (or back), use "cd -"

7.3.3 cp

This command is used to copy a file or a directory.

For example, to copy the file hello.c to hello_new.c in the same directory, we have to type cp hello.c hello_new.c

```
localhost:~# cp hello.c hello_new.c
localhost:~# ls
KKHSOU2       KKHSOU_NEW    hello.c       hello_new.c
KKHSOU3       bench.py      hello.js      readme.txt
localhost:~#
```

To copy a file or a directory to a different location, we have to mention the path of the destination directory.

- To copy multiple file or directories in a directory, we have to type cp file1 file 2 directory name

  For example, to copy the files hello.c and hello_new.c to the directory KKHSOU_NEW, the following command can be used.

```
localhost:~# cp hello.c hello_new.c KKHSOU_NEW
```

- cp -r command is used to copy a directory along with its sub directories.

  For example, the directory KKHSOU_NEW now contains two files named hello.c and hello_new.c. Let us copy the directory KKHSOU_NEW with all the files to the directory KKHSOU2 which is at present empty.

```
localhost:~# cp -r KKHSOU_NEW KKHSOU2
```

Now, to check the content of the directory KKHSOU2, we have to change the directory using cd command.

```
localhost:~# cp -r KKHSOU_NEW KKHSOU2
localhost:~# cd KKHSOU2
localhost:~/KKHSOU2# ls
KKHSOU_NEW
localhost:~/KKHSOU2#
```

7.3.5 rm

rm stands for remove and this command is used to remove a file. We have to be very careful while using this command as it permanently removes a file.

rm filename

```
localhost:~# ls
bench.py          file.tar.gz       hello.js          student_marks.txt
file.tar          hello.c           readme.txt        test.txt
localhost:~# rm file.tar.gz
localhost:~# ls
bench.py          hello.c           readme.txt        test.txt
file.tar          hello.js          student_marks.txt
localhost:~#
```

In the above example, you can see that we have removed/ deleted the file 'file.tar.gz' using the rm command.

- rm -r

With rm '-r' option, you can delete a directory having sub directories inside it. So you don't need to delete sub-directories manually.
The syntax is: rm -r directoryname

- rm -i

The option 'i' stands for 'interactivity'. Before deleting a file it will ask you whether you want to delete it or not. The syntax is:

rm -i filename

Similarly, to delete a directory 'interactively', the following command is used:

rm -ri directoryname

7.3.6 mkdir

To create a new directory, mkdir command is used. The mkdir stands for 'make directory'. For example, to create a new directory named KKHSOU, you have to type mkdir KKHSOU

```
localhost:~# mkdir KKHSOU
localhost:~#
```

The directory KKHSOU is created. To check whether the directory is created or not, type the ls command.

```
localhost:~# ls
KKHSOU      bench.py    hello.c     hello.js    readme.txt
localhost:~#
```

We can create multiple directories with a single command. Suppose, we want to create directories named KKHSOU1, KKHSOU2 and KKHSOU3

So, we have to type mkdir KKHSOU1 KKHSOU2 KKHSOU3

```
localhost:~# mkdir KKHSOU1 KKHSOU2 KKHSOU3
localhost:~# ls
KKHSOU      KKHSOU2     bench.py    hello.js
KKHSOU1     KKHSOU3     hello.c     readme.txt
localhost:~#
```

Note: By default, your directory will be created in your current

directory. If you want to create your directory somewhere else, then provide the path of your destination directory and your file will be created there.

7.3.7 rmdir

To delete a directory, rmdir command is used. But it can delete an empty directory only. If the directory contains sub-directories, it will not be able to delete the directory.

For example, to delete the directory KKHSOU, we have to type rmdir KKHSOU

```
localhost:~# rmdir KKHSOU
localhost:~# ls
KKHSOU1     KKHSOU3     hello.c     readme.txt
KKHSOU2     bench.py    hello.js
localhost:~#
```

7.3.8 pwd

This command displays the current working directory. It displays the whole path starting from the root ending to the directory.

```
localhost:~/KKHSOU2# pwd
/root/KKHSOU2
localhost:~/KKHSOU2#
```



CHECK YOUR PROGRESS

Q 1: _____ command displays the whole list of a directory including the hidden files.

Q 2: To display the whole list of content of a directory including the sub-directories also _____ command is used.

Q 3: To display the content of the parent directory, _____ command is used.

Q 4: _____ command is used to copy a directory along with its sub directories.

Q 5: _____ command displays the current working directory.

7.4 COMMANDS FOR CREATING AND VIEWING ORDINARY FILES

7.4.1 cat

The 'cat' command is one of the most frequently used commands in Linux. Let us discuss the various uses of the cat command.

- The 'cat' command can be used to display the content of a file. For example, to display the content of the file hello.c, we have to type

  cat hello.c

```
localhost:~# cat hello.c
#include <stdio.h>
int main(void)
{
        printf("hello world\n");
        return 0;
}

localhost:~#
```

To display the content of multiple files at once, type file names in one single line like "cat file1 file2 file3... filen".

For example, if we want to display the content of the files named bench.py and hello.c, we have to type the command as

cat bench.py hello.c

```
localhost:~# cat bench.py hello.c
def fib(n):
    if n < 2:
        return n
    return fib(n-2) + fib(n-1)

n = 25
print("fib(", n, ")= ", fib(n))
#include <stdio.h>
int main(void)
{
        printf("hello world\n");
        return 0;
}

localhost:~#
```

- The 'cat' command can be used to create a new file with greater

than sign (>).

For example, to create a new file named "file1.txt", we have to type the command

cat > file1.txt and press 'enter'. You will be directed to the next line. Press 'enter' after every line and you will be directed to the next line. To save your file, go to the next line, press 'ctrl+d' and your file will be saved.

```
localhost:~# cat > file1.txt
```

- The 'cat' command with double greater than sign (>>) append (add something in the last of a file) something in your already existing file.

```
localhost:~# cat >> file1.txt
This is line no1.
Welcome to KKHSOU.
```

Here, we have added two lines in the already existing file file1.txt. In this case, since the file1.txt is an empty file, the two lines will be appended. After typing the lines, press ctrl + d to save the file.

- The 'cat' command can be used to copy the content of a file into another file.

  For example, to copy the file readme.txt to another file named readme_copy.txt, we have to write the following command:

  cat readme.txt>readme_copy.txt

```
localhost:~# cat readme.txt > readme_copy.txt
localhost:~# cat readme_copy.txt
Some tests:

- Compile hello.c with gcc (or tcc):

  gcc hello.c -o hello
  ./hello

- Run QuickJS:

  qjs hello.js

- Run python:

  python3 bench.py
localhost:~#
```

- The 'cat' command can be used to concatenate the contents of multiple files in a single new file.

```
localhost:~# cat file1.txt hello.c > new_file.txt
```

In the above example, we have concatenate the files file1.txt and hello.c to a new file new_file.txt

Now, after concatenation, to view the content of the file new_file.txt, we have to write

cat > new_file.txt

```
localhost:~# cat new_file.txt
This is line no1.
Welcome to KKHSOU.

#include <stdio.h>
int main(void)
{
        printf("hello world\n");
        return 0;
}

localhost:~#
```

- The 'cat -n' option displays line numbers in front of each line in a file.

```
localhost:~# cat -n hello.c
     1  #include <stdio.h>
     2  int main(void)
     3  {
     4          printf("hello world\n");
     5          return 0;
     6  }
     7
localhost:~#
```

- The 'cat -b' option removes the empty lines.

```
localhost:~# cat -b hello.c
     1  #include <stdio.h>
     2  int main(void)
     3  {
     4          printf("hello world\n");
     5          return 0;
     6  }

localhost:~#
```

7.4.2 more

We have already learnt that 'cat' command can be used to display the content of a file. Similarly we can also use the 'more' command to display the content of a file. Now, the question may arise that what are the differences between the 'cat' and 'more' command while displaying the content of a file? The answer is that, in case of larger files, 'cat' command output will scroll off your screen while 'more' command displays output one screenful at a time.

For example, suppose we have a file named 'kkhsou.txt' which contains some long texts. Now, to display the content of the file 'kkhsou.txt', we have to type the command cat kkhsou.txt as shown below:

```
localhost:~# cat kkhsou.txt
Education is the backbone of a society. In common parlance education is any act or experience that has
 a formative effect on the mind, character or physical ability of an individual. In continuing with th
e aim of building a knowledge based society, Krishna Kanta Handiqui State Open University was establis
hed to providing quality higher education through the Open and Distance Learning to reach the unreache
d section of the society.

Education is the backbone of a society. In common parlance education is any act or experience that has
 a formative effect on the mind, character or physical ability of an individual. In continuing with th
e aim of building a knowledge based society, Krishna Kanta Handiqui State Open University was establis
hed to providing quality higher education through the Open and Distance Learning to reach the unreache
d section of the society.

KKHSOU, established under the provision of the KKHSOU, Act' 2005 enacted by the Govt. of Assam & publi
shed in the Assam Gazette (extraordinary) dt 29/9/05 has been recognized by the Former Distance Educat
ion Council, New Delhi (now DEB,UGC) vide letter no. DEC/Misc/07/5957 dated 03/10/07. The University G
rants Commission vide its letter No. F.9-13/2008(CPP-I) dated 18th March, 2009 also has empowered the
University to award degrees under Section 22 of UGC Act, 1956. It is the fourteenth of its kind along
with IGNOU and the only state Open University in the whole of NE India. The headquarters of the Univer
sity is located at Patgaon,Rani,Guwahati. The University started functioning with the joining of Prof.
 Srinath Baruah, former Professor, Deptt. of Economics, Gauhati University, as the Vice-Chancellor of
this University.

The University was established with the purpose of promoting education to reach the unreached through
the Open and Distance Learning System, with the motto of the University 'Education Beyond Barriers' of
 age, academic background and geographical boundaries.
```

You have seen that, we have to scroll to read the content of the file. Let us use the 'more' command to display the content of the same file 'kkhsou.txt' and see the differences.

- **more -num:** Limits the line displayed per page.(num is the total number of lines to be displayed)

```
localhost:~# more -10 kkhsou.txt
Education is the backbone of a society. In common parlance education is any act or experience that has
 a formative effect on the mind, character or physical ability of an individual. In continuing with th
e aim of building a knowledge based society, Krishna Kanta Handiqui State Open University was establis
hed to providing quality higher education through the Open and Distance Learning to reach the unreache
d section of the society.

Education is the backbone of a society. In common parlance education is any act or experience that has
 a formative effect on the mind, character or physical ability of an individual. In continuing with th
e aim of building a knowledge based society, Krishna Kanta Handiqui State Open University was establis
hed to providing quality higher education through the Open and Distance Learning to reach the unreache
localhost:~#
```

It displays the first 10 lines of the file 'kkhsou.txt'.

7.4.3 less

The 'less' command is same as 'more' command but include some more features.

It automatically adjust with the width and height of the teminal window, while 'more' command cuts the content as the width of the terminal window get shorter.

For example, less kkhsou.txt

```
KKHSOU, established under the provision of the KKHSOU, Act' 2005 enacted by the Govt. of Assam & pub
lished in the Assam Gazette (extraordinary) dt 29/9/05 has been recognized by the Former Distance Educ
ation Council, New Delhi (now DEB,UGC) vide letter no. DEC/Misc/07/5957 dated 03/10/07. The University
 Grants Commission vide its letter No. F.9-13/2008(CPP-I) dated 18th March, 2009 also has empowered th
e University to award degrees under Section 22 of UGC Act, 1956. It is the fourteenth of its kind alon
g with IGNOU and the only state Open University in the whole of NE India. The headquarters of the Univ
ersity is located at Patgaon,Rani,Guwahati. The University started functioning with the joining of Pro
f. Srinath Baruah, former Professor, Deptt. of Economics, Gauhati University, as the Vice-Chancellor o
f this University.

The University was established with the purpose of promoting education to reach the unreached through
the Open and Distance Learning System, with the motto of the University 'Education Beyond Barriers
 of age, academic background and geographical boundaries.

The main aim of the university is to develop and provide easily accessible modes of quality higher edu
cation and training with the use of latest educational inputs and technology. Because of the inherent
flexibility in terms of pace and place of learning, methods of evaluation, etc., The University holds
the promise of providing equality of opportunities for higher education and bringing into its fold the
 deprived and denied sections along with the fresh learners.

The programmes of the University are so designed that it can spread education to all the learners of t
he region.The course structures of the university have been designed at par with the national curricul
um. Along with the traditional programmes, the University offers various professional Certificate and
Diploma programmes to enable the already enrolled students in various colleges and educational institu
tions of the state to pursue add-on studies simultaneously with their regular courses.

:
```



7.5 LET US SUM UP

- ls command shows the full list or content of your directory.
- ls -l command displays the list in reverse order.
- ls -lX command groups the files with same extensions together in the

list.

- cp command is used to copy a file or a directory.
- cp -r command is used to copy a directory along with its sub directories.
- To rename a directory mv command is used.
- rm command permanently removes a file.
- You can delete a directory having sub directories inside it with the rm -r command.
- To create a new directory, mkdir command is used.



7.6 FURTHER READINGS

1) Das, S. (2000).UNIX: Concepts and Applications. Tata McGraw-Hill.
2) Das, S. (2012). Your UNIX/Linux: The Ultimate Guide. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
3) Christopher Negus (2012). Linux Bible. (8th Edition). Wiley.
4) https://www.javatpoint.com/linux-tutorial



7.7 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: ls -a

Ans to Q No 2: ls -R

Ans to Q No 3: ls ../

Ans to Q No 4: cp -r

Ans to Q No 5: pwd



7.8 MODEL QUESTIONS

Q 1: Create a directory named EMPLOYEE and DEPARTMENT.

Q 2: Copy a file to the directory EMPLOYEE.

Q 3: Rename the directory DEPARTMENT to DEPT.

Q 4: Change your current directory to a different directory and then display your current working directory.

*** ***** ***

UNIT 8: LINUX FILTER COMMANDS

UNIT STRUCTURE

8.1 Learning Objectives

8.2 Introduction

8.3 Filter Commands

- 8.3.1 wc
- 8.3.2 head
- 8.3.3 tail
- 8.3.4 cut
- 8.3.5 tr
- 8.3.6 top
- 8.3.7 grep

8.4 File Compression and System Monitoring Commands

- 8.4.1 tar
- 8.4.2 unzip
- 8.4.3 kill
- 8.4.4 netstat

8.5 File Access Permission Commands

- 8.5.1 chmod
- 8.5.2 chown
- 8.5.3 chgrp

8.6 Let Us Sum Up

8.7 Further Readings

8.8 Answer to Check Your Progress

8.9 Model Questions

8.1 LEARNING OBJECTIVES

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

- understand the use of wc command with various options
- learn head, tail, cut, tr and top commands
- search words in a file using the grep command

- understand to compress and decompress files
- work with various file access permission commands like chmod, chown and chgrp

8.2 INTRODUCTION

In the previous unit, we have discussed various file handling commands with examples. We have also discussed various commands for Creating and Viewing ordinary files. In this unit, we will discuss various filter commands and commands for creating and viewing ordinary files. In the last part of this unit, we will discuss various file access permission commands also.

8.3 FILTER COMMANDS

When a program takes its input from another program, it performs some operation on that input, and writes the result to the standard output. It is referred to as a filter. A large numbers of filter commands are available in Linux. In the next sections, we will discuss these commands with examples.

8.3.1 wc Command

The wc command displays the number of lines, number of characters, and the number of words in a file.

Example: To display the number of lines, words, and bytes from a file.

wc <filename>

```
localhost:~# wc kkhsou.txt
      31      933     6227 kkhsou.txt
localhost:~#
```

Example: Display the complete count information of multiple files.

wc <file1><file2>

```
localhost:~# wc hello.c kkhsou.txt
       7       10       76 hello.c
      31      933     6227 kkhsou.txt
      38      943     6303 total
localhost:~#
```

The above command displays the number of lines, words, and the number of the bytes from the files 'hello.c' and 'kkhsou.txt'. It also displays the total number of lines, words, and the number of the bytes of the two files.

Example: Display the number of lines in a file.

wc - l filename

```
localhost:~# wc -l kkhsou.txt
31 kkhsou.txt
localhost:~#
```

Example: Display the number of characters in a file

wc -m kkhsou.txt

```
localhost:~# wc -m kkhsou.txt
6215 kkhsou.txt
localhost:~#
```

Example: Display the number of bytes in a file

```
localhost:~# wc -c kkhsou.txt
6227 kkhsou.txt
localhost:~#
```

Example: Display the number of words in a file

wc -w kkhsou.txt

```
localhost:~# wc -w kkhsou.txt
933 kkhsou.txt
localhost:~#
```

Example: Display the total number of files and folders in a directory

ls l wc -l

```
localhost:~# ls | wc -l
8
localhost:~#
```

Example: Display the length of the longest line

wc -L kkhsou.txt

```
localhost:~# wc -L kkhsou.txt
941 kkhsou.txt
localhost:~#
```

8.3.2 head Command

The 'head' command displays the starting content of a file. By default, it displays starting 10 lines of any file.

head <filename>

```
localhost:~# head hello.c
#include <stdio.h>
int main(void)
{
        printf("hello world\n");
        return 0;
}

localhost:~#
```

Example: Display the first 10 lines of multiple files

head <file1><file2>

```
localhost:~# head hello.c readme.txt
==> hello.c <==
#include <stdio.h>
int main(void)
{
        printf("hello world\n");
        return 0;
}


==> readme.txt <==
Some tests:

- Compile hello.c with gcc (or tcc):

  gcc hello.c -o hello
  ./hello

- Run QuickJS:

  qjs hello.js
localhost:~#
```

In the above screenshot, we have seen that the lines are displayed from two files named 'hello.c' and 'readme.txt' with separate headings. (Since, the two files contains less than 10 lines, it displays the lines less than 10)

Example: Display specific number of lines from a file.

head -n filename

```
localhost:~# head -3 hello.c
#include <stdio.h>
int main(void)
{
localhost:~#
```

Example: Display specific number of bytes of a file.

head -c <number><filename>

```
localhost:~# head -c 220 kkhsou.txt
Education is the backbone of a society. In common parlance education is any act or experience that has
 a formative effect on the mind, character or physical ability of an individual. In continuing with th
e aim of buildinlocalhost:~#
```

In the above example, we have displayed the 220 bytes of content from the file 'kkhsou.txt'.

8.3.3 tail Command

Linux tail command is used to display the last ten lines of one or more files.

tail <filename>

```
localhost:~# tail test.txt
11
12
13
14
15
16
17
18
19
20
localhost:~#
```

Example: Display specific number of lines from last.

tail -n <number><file name>

```
localhost:~# tail -5 test.txt
16
17
18
19
20
localhost:~#
```

Example: Display specified number of bytes from the last.

tail -c <number><file name>

```
localhost:~# tail -10 test.txt
11
12
13
14
15
16
17
18
19
20
localhost:~#
```

From the above output, the last six bytes of the file 'test.txt' are displayed.

Example: Display the first 10 lines of multiple files

tail <file1><file2>

```
localhost:~# tail hello.c test.txt
==> hello.c <==
#include <stdio.h>
int main(void)
{
        printf("hello world\n");
        return 0;
}


==> test.txt <==
11
12
13
14
15
16
17
18
19
20
localhost:~#
```

8.3.4 cut Command

It is used to cut a specific section by byte position, character, and field and writes them to standard output. To cut a specific section, it is necessary to specify the delimiter and it can be a space (' '), a

hyphen (-), a slash (/), or anything else. After '-f' option, the column number is to be mentioned.

- To cut by using the hyphen (-) as the delimiter, execute the below command:

  cut -d- -f(columnNumber) <fileName>

  Let us explain with an example.

```
localhost:~# cat >> student_marks.txt
Rajib-90
Pranjit-67
Pradip-45
Pritam-78
Prakash-89
localhost:~# cut -d- -f2 student_marks.txt
90
67
45
78
89
localhost:~# cut -d- -f1 student_marks.txt
Rajib
Pranjit
Pradip
Pritam
Prakash
localhost:~#
```

Here, we have created a new file 'student_marks' and added the content of the marks obtained by students using 'cat' command. Then, we have displayed only the marks using the 'cut' command. In the next section, we have displayed the name of the students only.

- The '-b' option is used to cut a section of line by byte. To cut a file by its byte position, execute the command as follows:

  cut -b <byte number> <file name>

```
localhost:~# cat student_marks.txt
Rajib-90
Pranjit-67
Pradip-45
Pritam-78
Prakash-89
localhost:~# cut -b 5 student_marks.txt
b
j
i
a
a
localhost:~#
```

8.3.5 tr Command

The command 'tr' stands for 'translate'. It is used to translate, like from lowercase to uppercase and vice versa.

command I tr <'old'> <'new'>

```
localhost:~# cat student_marks.txt
Rajib-90
Pranjit-67
Pradip-45
Pritam-78
Prakash-89
localhost:~# cat student_marks.txt | tr 'rai' 'RAI'
RAjIb-90
PRAnjIt-67
PRAdIp-45
PRItAm-78
PRAkAsh-89
localhost:~#
```

8.3.6 top Command

It is used to display all the running process within the environment of a system. It helps in monitoring system usage and performances. It is mainly used to detect load on the server by system administrators.

Type the following command:

top

```
Mem: 5868K used, 1813768K free, 8K shrd, 0K buff, 872K cached
CPU:   0% usr   0% sys   0% nic  96% idle   0% io   0% irq   1% sirq
Load average: 0.00 0.00 0.00 1/72 151
  PID  PPID USER     STAT   VSZ %VSZ CPU %CPU COMMAND
  151    61 root     R     1516   1%   0   1% top
   61     1 root     S     1552   1%   0   0% sh -l
   97    61 root     T     1524   1%   0   0% cat
    1     0 root     S     1512   1%   0   0% {init} /bin/sh /sbin/init
   56     1 root     S     1260   1%   0   0% dhcpcd -q
   55     1 root     S      744   0%   0   0% settime -d /
    7     2 root     SW       0   0%   0   0% [ksoftirqd/0]
   15     2 root     SW       0   0%   0   0% [kworker/0:1]
   62     2 root     SW       0   0%   0   0% [kworker/u2:1]
   10     2 root     SW<      0   0%   0   0% [writeback]
   11     2 root     SW       0   0%   0   0% [kcompactd0]
   12     2 root     SW<      0   0%   0   0% [crypto]
   13     2 root     SW<      0   0%   0   0% [bioset]
   14     2 root     SW<      0   0%   0   0% [kblockd]
    6     2 root     SW<      0   0%   0   0% [mm_percpu_wq]
   16     2 root     SW       0   0%   0   0% [kswapd0]
   17     2 root     SW<      0   0%   0   0% [bioset]
   34     2 root     SW       0   0%   0   0% [khvcd]
   35     2 root     SW<      0   0%   0   0% [bioset]
   36     2 root     SW<      0   0%   0   0% [bioset]
   37     2 root     SW<      0   0%   0   0% [bioset]
   38     2 root     SW<      0   0%   0   0% [bioset]
   39     2 root     SW<      0   0%   0   0% [bioset]
   40     2 root     SW<      0   0%   0   0% [bioset]
    8     2 root     SW       0   0%   0   0% [kdevtmpfs]
    4     2 root     SW<      0   0%   0   0% [kworker/0:0H]
```

8.3.7 grep Command

The 'grep' command stands for "global regular expression print". It filters the content of a file which makes our search easy.

- The 'grep' command is generally used with pipe (|).

Syntax: command | grep <searchWord>

Example: cat student_marks.txt | grep 9

```
localhost:~# cat student_marks.txt
Rajib-90
Pranjit-67
Pradip-45
Pritam-78
Prakash-89
localhost:~# cat student_marks.txt | grep 9
Rajib-90
Prakash-89
localhost:~#
```

- The 'grep' command can also be used without the pipe (|).

Syntax: grep <searchWord> <file name>

Example: grep 9 student_marks.txt

```
localhost:~# cat student_marks.txt
Rajib-90
Pranjit-67
Pradip-45
Pritam-78
Prakash-89
localhost:~# cat student_marks.txt | grep 9
Rajib-90
Prakash-89
localhost:~#
```



CHECK YOUR PROGRESS

Q 1: The syntax to display the complete count information of multiple files is _____

Q 2: _____ command displays the number of lines in a file.

Q 3: wc -w kkhsou.txt command displays the number of _____ of the file 'kkhsou.txt'

Q 4: _____ command can be used to display the starting 10 lines of any file.

Q 5: _____ command is used to translate from lowercase to uppercase and vice versa.

Q 6: To search word in a file, _____ command is used.

8.4 FILE COMPRESSION AND SYSTEM MONITORING COMMANDS

We often need to compress files and directories to save space without compromising the quality. In Linux, few handy commands are available to achieve the task. In the next sections, we will discuss those commands. We will also discuss how to work with some system related commands like kill and netstat in the next sections.

8.4.1 tar Command

The tar command stands for tape archive. This command is used for creating Archive and extracting the archive files.

- Making an uncompressed tar archive with -cvf option

  The command is:

  tar cvf file.tar *.txt

```
localhost:~# tar cvf file.tar *.txt
readme.txt
student_marks.txt
test.txt
localhost:~#
```

The above example makes a tar file known as file.tar. It is the archive of every .txt file inside the current directory.

- Extracting files through the archive with -xvf option

  The command is:

  tar xvf file.tar

```
localhost:~# tar xvf file.tar
readme.txt
student_marks.txt
test.txt
localhost:~#
```

- gzip compression over tar archive with -z option

This option makes a tar file known as file.tar.gz. It is the archive of every .txt file.

The command is as follows:

tar cvzf file.tar.gz *.txt

```
localhost:~# tar cvzf file.tar.gz *.txt
readme.txt
student_marks.txt
test.txt
localhost:~# ls
bench.py          file.tar.gz       hello.js          student_marks.txt
file.tar          hello.c           readme.txt        test.txt
localhost:~#
```

- Extracting the gzip tar archive with -xvzf option

This option can extract the files through file.tar.gz tar archived files.

The command is as follows:

tar xvzf file.tar.gz

```
localhost:~# tar xvzf file.tar.gz
readme.txt
student_marks.txt
test.txt
localhost:~# ls
bench.py          file.tar.gz       hello.js          student_marks.txt
file.tar          hello.c           readme.txt        test.txt
localhost:~#
```

8.4.2 unzip Command

To compress a file, the zip command can be used as follows:

zip myfile.zip filename.txt

To unzip the file, the following command is used:

unzip myfile.zip

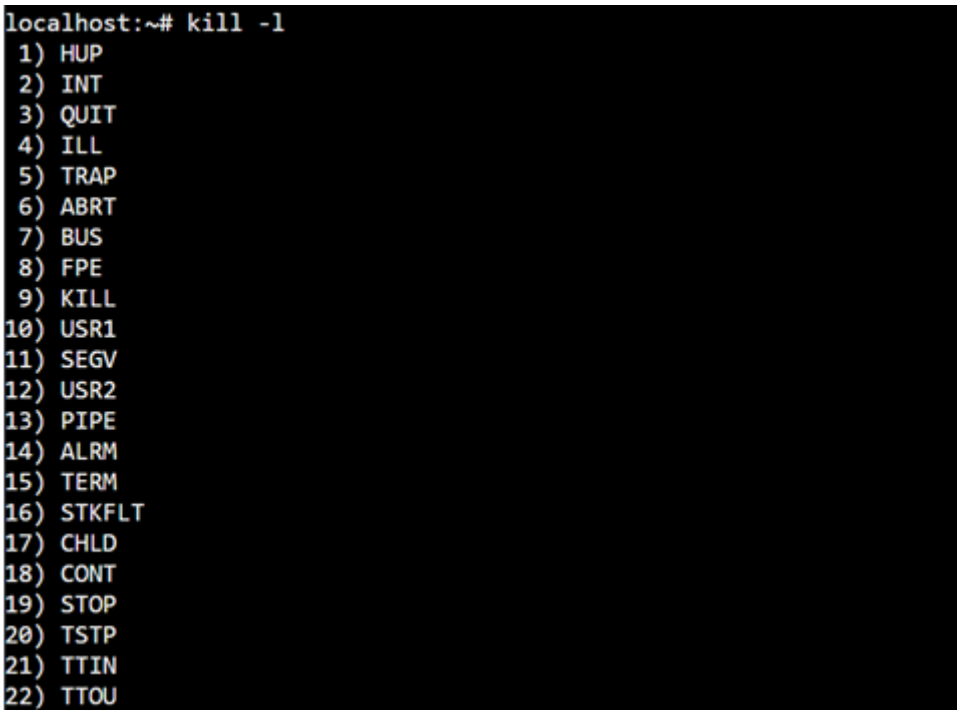
8.4.3 kill Command

It is used for manually terminating the processes. The command is:

type -a kill

The kill command transfers a signal towards a process in which further terminates the process. We can use the kill command along with the -l option for getting the list of every available signal:

kill -l



8.4.4 netstat Command

netstat command stands for Network statistics. It displays information about different interface statistics, including open sockets, routing tables, and connection information.

The basic syntax of the netstat command is as follows:

netstat

- Installation of the netstat command

If the netstat command is not installed on your machine, it will display the traditional Linux installation error message "Command 'netstat' not found."

To install it, execute the below command:

sudo apt install net-tools

The above command will ask for the administration password to install the command.

netstat -a

The above command will list all the existing connections.

```
localhost:~# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
netstat: /proc/net/tcp6: No such file or directory
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
netstat: /proc/net/udp6: No such file or directory
netstat: /proc/net/raw6: No such file or directory
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING       2930 /run/dhcpcd.sock
unix  2      [ ACC ]     STREAM     LISTENING       2932 /run/dhcpcd.unpriv.sock
unix  2      [ ]         DGRAM                      2961
localhost:~#
```

8.5 FILE ACCESS PERMISSION COMMANDS

In linux, three popular file access permission commands are used. They are (a) chmod (b) chown (c) chgrp. In the next sections, we will discuss how they can be used.

8.5.1 chmod Command

Linux chmod command is used to change the access permissions of files and directories. chmod stands for change mode. In the Linux, each file is associated with a particular owner and have permission access for different users. The user classes may be:

- owner
- group member
- Others (Everybody else)

The file permissions in Linux are the following three types:

- read (r)
- write (w)
- execute (x)

Let's see how to change the file permission using the chmod command.

The basic syntax of chmod command is as follows:

chmod <options> <permissions> <file name>

To view the permission of all text file from the current working directory, execute the below command:

```
localhost:~# ls -l *.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root            51 Oct  8 13:54 student_marks.txt
-rw-r--r--    1 root     root            51 Oct  8 13:09 test.txt
localhost:~#
```

To set the read and write permission for other users, execute the below command:

chmod o+w *.txt

It will set the read and write permission for other users of the text files. Consider the below output:

```
localhost:~# ls -l *.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root            51 Oct  8 13:54 student_marks.txt
-rw-r--r--    1 root     root            51 Oct  8 13:09 test.txt
localhost:~# chmod o+w *.txt
localhost:~# ls -l *.txt
-rw-r--rw-    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--rw-    1 root     root            51 Oct  8 13:54 student_marks.txt
-rw-r--rw-    1 root     root            51 Oct  8 13:09 test.txt
```

Numerical Shorthand

We can use the numeric values instead of letters to specify the permissions. A three-digit value is used to specify the permission. The leftmost digit represents the owner (u), and the middle digit represents the group members (g). The rightmost digit represents the others (o).

The following table represents the digits and their permissions:

Digits	**Permissions**
000	No permission
001	Execute permission
010	Write permission
011	Write and execute permissions
100	Read permission
101	Read and execute permissions
110	Read and write permissions
111	Read, write, and execute permissions

8.5.2 chown Command

The chown stands for change owner. chown command is used to change a file's ownership, directory, or symbolic link for a user or group.

- Display the UID, GID, and Groups

To display all the groups, execute the groups command as follows:

groups

To display the UID and GID, execute the id command as follows:

id

```
localhost:~# id
uid=0(root) gid=0(root)
localhost:~#
```

- Display the User and Group Ownership of a File

To display the user and group owner of a file, execute the "ls -l" command with a specific file name. Consider the below command:

ls -l hello.c

The above command will display the user and group ownership of the 'hello.c' Consider the below output:

```
localhost:~# ls -l hello.c
-rw-r--r--    1 root     root           76 Jul  3  2020 hello.c
localhost:~#
```

- Change the owner of a File (Using user name)

To change the owner of a file, pass the user name (new owner) with the chown command as follows:

sudo chown <username> <File name>

- Change the Owner of a File (Using UID)

We can change the owner of a file or directory using the UID of a user. To change the ownership by using the UID, pass the UID with chown command as follows:

sudo chown <userid> <File name>

- Change the Group of a File

To change the group of a file, specify group name or GID

with the chown command. Execute the chown command specifying the group name separated by a colon as follows:

sudo chown :<groupname> <filename>

or

sudo chown :<GID> <filename>

8.5.3 chgrp Command

chgrp command is used to change the group ownership of a file or directory.

- Change the Group Ownership of a File

To change the group ownership of a file, execute the command as follows:

sudo chgrp groupname filename

- Change the group ownership of a folder

To change the group ownership of a folder, execute the 'chgrp' command with the folder name as follows:

sudo chgrp groupname directoryname



8.6 LET US SUM UP

- The wc command displays the number of lines, number of characters, and the number of words in a file.
- The 'head' command displays the starting content of a file. By default, it displays starting 10 lines of any file.
- tail command is used to display the last ten lines of one or more files.
- Cut command is used to cut a specific section by byte position, character, and field and writes them to standard output.
- tr command is used to translate, like from lowercase to uppercase and vice versa.
- Top command is used to display all the running process within the environment of a system.
- grep command is used to search a keyword in a file.
- tar command is used for creating Archive and extracting the archive

files.

- kill command is used for manually terminating the processes.
- Netstat command displays information about different interface statistics, including open sockets, routing tables, and connection information.
- chmod command is used to change the access permissions of files and directories.
- chown command is used to change a file's ownership, directory, or symbolic link for a user or group.
- chgrp command is used to change the group ownership of a file or directory.



8.7 FURTHER READINGS

1) Das, S. (2000).UNIX: Concepts and Applications. Tata McGraw-Hill.
2) Das, S. (2012). Your UNIX/Linux: The Ultimate Guide. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
3) Christopher Negus (2012). Linux Bible. (8th Edition). Wiley.
4) https://www.javatpoint.com/linux-tutorial



8.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: wc <file1><file2>

Ans to Q No 2: wc - l filename

Ans to Q No 3: word

Ans to Q No 4: head <filename>

Ans to Q No 5: tr

Ans to Q No 6: grep



8.9 MODEL QUESTIONS

Q 1: Explain the Linux wc command with various options.

Q 2: Differentiate between head and tail command with examples.

Q 3: What are the functions of cut command?

Q 4: Explain the function of (a) tr and (b) top commands .

Q 5: Explain the grep command with examples.

Q 6: How will you compress and un-compress files using tar and untar commands? Explain with examples.

Q 7: How will you compress and un-compress files using zip and unzip commands? Explain with examples.

Q 8: For what purpose kill command is used?

Q 9: Explain the netstat command.

Q 10: Explain the chmod command with examples.

Q 11: Explain the chown command with examples.

Q 12: Explain the chgrp command with examples.

*** ***** ***

UNIT 9: LINUX SHELLS

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 Linux Shells

9.4 Shell Scripting

- 9.4.1 Shell Scripts
- 9.4.2 Shell Variables
- 9.4.3 Array in Shell Script
- 9.4.4 Operators used in Shell
- 9.4.5 Conditional and Loop Control Statements in Shell
- 9.4.6 Loop in Shell
- 9.4.7 Linux Shell Substitution
- 9.4.8 Input/Output Redirections in Shell
- 9.4.9 Functions in Shell

9.5 Let Us Sum Up

9.6 Further Readings

9.7 Answer to Check Your Progress

9.8 Model Questions

9.1 LEARNING OBJECTIVES

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

- understand about shell
- learn different shell in Linux
- understand shell script writing
- describe array in shell script
- learn to use conditional statement in shell script
- learn to use function in shell script

9.2 INTRODUCTION

A shell can be named as a special user program which provides an

interface to facilitate the users to use different operating system services. Shell accepts commands from user and converts them into kernel understandable format. This is why shell can also be termed as a command interpreter in Linux operating system. Shell is used to run other programs and commands. The shell enables the users with an interface to run the programs on Unix/GNU Linux environment. By using shell, the users can execute different tools/utilities or commands with a few input data. After executing the programs/codes, the shell sends the result to the common output device. This is why it is known as "command interpreter".

In addition to act as a command interpreter, the shell is also called a programming language with all the features of the programming language such as functions, variables, loops, conditional statements etc. Layered structure of Linux operating system is presented in figure 1 which includes Kernel, Shell, and Utilities along with Hardware in Innermost layer.

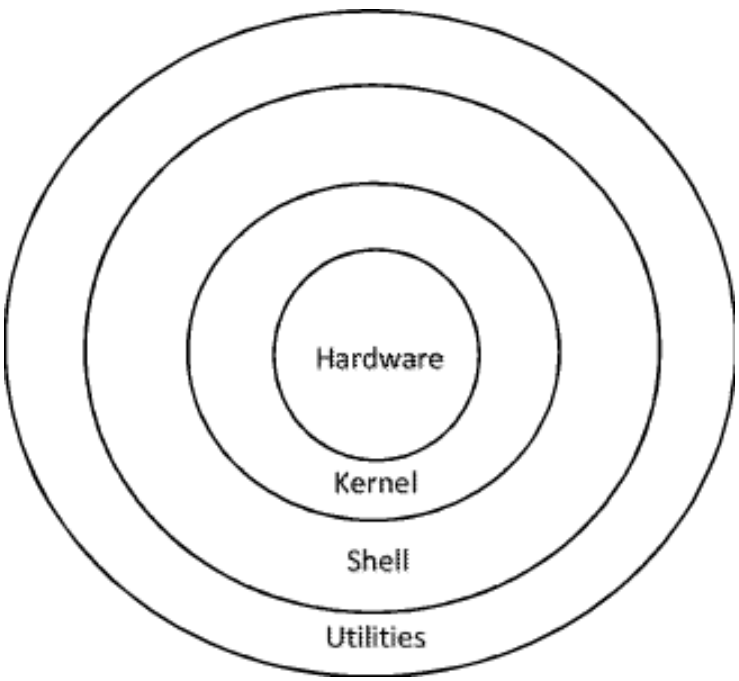


Figure 1. Different layers in Linux

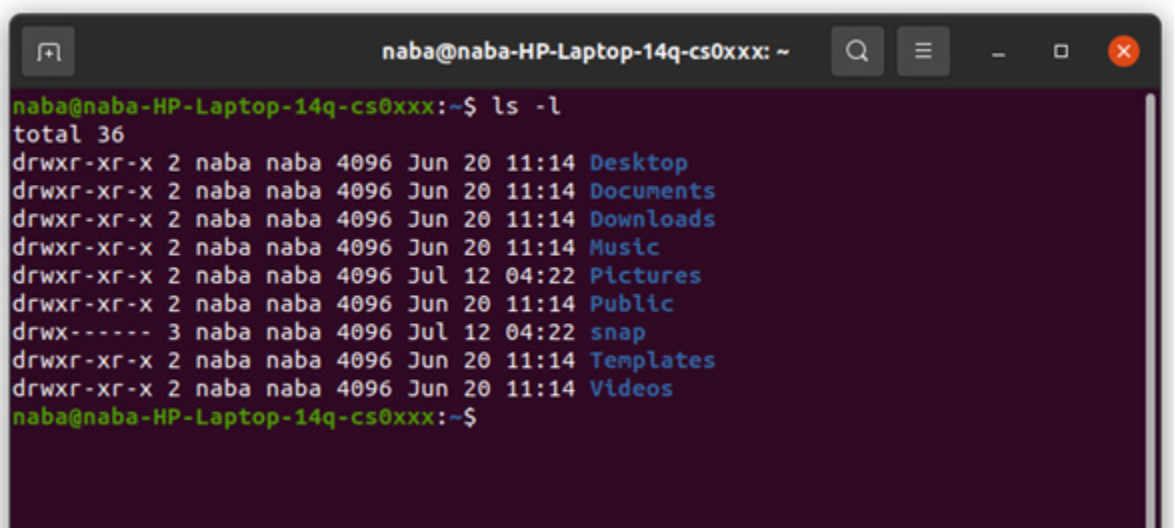
The shell can be categorized into two main classes depending on the user interface:

- Command-line Shell
- Graphical Shell

This categorization basically specifies the manipulation of programs, whether they are based on the graphical user interface (GUI) by permitting for operations like moving, closing, resizing, and opening windows etc. or command line where user has to input command for every task performed. Ubuntu OS or Windows OS could be examined as a good example that offers a graphical user interface to the user to interact with the program. Various users don't need for typing in any command for all the actions.

Command-line Shell

Shell can be accessed by user using a command line interface. Terminal in Linux / macOS and Command Prompt in Windows OS seems to be similar, providing an interface to the users to type Linux commands such as "cat", "ls" etc. which are to be executed by the kernel. The result is then displayed on the terminal for the user. An example of the Linux command "ls - l" in ubuntu version is given below.



"ls - l" list all the files in current working directory in long listing format.

Working in command line shell is difficult for the beginners because one has to memorize all the commands before using command line shell. Command line shell is very powerful, it also allows user to write commands in a file and execute them together. This way any repetitive task can be easily automated. These files are usually called Shell Scripts in Linux/mac OS.

Graphical Shells

Graphical shells provide users a graphical interface by which different operations such as opening, closing, moving and resizing windows etc. are performed. Window OS or Ubuntu OS can be considered as good example which provide GUI to user for interacting with program. User is not required to type in command for every actions. A typical GUI in Ubuntu system is shown in figure 2.



Figure 2. Typical GUI in Ubuntu system



EXERCISE

Q 1: Run different Linux commands in the shell and check the output.

9.3 LINUX SHELLS

Shells are one of the most powerful tools available for Linux user. Without shells, it is practically impossible for a user to utilise all the features and functionality offered by the kernel in a Linux system. Different types of Linux shells are discussed below. Each of these shells has some properties that make them highly efficient for a specific type of use over other shells and thereby it makes them different from each other.

- **The Bourne Shell (sh):** Bourne Shell is developed at AT&T Bell Labs by Steve Bourne. This shell is known as the first UNIX shell ever. It is

denoted as sh. It gained popularity due to its compact nature and high speeds of operation. Bourne Shell is used as the default shell for all Solaris system administration scripts.

Some major drawbacks in Bourne shell are observed. It doesn't have in-built functionality to handle logical and arithmetic operations. Also, unlike most different types of shells in Linux, it cannot recall previously used commands. It also does not have comprehensive features to offer a proper interactive use.

The complete path-name for the Bourne shell is /bin/sh and /sbin/sh. By default, it displays the command prompt # for the root user and $ for the non-root users.

- **The GNU Bourne-Again Shell (bash):** GNU Bourne-Again Shell is popularly known as the Bash shell. This bash shell was designed to be compatible with the Bourne shell. It includes useful features from different types of shells in Linux such as Korn shell and C shell. It allows us to automatically recall previously used commands and edit them with the help of arrow keys, unlike the Bourne shell. The complete path-name for the bash shell is /bin/bash. By default, it uses the prompt bash-VersionNumber# for the root user and bash-VersionNumber$ for the non-root users. Bash shell has following functionalities:
  - Job Control
  - Command-line editing
  - Shell Aliases and Functions
  - Unlimited size command history
  - Integer arithmetic in a base from 2-64
- **The C Shell (csh)** : The C shell was introduced by Bill Joy at the University of California. It is denoted as csh, also known as Tcsh. It was developed to include useful programming features like in-built support for arithmetic operations and the syntax similar to the C programming language. Further, it integrates command history which was missing in different types of shells in Linux like the Bourne shell. Another prominent feature of C shell is "aliases". The complete path-name for the C shell is /bin/csh. By default, it uses the prompt hostname# for the root user

and hostname% for the non-root users. Csh shell contains the following characteristics:

- C like syntax
- Filename completion and programmable word
- Command-line editor
- Job control
- Spelling correction

- **The Korn Shell (ksh):** The Korn shell was developed at AT&T Bell Labs by David Korn, to extend the features which are not available in the Bourne shell. It is denoted as ksh. The Korn shell is essentially a superset of the Bourne shell. Besides the functionalities of the Bourne shell, Korn shell supports additional functionalities to the user. It allows in-built support for arithmetic operations while offering interactive features which are similar to the C shell. The Korn shell runs scripts made for the Bourne shell, while offering string, array and function manipulation similar to the C programming language. It also supports scripts which were written for the C shell. Further, it is faster than other shells in Linux, including the C shell. The complete path-name for the Korn shell is /bin/ksh. By default, it uses the prompt # for the root user and $ for the non-root users.
- **The Z Shell (zsh):** The Z Shell or zsh is an extention of sh shell with various improvements and customization. If someone looking for a modern shell that has all the features and much more, the zsh shell can be considered. Some notable features of the z shell are:
  - It can generate filenames based on given conditions.
  - It has plugins and theme support.
  - It has index of built-in functions.
  - It has feature like auto completion of command.

A summary of different shells available in Linux is given below in a tabular format.

Shell	Complete path-name	Prompt for root user	Prompt for non-root user
Bourne shell (sh)	/bin/sh and /sbin/sh	#	$
GNU Bourne-Again shell (bash)	/bin/bash	bash-VersionNumber#	bash-VersionNumber$
C shell (csh)	/bin/csh	#	%
Korn shell (ksh)	/bin/ksh	#	$
Z Shell (zsh)	/bin/zsh	<hostname>#	<hostname>%



CHECK YOUR PROGRESS-1

Q 1: Which of the following is true?

a) Shell is a process and can be started by super user only

b) Shell is a built-in Kernel functionality

c) Shell is a wrapper for all the commands and utilities

d) None of the above

Q 2: BASH shell stands for?

Q 3: Shell is a Command Interpreter. (TRUE/ FALSE)

Q 4: Korn Shell is developed at AT & T Bell lab (TRUE/ FALSE)

9.4 SHELL SCRIPTING

Shell provides the interface to the user to work on Unix/Linux system. It acts as an environment in which we can run our commands, programs, and shell scripts.

Shell Prompt: The prompt, $, also called the command prompt, is provided by the shell to user. User can type commands in this prompt. After entering the command, shell reads the input, delivers it to the kernel for execution and finally displays the output.

For example, the date command displays the current date and time?

```
$ date
```

Thu Jun 25 08:30:19 MST 2009

We can customize the command prompt using the environment

variable.

9.4.1 Shell Scripts

Basically, a shell script is nothing but a program having a list of commands, which are listed in order. Following are some important points about shell scripts.

- Comments in the shell scripts are written, preceded by # sign.
- Shell functions and scripts are interpreted by the shell, not compiled.
- All the shell scripts have .sh as file extension.
- The shells are usually interactive where users can run their commands.
- The shells have conditional statements and looping statements, files to store and read data and variables etc.
- Syntax of shell script is similarly to other programming languages.

The shell script basically has following components:

- Functions
- Control flow: if, else, then, shell loops, case, etc.
- Shell commands: touch, pwd, echo, ls, cd, etc.
- Shell keywords: break, if, else, etc.

There are many reasons to write shell scripts -

- To avoid repetitive work and automation
- System admins use shell scripting for routine backups
- System monitoring
- Adding new functionality to the shell etc.

Advantages of shell scripts

- The command and syntax are exactly the same as those directly entered in command line.
- Writing shell scripts are much quicker.
- Interactive debugging etc.

Disadvantages of shell scripts

- Prone to errors; a single mistake can change the command which might be harmful.
- Slow execution speed.
- Design flaws within the language syntax or implementation.
- Not well suited for large and complex task.
- Provide minimal data structure unlike other scripting languages.

Shell Script Example 1:

Let us create a test.sh script file. First, we need to tell the system that we are going to use shell, this is done using the shebang construct. For example :

```
#!/bin/sh
```

This tells the system that the commands that follow are to be executed by the Bourne shell. This command is called a shebang because the # symbol is called a hash, and the ! symbol is called a bang.

Next, we need to add the commands in the script by using an editor like vi, vim, as follows -

```
#!/bin/bash
pwd
ls
```

In the above shell script #!/bin/sh is used to tell the system that we are going to use bash shell and pwd command prints the present working directory followed by ls command listing the files in the present working directory.

Shell Comments

We can put comments in the script as follows -

```
#!/bin/sh
#Author: Rahul Das
#Script follows here:
pwd
ls -l
```

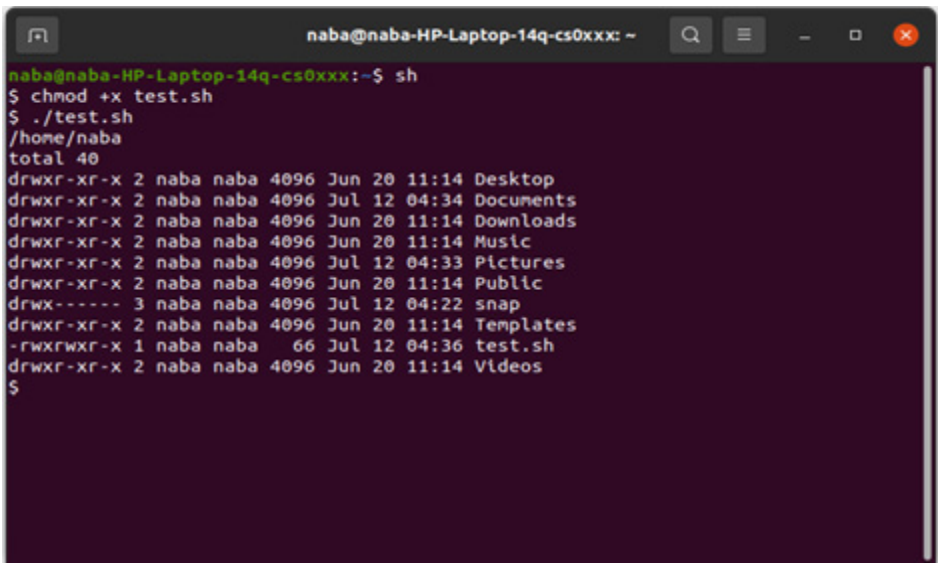
Then, we need to save the shell script and make the script executable by using chmod :

```
$ chmod +x test.sh
```

Finally, the shell script is executed as follows –

```
$ ./ test.sh
```

After execution, we will receive the following result -



Note ? To execute a program in the current directory, we have to run as ./ program_name

Shell Scripts Example 2:

Shell scripts have several required constructs that tell the shell environment what to do and when to do it. The shell script is a real programming language, having with variables, control structures, and so forth. Apart from command execution shell scripts are used for more task to perform. The following script uses the read command which takes the input from the keyboard and assigns it as the value of the variable and finally prints it on the standard output device.

```
#!/bin/sh
echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

output of the script –

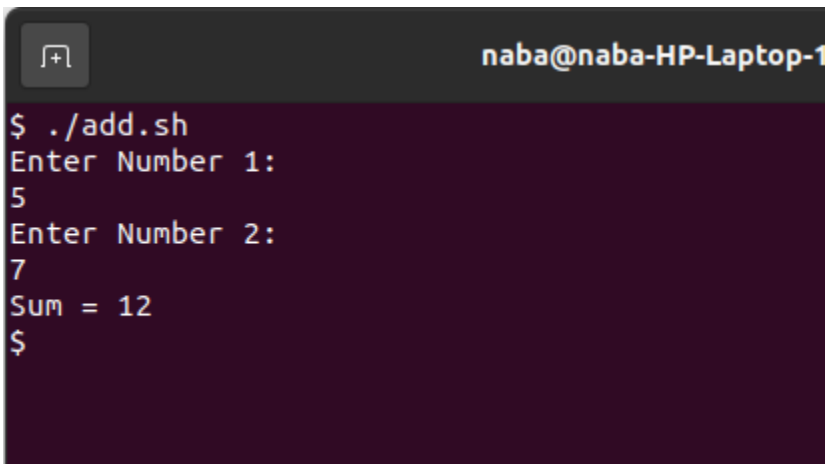


Shell Scripts Example 3:

The following script uses the read command to read two values from the keyboard and print the sum of two numbers on standard output devices.

```
#!/bin/sh
echo "Enter Number 1: "
read A
echo "Enter Number 2: "
read B
sum=$(expr $A + $B)
echo "Sum = $sum"
```

Output of the script is –





EXERCISE

Q 1: Write a shell script including different Linux commands and check the output.

9.4.2 Shell Variables

A variable is a character string where we can store value. The value assigned in the variable could be number, text, filename, device, or any other type of data.

A variable is nothing more than a pointer to the actual data. The shell enables us to create, store values to the variable, and delete variables.

Variable Names

Name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_). The variable names must start with letters.

The following examples are valid variable names :

TOKEN_A

VAR_1

VAR_2

Defining Variables

Variables are defined as follows :

variable_name = value

For example –

```
NAME="Rahul Roy"
```

The above example defines the variable NAME and assigns the value "Rahul Roy" to it. Variables of this type are called scalar variables. A scalar variable can hold only one value at a time.

Accessing Values

To access the value stored in a variable, dollar sign ($) is added as prefix to the variable name.

For example

```
#!/bin/sh
NAME="Rahul Roy"
echo $NAME
```

Output –

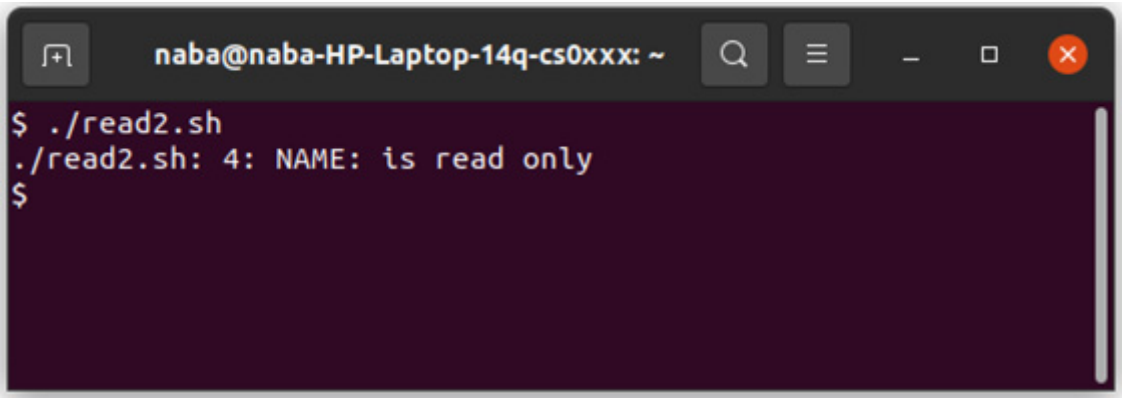
Rahul Roy

Read-only Variables

In shell script we can define a variable as read-only by using the read-only command as follows-

```
#!/bin/sh
NAME="Rahul Roy"
readonly NAME
NAME="Partha Das"
```

The above script will generate error result as bellow -



Unsetting or Deleting Variables

Unsetting or deleting a variable will remove the variable from the list of variables. Once the variable is deleted, we cannot access the stored value in the variable.

Following is the syntax to unset a variable –

unset variable_name

Note- We cannot use the unset command to unset variables that are marked readonly.

Variable Types

Variables may be of three types which are –

- Local Variables - A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at the command prompt.
- Environment Variables - An environment variable is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually, a shell script defines only those environment variables that are required for the programs during runtime.

- Shell Variables - A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

Special Variables

In shells some variables are reserved for specific functions, which are known as special or environment variables.

For example, the $ character represents the process ID number, or PID, of the current shell –

$echo $$

The above command tells the PID of the current shell -

29949

The table 1 lists the number of special variables that are used in shell scripts.

Table 1. Special variables with their purposes

Sl No	Variable	Description
1	$0	The filename of the current script.
2	$n	The variables correspond to the arguments with which a script was invoked.
3	$#	The number of arguments supplied to the script.
4	$*	All the arguments are double quoted. If a script receives two arguments, $* is equivalent to $1 $2.
5	$@	All the arguments are individually double quoted.
6	$?	Tells the exit status of the last command executed.
7	$$	The process number of the current shell. That is the process ID under which shell is executing.
8	$!	The process number of the last background command.

Command-Line Arguments

We can deliver command-line arguments when we run a shell script. Following example shows the use of command line arguments.

```
#!/bin/sh
Sum=$(expr $1 + $2 + $3)
```

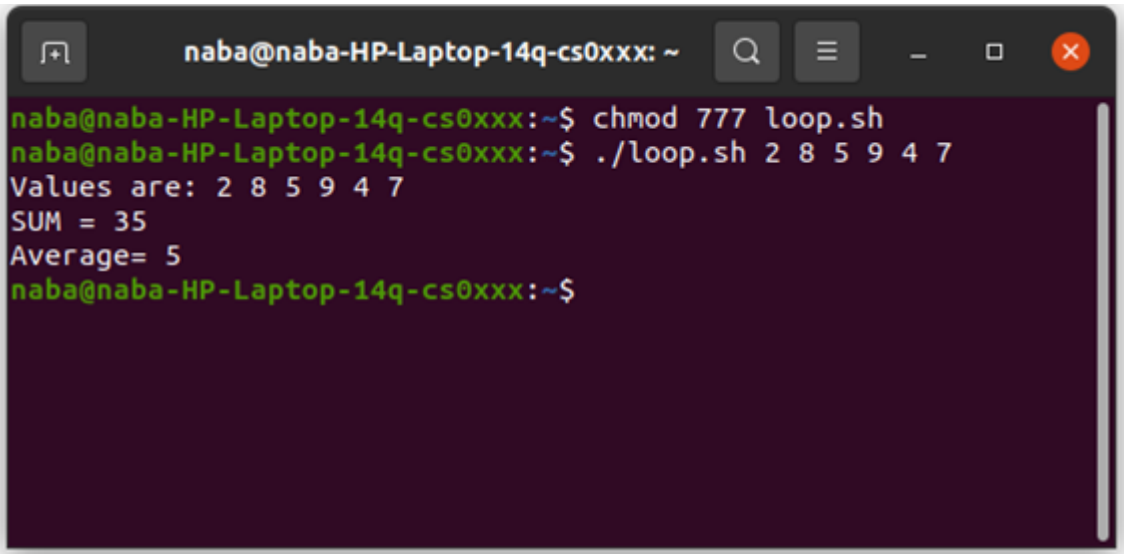
```
Avg=$(expr $Sum / $#)
echo "SUM = $Sum"
echo "Average= $Avg"
echo "Values are: $@"
echo "File Name: $0"
```

Output of the above script –



Following example is to find out the average of n numbers using $*

```
#!/bin/sh
SUM=0
for VALUE in $*
do
SUM=$(expr $SUM + $VALUE)
done
AVG=$(expr $SUM / $#)
echo "Values are: $@"
echo "SUM = $SUM"
echo "Average= $AVG"
Output of the above script -
```



Note - In the above example, for loop is used.

EXERCISE

Q 1: Write a shell script to input 3 numbers as the command line arguments and find the maximum value of these numbers.

9.4.3 Array in Shell Script

A shell variable which is capable of holding only single value are called scalar variables. Shell supports a different type of variable called an array, which can hold multiple values (may be of same type or different type) in consecutive memory locations at the same time. Arrays provide a method of grouping a set of variables.

Defining Array Values

The syntax of declaring an array variable is-

array_name=(value1 value2 …. valuen)

Here array_name is the name of the array, index is the index of the item. Index generally starts at 0 (zero) and go on. Example of array?

Arr=(10 20 30 40 50)

In ksh shell, the syntax of array initialization is –

set -A array_name value1 value2 ... valuen

In bash shell, the syntax of array initialization is –

array_name=(value1 ... valuen)

Accessing Array Values

After initialization of the array variable, we can access the array as follows –

${array_name[index]}

Here array_name is the name of the array, and index is the index of the value to be accessed.

We can access all the items in an array as follows ?

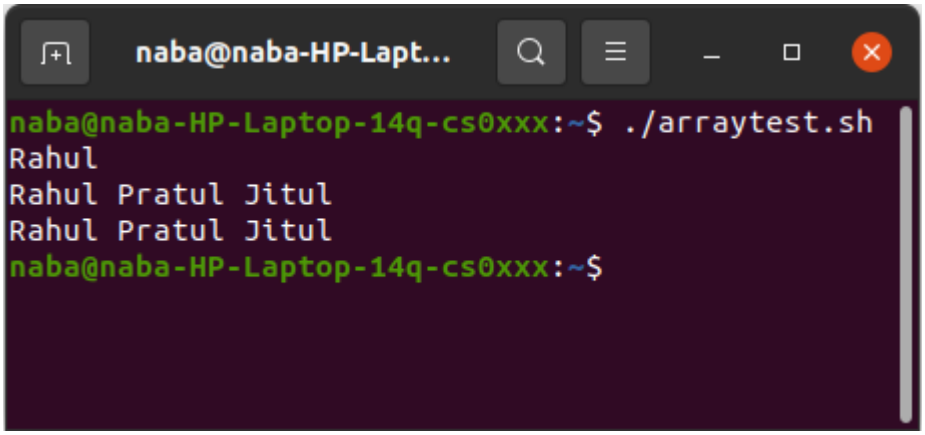
${array_name[*]}

${array_name[@]}

Example

```
#!/bin/sh
arr=(Rahul Pratul Jitul)
echo "${arr[0]}"
echo "${arr[*]}"
echo "${arr[@]}"
```

The above shell script once executed will generate the following result -





EXERCISE

Q 1: Write a shell script to find the average of 10 integers stored in an array.

Q 2: Write a shell script to find the maximum of n numbers in an array.

9.4.4 Operators used in Shell

There are various operators supported by shell. In this unit bash shell is used to discuss the different operators supported by shell. The operators in bash shell are broadly categorized as follows-

- Arithmetic Operators
- Relational Operators
- Boolean Operators
- String Operators
- File Test Operators

Arithmetic Operators

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator	`expr $a + $b`
- (Subtraction)	Subtracts right hand operand from left hand operand	`expr $a - $b`
* (Multiplication)	Multiplies values on either side of the operator	`expr $a \* $b`
/ (Division)	Divides left hand operand by right hand operand	`expr $b / $a`
% (Modulus)	Divides left hand operand by right hand operand and returns remainder	`expr $b % $a`
= (Assignment)	Assigns right operand in left operand	a = $b
== (Equality)	Compares two numbers, if both are same then returns true.	[$a == $b]
!= (Not Equality)	Compares two numbers, if both are different then returns true.	[$a != $b]

Note: All the conditional expressions should be inside square braces with spaces around them, for example [$a == $b] is correct whereas, [$a==$b] is incorrect. All the arithmetical calculations are

done using long integers.

Relational Operators

Operator	Description	Example
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a -eq $b]
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[$a -ne $b]
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[$a -gt $b]
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[$a -lt $b]
-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -ge $b]
-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -le $b]

Boolean Operators

Operator	Description	Example
!	This is logical negation. This inverts a true condition into false and vice versa.	[! false] is true.
-o	This is logical **OR**. If one of the operands is true, then the condition becomes true.	[$a -lt 20 -o $b -gt 100] is true if a=45 and b=120

-a	This is logical **AND**. If both the operands are true, then the condition becomes true otherwise false.	[$a -lt 20 -a $b -gt 100] is false if a=45 and b=120

String Operators

Operator	Description	Example
=	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a = $b]
!=	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.	[$a != $b]
-z	Checks if the given string operand size is zero; if it is zero length, then it returns true.	[-z $a]
-n	Checks if the given string operand size is non-zero; if it is nonzero length, then it returns true.	[-n $a]
Str	Checks if **str** is not the empty string; if it is empty, then it returns false.	[$a]

File Test Operators

Operator	Description	Example
-b file	Checks if file is a block special file; if yes, then the condition becomes true.	[-b $file]
-c file	Checks if file is a character special file; if yes, then the condition becomes true.	[-c $file]
-d file	Checks if file is a directory; if yes, then the condition becomes true.	[-d $file]
-f file	Checks if file is an ordinary file as opposed to a directory or special file; if yes, then the condition becomes true.	[-f $file]
-g file	Checks if file has its set group ID (SGID) bit set; if yes, then the condition becomes true.	[-g $file]
-k file	Checks if file has its sticky bit set; if yes, then the condition becomes true.	[-k $file]

-p file	Checks if file is a named pipe; if yes, then the condition becomes true.	[-p $file]
-t file	Checks if file descriptor is open and associated with a terminal; if yes, then the condition becomes true.	[-t $file]
-u file	Checks if file has its Set User ID (SUID) bit set; if yes, then the condition becomes true.	[-u $file]
-r file	Checks if file is readable; if yes, then the condition becomes true.	[-r $file]
-w file	Checks if file is writable; if yes, then the condition becomes true.	[-w $file]
-x file	Checks if file is executable; if yes, then the condition becomes true.	[-x $file]
-s file	Checks if file has size greater than 0; if yes, then condition becomes true.	[-s $file]
-e file	Checks if file exists; is true even if file is a directory but exists.	[-e $file]

Note:- Apart from the above operators, different shell supports different set of operators. Some shells like C Shell and K Shell have few more additional operators and few operators are different from bash shell operators.



EXERCISE

Q 1: List different operators used in shell programming. Find the difference between C programming operators and shell programming operators.

9.4.5 Conditional and loop control Statements in Shell

Unix Shell supports conditional statements which are used to perform different tasks based on the condition values. There are mainly two different conditional statements supported by shell ?

- if...else statement
- case...esac statement

if...else statements

If else statements are useful in decision-making where condition like either or is present.

Linux Shell supports following types of if…else statement –

- if...fi statement
- if...else...fi statement
- if...elif...else...fi statement

The if statements check the condition by using relational operators and depending on the value of the conditional statement, the if or else block is executed.

- if...fi statement

Syntax of if…fi statement

if [expression]
then
Statement(s) to be executed if expression is true
fi

- if...else…fi statement

Syntax of if…else…fi statement

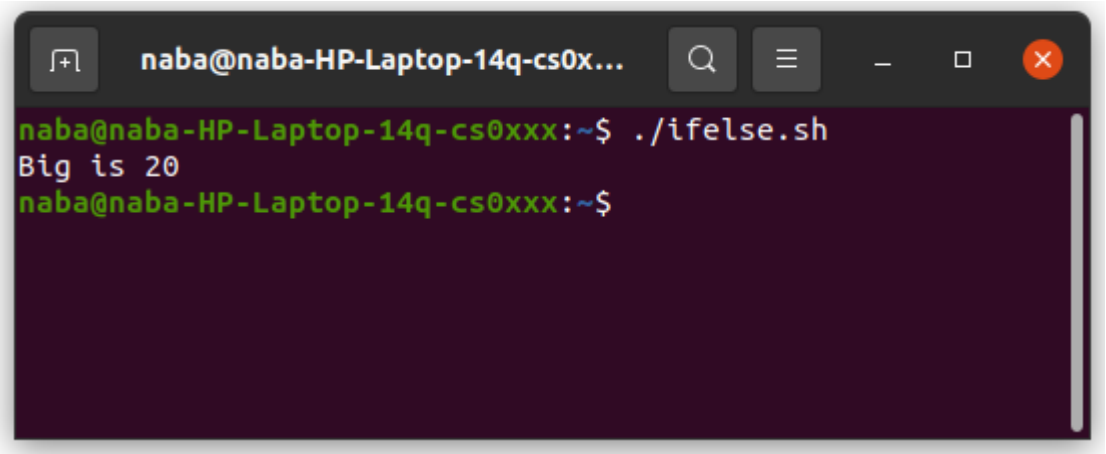
if [expression]
then
Statement(s) to be executed if expression is true
else
Statement(s) to be executed if the expression is false
fi

Example

```
#!/bin/sh
a=10
b=20
if [ $a -gt $b ]
then
   echo "Big is $a"
else
      echo "Big is $b"
```

```
fi
```

OUTPUT



- if...elif…else…fi statement

Syntax of if…else…fi statement

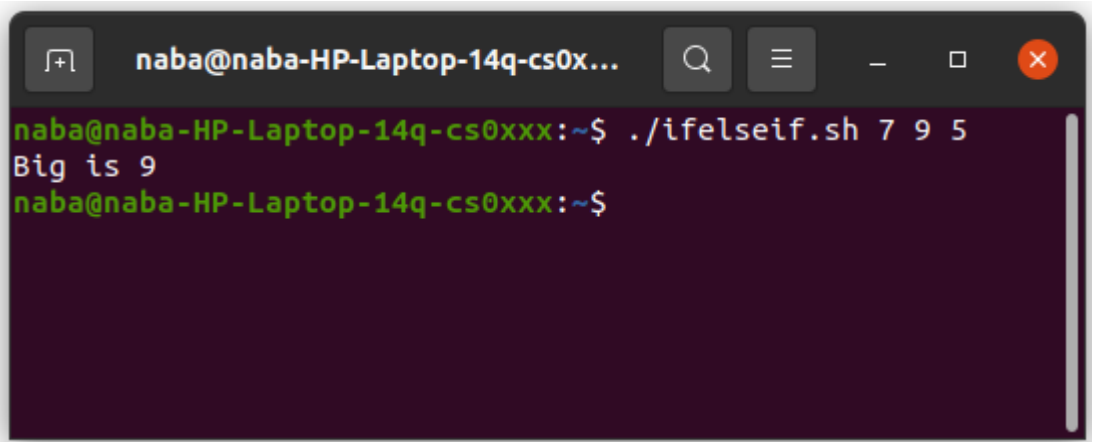
```
if [ expression 1 ]
then
Statement(s) to be executed if expression 1 is
   true
elif [ expression 2 ]
then
   Statement(s) to be executed if expression 2 is
   true
   elif [ expression 3 ]
   then
      Statement(s) to be executed if expression 3
   is true
   else
      Statement(s) to be executed if no expression
   is true
fi
```

Example

```
#!/bin/sh
a=$1
b=$2
```

```
c=$3
if [ $a -gt $b ]
then
if [ $a -gt $c ]
then
   echo "Big is $a"
fi
elif [ $b -gt $c ]
then
   echo "Big is $b"
else
   echo "Big is $c"
fi
```

OUTPUT



The case...esac Statement

We can use multiple if...elif statements to perform a multiway branch. However, it is better to use case...esac statement which handles exactly the same situation, and it does so more efficiently than repeated if...elif statements.

The case...esac statement in the Linux shell is very similar to the switch...case statement in C/C++ programming language.



CHECK YOUR PROGRESS-2

Q 5: The shell script is a file containing a series of commands. (TRUE/ FALSE)

Q 6: To read data from user, which command is used in Linux?

Q 7: The command line arguments $0 represents_____.

Q 8: C=A+B is written in shell script as_____.

Q 9: A>B is written in shell script as_____.

Q 10: All the elements of an array can be displayed by _____.

Q 11: != is a legal operator in shell script. (TRUE / FALSE)

Q 12: >= is a legal operator in shell script. (TRUE / FALSE)

9.4.6 Loop in Shell

A loop is a powerful programming construct which enables us to execute a set of commands repeatedly. Following types of loops available in shell –

- The while loop
- The for loop
- The until loop
- The select loop
- The while loop

Syntax

while command

do

Statement(s) to be executed if command is true

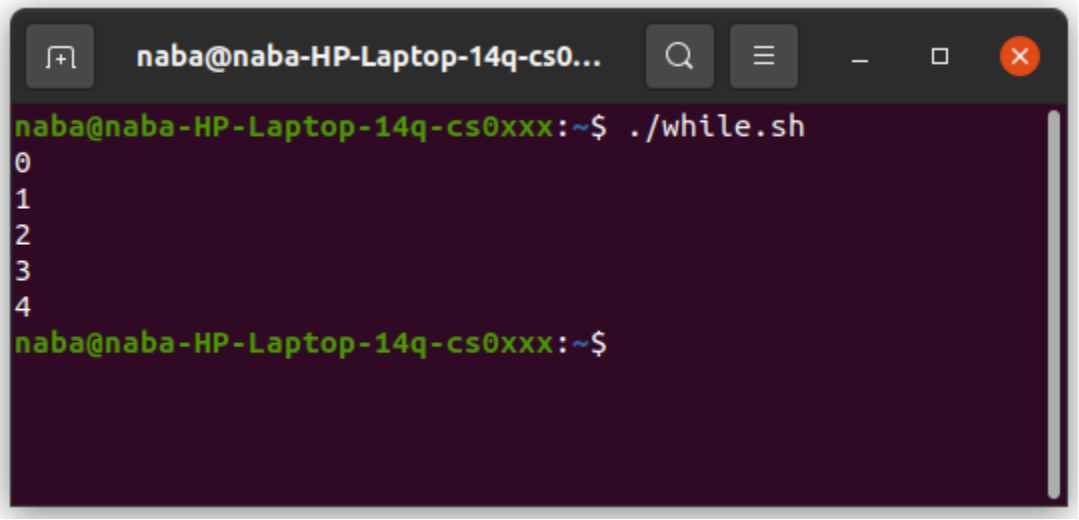
done

Example

```
#!/bin/sh
a=0
while [ $a -lt 5 ]
do
   echo $a
```

```
    a=`expr $a + 1`
done
```

OUTPUT



- The for loop

Syntax

for var in word1 word2 ... wordN

do

Statement(s) to be executed for every word.

done

Example

```
#!/bin/sh
for var in 7 8 9 6 3
do
    echo $var
done
```

OUTPUT



- The until loop

Syntax

until condition

do

Statement(s) to be executed until command is true

done

- The select loop

Syntax

select var in word1 word2 ... wordN

do

Statement(s) to be executed for every word.

done

Nesting Loops

All the loops support nesting concept which means that we can put one loop inside another similar one or different loops. This nesting can go up to unlimited number of times based on our requirement.

Example

Here is a simple example of nested loop

```
#!/bin/sh
a=0
while [ $a -lt 10 ]    # this is loop1
do
b=$a
```

```
        while [ $b -ge 0 ]  # this is loop2
           do
              echo -n $b
              b=`expr $b - 1`
  done
    echo
    a=`expr $a + 1`
done
```

Output of the above shell script is as follows.



Note: In the above shell script, -n option lets echo avoid printing a new line character.

Loop Control statements

There are two loop control statements supported by shell ?

- break statement
- continue statement

The infinite Loop

A loop continues forever if the required condition is not met. A loop that executes forever without terminating infinite number of times is called an infinite loop.

The break Statement

The break statement is used to terminate the execution of loop

statements. After execution of break statement, the control came out of the loop and executes the next statement after loop.

Syntax

break

break n

Here n specifies the nth enclosing loop to break from.

The continue statement

The continue statement is similar to the break, except that it causes the current iteration of the loop to exit, rather than the entire loop and again start from beginning of the loop.

Syntax

continue

continue n

Here n specifies the nth enclosing loop to continue from.



EXERCISE

Q 1: List different loops used in shell programming. Find the difference between loops used in C programming with loops used in shell programming.

9.4.7 Linux Shell Substitution

The following escape sequences are used with echo command on

Sl. No.	Escape sequence	Description
1	\\	backslash
2	\a	alert (BEL)
3	\b	Backspace
4	\c	suppress trailing newline
5	\f	form feed
6	\n	new line
7	\r	carriage return
8	\t	horizontal tab
9	\v	vertical tab

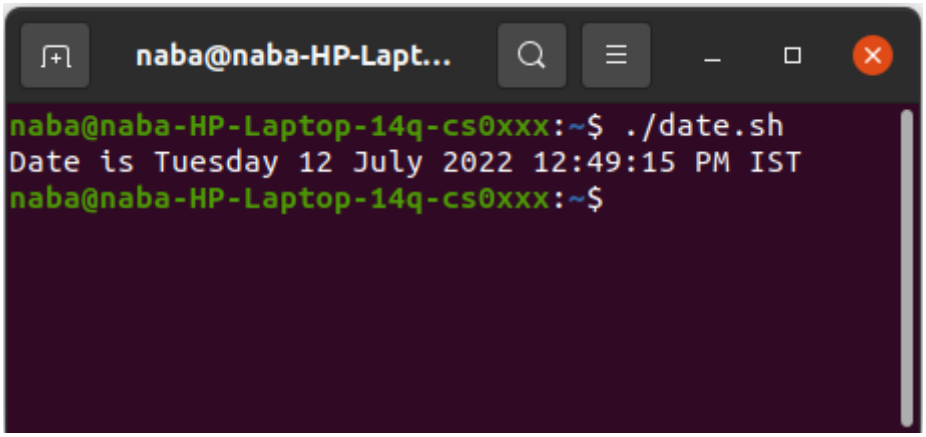
Command Substitution

By command substitution shell performs a given set of commands and then substitutes by the output in place of commands.

Example

```
#!/bin/sh
DATE=`date`
echo "Date is $DATE"
```

Output of the above shell is



Variable Substitution

Variable substitution enables the shell programmer to manipulate the value of a variable based on its state.

Sl. No.	Command	Form & Description
1	**${var}**	Substitute the value of *var*.
2	**${var:-word}**	If *var* is null or unset, *word* is substituted for **var**. The value of *var* does not change.
3	**${var:=word}**	If *var* is null or unset, *var* is set to the value of **word**.
4	**${var:?message}**	If *var* is null or unset, *message* is printed to standard error. This checks that variables are set correctly.
5	**${var:+word}**	If *var* is set, *word* is substituted for var. The value of *var* does not change.

Metacharacters in Shell

Unix Shell provides various metacharacters which have special meaning in the Shell Script and causes termination of a

word unless quoted.

For example, ? matches with a single character while listing files in a directory and * matches more than one character. List of popular metacharacters in shell is given bellow?

```
* ? [ ] ' " \ $ ; & ( ) | ^ < > new-line space tab
```

Sl. No.	Quoting & Description
1	**Single quote** All special characters between these quotes lose their special meaning.
2	**Double quote** Most special characters between these quotes lose their special meaning with these exceptions – • $ • ` • \$ • \' • \" • \\
3	**Backslash** Any character immediately following the backslash loses its special meaning.
4	**Back quote** Anything in between back quotes would be treated as a command and would be executed.

9.4.8 Input/Output Redirections in Shell

Most Unix system commands take input from the terminal and show the result in the terminal. A command normally reads the input from the standard input devices and displays the output in the standard output devices. But in some situations, input/output is not performed from standard I/O devices instead they are redirected from other place like file etc. This is called input/output redirection

Output Redirection

If the output of a shell script or a command is diverted from standard output device (STDOUT) to some other location then it is called as

output redirection.

The notation '>' is used for output redirection, for example

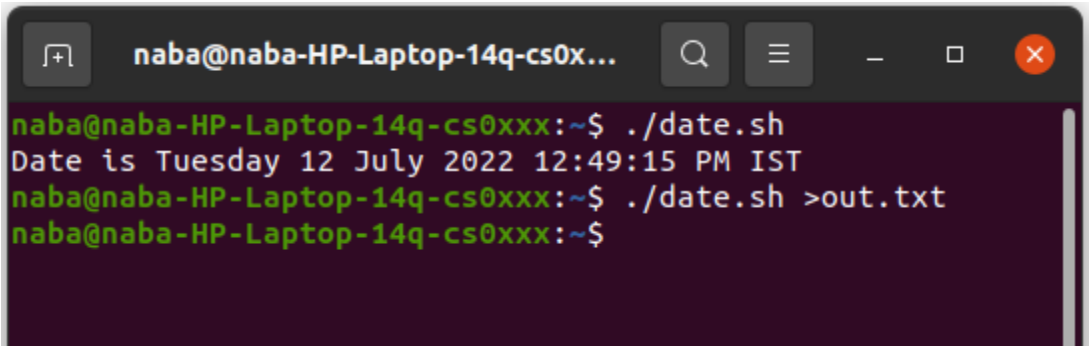
$ who > users

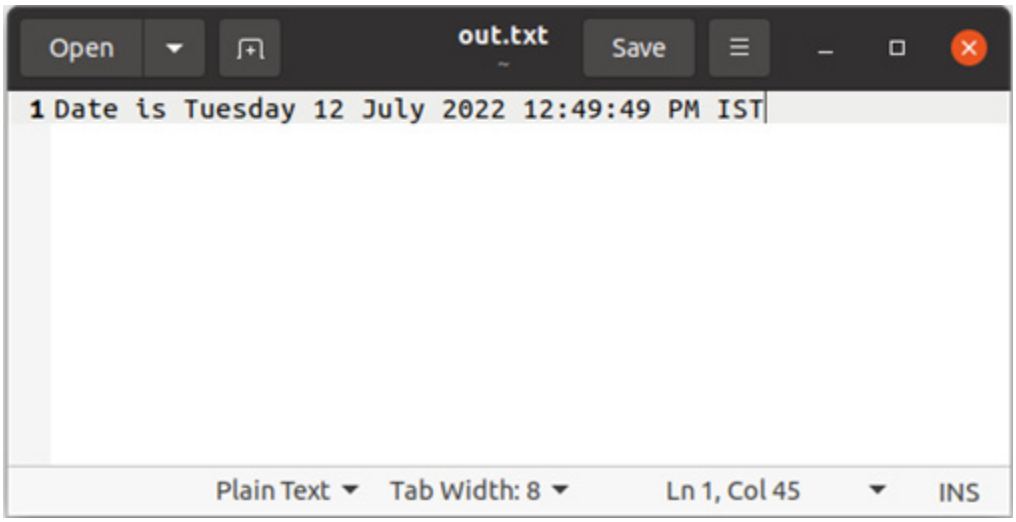
Runs the who command and write the output of the command in the file 'users'.

Input Redirection

Similar to output redirection, input to a shell script or a command can also be redirected from some other location instead of standard input device (STDIN). less-than character < is used for input redirection.

For example :





Here output of the previous program date is redirected to out.txt file.

Here Document

Here document is used to redirect input into an interactive

shell script or program.

We can run an interactive program within a shell script without user action by supplying the required input for the interactive program, or interactive shell script. This is done by using '<<' symbol.

Discard the output

Sometimes we need to execute command or shell script, but we don't want the output displayed on the monitor. In such cases, we can discard the output by redirecting it to the file /dev/null –

$ command > /dev/null

Redirection Commands

Following is a complete list of commands which you can use for redirection –

Sl. No.	Command	Description
1	**pgm > file**	Output of pgm is redirected to file
2	**pgm < file**	Program pgm reads its input from file
3	**pgm >> file**	Output of pgm is appended to file
4	**n > file**	Output from stream with descriptor **n** redirected to file
5	**n >> file**	Output from stream with descriptor **n** appended to file
6	**n >& m**	Merges output from stream **n** with stream **m**
7	**n <& m**	Merges input from stream **n** with stream **m**
8	**<< tag**	Standard input comes from here through next tag at the start of line
9	**|**	Takes output from one program, or process, and sends it to another

Note: The file descriptor 0 is used for standard input (STDIN), 1 for standard output (STDOUT) and 2 for standard error output (STDERR).

9.4.9 Functions in shell

Functions enable us to break down the shell script into smaller, logical subsections, which can perform some tasks individually.

Using functions, we can perform repetitive tasks very easily. Shell functions are similar to subroutines, procedures, and functions in other programming languages.

Function Creation

Syntax for function declaration in shell is

```
function_name ()
  {
      list of commands
  }
```

Example:

```
#!/bin/sh
# Function Definition
Hello () {
   echo "Hello World"
}
# Function call
Hello
```

Output of the shell script -



Parameter in Function

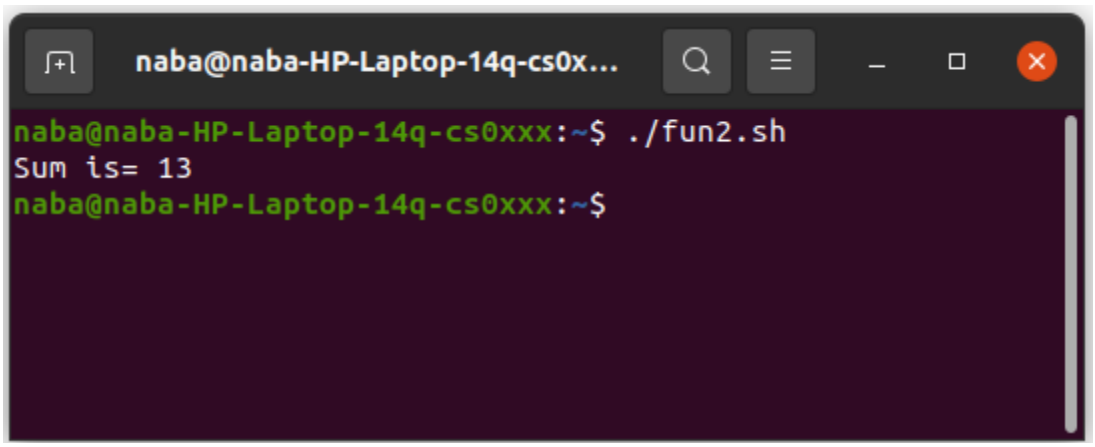
We can define function which accept parameters while calling the function. These parameters would be represented by the variable $1, $2 and so on.

Example:

```
#!/bin/sh
```

```
# Function Definition
Add () {
   SUM=$(expr $1 + $2)
   echo "Sum is= $SUM "
}
# Function Call
Add 5 8
```

Output of the script



Returning Values from Function

Function can return a value by using 'return' command.

Syntax of 'return' ?

return code

Here code can be anything that we want to return.

Example

Following function returns a value ?

```
#!/bin/sh
# Function Definition
Add () {
   SUM=$(expr $1 + $2)
   return $SUM
}
# Function Call
Add 5 8
# Return value of the function
```

```
ret=$?
echo "Sum is = $ret"
```

Output of the shell script -



Nested Functions

A function can call another function. This feature of function calling is called nested function. A function that calls itself is known as a recursive function.



CHECK YOUR PROGRESS-3

Q 13: In shell script a function can return a value. (TRUE/ FALSE)

Q 14: break and continue statement are supported in shell programming. (TRUE/ FALSE)

Q 15: We cannot redirect output from shell script. (TRUE/ FALSE)

Q 16: Syntax of for loop is _____



9.5 LET US SUM UP

- A shell is a special user program which provide an interface to user to use operating system services.
- Shell accepts commands from user and converts them into kernel understandable version.
- Shell is also known as "command interpreter".
- Broadly, the shell is categorized into two main classes- Command-line Shell, Graphical Shell

- There are five different shells in Linux- The Bourne Shell (sh), The GNU Bourne-Again Shell (bash), The C Shell (csh), The Korn Shell (ksh), The Z Shell (zsh).
- Bourne shell is the first UNIX shell. It is compact and high speeds.
- GNU Bourne-Again Shell is popularly known as the Bash shell, the GNU Bourne-Again shell was designed to be compatible with the Bourne shell. It incorporates useful features from different types of shells in Linux such as Korn shell and C shell.
- The C shell support for arithmetic operations and a syntax similar to the C programming language.
- The Korn shell is an improvement over the Bourne shell.
- The Z Shell is a sh shell extension with various improvements and customization.
- Shells are one of the most powerful tools available for Linux user. Without shells, it is practically impossible for a user to utilise all the features and functionality offered by the kernel in a Linux system.
- A shell script is a list of commands, which are listed in order they need to be executed.
- Shell scripts have several required constructs that tell the shell environment what to do and when to do it. The shell script is a real programming language, complete with variables, control structures, and so forth.
- In shell script, a variable is nothing more than a pointer to the actual data. The shell enables us to create, store values to the variable, and delete variables.
- There are three main types of variables- Local Variables, Environment Variables, Shell Variables.
- Shell Programming supports Command-Line Arguments. Via command line arguments we can provide input to the shell script.
- Shell supports array, which can hold multiple values at the same time. Arrays provide a method of grouping a set of variables.
- Shell programming supports the different types of operators as supported by any high-level languages.

- Shell programming also supports conditional statements- if, if-else loop statements while, for, etc .
- Shell script support Input/Output Redirections, via which user can redirect the output to some other file or can provide input from file.
- Like any other high level programming languages shell scripts also support functions.



9.6 FURTHER READINGS

1) Shell Scripting by Jason Cannon
2) Linux Command Line and Shell Scripting Bible by Richard Blum & Christine Bresnahan.
3) Mastering Linux Shell Scripting by Andrew Mallett & Mokhtar Ebrahim



9.7 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: c) Shell is a wrapper for all the commands and utilities

Ans to Q No 2: Bourne-again Shell

Ans to Q No 3: TRUE

Ans to Q No 4: TRUE

Ans to Q No 5: TRUE

Ans to Q No 6: Read

Ans to Q No 7: Name of the file

Ans to Q No 8: C=$(expr $A + $B)

Ans to Q No 9: $A -gt $B

Ans to Q No 10: ${arr[*]} or ${arr[@]}

Ans to Q No 11: TRUE

Ans to Q No 12: FALSE

Ans to Q No 13: TRUE

Ans to Q No 14: TRUE

Ans to Q No 15: FALSE

Ans to Q No 16: for var in word1 word2 ... wordN

```
do
Statement(s) to be executed for every word.
```

```
done
```



9.8 MODEL QUESTIONS

Q 1: What is shell? What are the different shells available in Linux Operating System? Explain briefly.

Q 2: What is shell scripting? What are the advantages of it?

Q 3: What are the different operators supported by shell? Explain briefly.

Q 4: Write a shell script to find the average of n numbers.

Q 5: Write a shell script to find the minimum of n numbers.

Q 6: Write a shell script to print the Fibonacci sequence up to n.

Q 7: Write a shell script to find the average of array elements.

*** ***** ***

UNIT 10: PROCESS

UNIT STRUCTURE

10.1 Learning Objectives

10.2 Introduction

- 10.2.1 Types of Process
- 10.2.2 Starting a Process
- 10.2.3 Listing Running Processes
- 10.2.4 Stopping a Process
- 10.2.5 Attributes of a Process
- 10.2.6 Job, Task and Process

10.3 Job Control

10.4 Crontab File Format

10.5 Linux Backup and Recovery

10.6 Let Us Sum Up

10.7 Further Readings

10.8 Answer to Check Your Progress

10.9 Model Questions

10.1 LEARNING OBJECTIVES

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

- understand about services and processes in Linux
- learn the basic commands for process management
- explain the attributes of process status word
- describe job, task and processes
- understand crontab file format and its use
- understand to backup Linux system and restore from backup.

10.2 INTRODUCTION

In the previous unit, we have learnt the concept of shell script. We have discussed the use of conditional and loop statements in a shell script. We have also discussed how functions can be used in a shell script. In this

unit, we will discuss the various tasks associated with a process like starting, listing, and stopping a process.

Process: A program in running state (execution) is called a process. When a program is executed on Unix/ Linux operating system, the operating system creates an environment for the program. This environment contains all the resources required by the program to execute in the system if no other processes were running on the system.

When we run a command in Unix/ Linux operating system, it creates or starts a new process. For example, when we execute the ls command, a process is started to list all the directory and files in the current directory and display it. A process, in simple terms, is an instance of a running program.

When a process starts executing, the operating system assigns a process ID (pid) to the process. This process ID is unique for all the processes running in the system but after termination of the process, same process id may be assigned to some other process. Linux uses PID to track each process in the system.

Service: Linux systems provide variety of system services (process management, login, syslog, cron, etc.), network services (remote login, e-mail, printers, web hosting, data storage, file transfer, domain name resolution, dynamic IP address assignment) and much more.

A service is a program or an application in Linux that runs in the background. It runs without the need of user intervention and sometimes users are unaware of it. Most of the services stated by the operating system for smooth running of the system. Generally, a Linux service has the following characteristics:

- A service does not have a graphical interface. So, users cannot interact with services through graphical interface.
- Most of the services are started with the operating system. Third-party services can be configured to start or not with the system by the user. Even few third-party services can be started by the user as per requirement.

- Services running in the background, wait for certain signal to start the processing. For example, mail demon checks for new email and displays it when any new mail is received.

Linux supports different ways to manage services such as start, stop, restart, enable auto-start at system boot, etc.

10.2.1 Types of Process

Process in Linux can be of two types-

- Foreground Processes
- Background Processes

Foreground Processes: By default, every process that we start runs in the foreground. A foreground process reads input from standard input device and displays the output in the standard output device. For example, if we run ls command, it will list all the files in the current directory on the terminal. Most of the foreground processes can be run using graphical user interface, for example media player, word processing software etc.

When a process runs in the foreground, no other process can run (or start any other processes) even if the process is time-consuming. The terminal (or command prompt) become inaccessible when a process runs in the foreground until the process finishes its execution.

Background Processes: A background process runs in the background without being directly connected to the input/output devices. When a background process requires any input or need to display an output, it waits for the system to make them available. Most of the background process are stated by the system for smooth functioning of the system. The advantage of running a background process is that we can run other processes in the system at the same time; we need not have to wait for completion of the background process to start another process.

If a process running in the background needs any input, it

goes into waiting state until some other process moves it into the foreground and provides the data from standard input device.

10.2.2 Starting a process

A process can be run either in foreground or in background.

Foreground Process: Every process when started runs in foreground by default, receives input from the keyboard, and sends output to the screen. Foreground processes can be run from graphical user interface by typing command in the terminal. For example

$ pwd

will give the current working directory of the user like

/home/student

Background Process: Background process can be started by adding '&' along with the command.

$ pwd &

will give the following output

[1] 2567

/home/student

Since pwd does not require any input from the keyboard, it goes to the stop state until moved to the foreground.

10.2.3 Listing Running Processes

We can list the running processes started by current user by using the ps (process status) command as follows ?

$ps

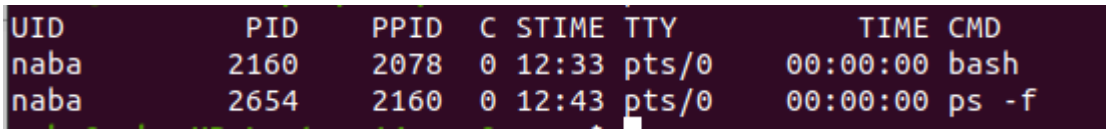
OUTPUT:

```
  PID TTY          TIME CMD
 2160 pts/0    00:00:00 bash
 2583 pts/0    00:00:00 ps
```

One of the most commonly used flags for ps is the -f (f for full) option, this provides more information about the currently running processes ?

$ps -f

OUTPUT:



Description of the fields displayed by ps -f command is given below:

Sl. No.	Column	Description
1	**UID**	User ID that this process belongs to (the person runn
2	**PID**	Process ID
3	**PPID**	Parent process ID (the ID of the process that started
4	**C**	CPU utilization of process
5	**STIME**	Process start time
6	**TTY**	Terminal type associated with the process
7	**TIME**	CPU time taken by the process
8	**CMD**	The command that started this process

Other common options used with ps command are -

Sl. No.	Option	Description
1	-a	Displays all processes on a terminal, with the exception leaders.
2	-c	Displays scheduler data.
3	-d	Displays all processes with the exception of session lead
4	-e	Displays all processes.
5	-f	Displays a full listing.
6	-g *list*	Displays data for the *list* of group leader IDs.
7	-j	Displays the process group ID and session ID.
8	-l	Displays a long listing
9	-p *list*	Displays data for the *list* of process IDs.
10	-s *list*	Displays data for the *list* of session leader IDs.
11	-t *list*	Displays data for the *list* of terminals.
12	-u *list*	Displays data for the *list* of usernames.

10.2.4 Stopping a Process

Process can be stopped by several different ways. If the process is running on console, we can stop that process by pressing CTRL + C. In Linux, CTRL + C is used to send a SIGINT signal, which cancels or terminates the currently-running program. This works when the process is running in the foreground mode.

If we want to run a process which is not currently running on the console or the process is running in the background, then by

using ps command we can get the process ID. The kill command is used to kill the process by using process ID. Following example shows the use of kill command ?

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2160    2078  0 12:33 pts/0    00:00:00 bash
naba        2988    2160 10 12:52 pts/0    00:00:00 gedit
naba        2995    2160  0 12:52 pts/0    00:00:00 ps -f
naba@naba-HP-Laptop-14q-cs0xxx:~$ kill 2988
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2160    2078  0 12:33 pts/0    00:00:00 bash
naba        3001    2160  0 12:52 pts/0    00:00:00 ps -f
[1]+  Terminated              gedit
```

Here, the kill command terminates the process gedit (PID 2298). If a process cannot be killed by kill command, then kill -9 command is used to kill the process by force. Following example shows the use of kill -9 command :

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2855    2847  2 12:48 pts/2    00:00:00 gedit
naba        2864    2847  0 12:49 pts/2    00:00:00 ps -f
naba@naba-HP-Laptop-14q-cs0xxx:~$ kill 2855
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2855    2847  1 12:48 pts/2    00:00:00 gedit
naba        2932    2847  0 12:49 pts/2    00:00:00 ps -f
naba@naba-HP-Laptop-14q-cs0xxx:~$ kill -9 2855
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2939    2847  0 12:50 pts/2    00:00:00 ps -f
[1]+  Killed                  gedit
```

Parent and Child Processes

All the processes in Linux have two ID numbers assigned to it- Process ID (PID) and Parent process ID (PPID). Each process in the system has a parent process. Parent process is the process which starts the current process.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2855    2847  2 12:48 pts/2    00:00:00 gedit
naba        2864    2847  0 12:49 pts/2    00:00:00 ps -f
```

In the above example the terminal process id (bash) is 2847 and the other two processes gedit and ps-f are executed in the terminal, so parent process of both the processes gedit and ps-f is bash(terminal) which is shown as above. Most of the commands that we run have the shell as their parent process as we execute those commands from the shell.

The top Command

The top (table of processes) command shows a real-time view of running processes in Linux and displays kernel-managed tasks. The command also provides a system information summary that shows resource utilization, including CPU and memory usage. The top command is a very useful tool for quickly displaying the currently running processes sorted by various criteria. It is an interactive command and updates frequently by showing information about physical and virtual memory, CPU usage, load averages, and busy processes. Following example shows the top command and its output ?

$top

```
top - 12:57:12 up 24 min,  1 user,  load average: 0.32, 0.35, 0.29
Tasks: 227 total,   1 running, 226 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.8 us,  1.3 sy,  1.8 ni, 93.8 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
MiB Mem :   7820.6 total,   4625.5 free,   1162.7 used,   2032.4 buff/cache
MiB Swap:   3906.0 total,   3906.0 free,      0.0 used.   6185.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1155 naba      20   0 4180088 253204 104836 S  11.3   3.2   1:10.49 gnome-shell
   2357 _apt      25   5   24056   9368   8340 S   5.0   0.1   0:21.10 http
    856 naba      20   0  570140  88504  51956 S   4.3   1.1   1:16.89 Xorg
   2272 naba      30  10  946720 174336  99796 S   2.3   2.2   0:38.98 update-manager
    414 root     -51   0       0      0      0 S   0.3   0.0   0:19.36 irq/135-rtw88_p
    646 message+  20   0    9828   6232   3812 S   0.3   0.1   0:05.68 dbus-daemon
   2703 root      0 -20       0      0      0 I   0.3   0.0   0:00.53 kworker/u9:1-i915_flip
   3055 root     20   0       0      0      0 I   0.3   0.0   0:00.36 kworker/1:2-events
   3071 root      0 -20       0      0      0 I   0.3   0.0   0:00.11 kworker/u9:2-rtw_tx_wq
   3080 naba     20   0   12000   3872   3216 R   0.3   0.0   0:00.12 top
      1 root     20   0  167616  11552   8352 S   0.0   0.1   0:01.40 systemd
      2 root     20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root      0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root      0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root      0 -20       0      0      0 I   0.0   0.0   0:00.11 kworker/0:0H-acpi_thermal_pm
      8 root     20   0       0      0      0 I   0.0   0.0   0:00.47 kworker/u8:0-i915
      9 root      0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     10 root     20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
     11 root     20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
     12 root     20   0       0      0      0 S   0.0   0.0   0:00.07 ksoftirqd/0
     13 root     20   0       0      0      0 I   0.0   0.0   0:01.04 rcu_sched
     14 root     rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
```

10.2.5 Attributes of a process

When a program starts executing, it is called a process. When a process starts, some attributes are set to the process. These attributes are assigned by the Operating System to create the process control block (PCB) for each process. Attributes of the process include the following.

Process ID: When a process is created, a unique id is assigned to the process called the process ID or PID. PID is used to uniquely identify the process in the system.

Program counter: A program counter is a register, which contains the address (location) of the instruction being executed at the current time. As each instruction gets fetched, the program counter increases its value by 1. After each instruction is fetched, the program counter points to the next instruction to be executed. When a process is suspended, the program counter contains the instruction number to be executed after resumption of the process.

Process State: The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting. Process switches from one state to another depending on various condition. When a program is ready for execution for the first time it enters to new state, once job scheduler selects the process for execution it enters to the ready state. When processor choose the process for execution, it enters to the running state. Whenever process needs any I/O, it enters to the waiting state (waiting for I/O). If process switch occurs or processor in the waiting state completes its I/O, then again it enters to the ready state. When process completes its task, it enters to the terminate state. The process transition diagram is depicted in figure 1.

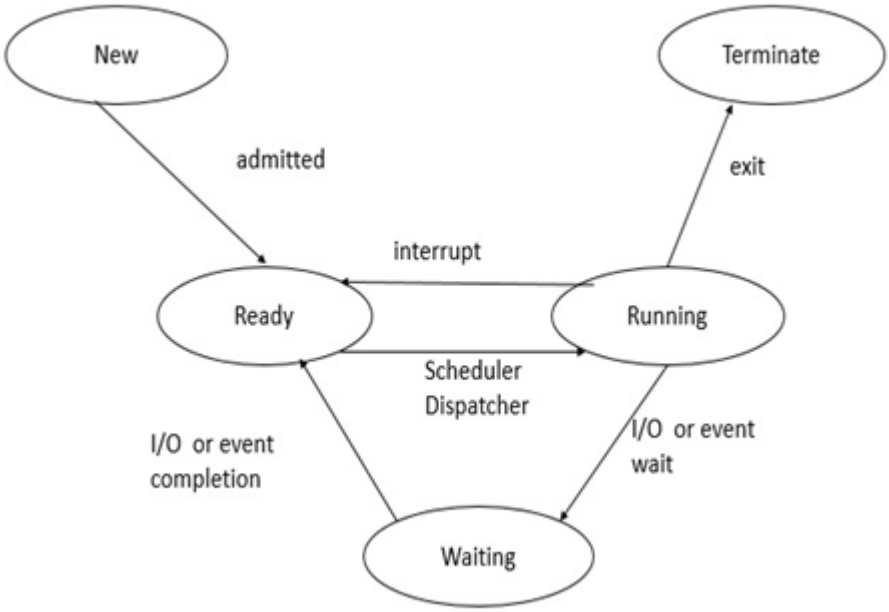


Figure 1. Process transition diagram

Priority: Every process has its own priority and this priority is set by the operating system (OS). The process with the highest priority gets the CPU first. This information is also stored on the process control block.

General Purpose Registers: Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

List of open files: During the execution, every process uses some files which need to be present in the main memory. Process control block stores the information about the open files. All the files opened by the process are closed when process is terminated.

List of open devices: The devices used by the process is maintained in the process control block. OS also maintains the list of all open devices which are used during the execution of the process.

10.2.6 Job, Task and Process

Job: A job is a complete unit of work under execution. A job consists

of many tasks which consist of many processes. A job is a series of tasks in a batch mode. Programs are written to execute a job. Jobs and tasks are used synonymously in computational work.

Task: Task is a unit of work being executed. Task in Operating System may be synonymous with process. A task is a subpart of a job. Tasks combine to form a job. The task may be a thread, process, a single job and much more.

Process: The process is a series of actions that is done for a particular purpose.

Job and task define the work to be done, whereas process defines the way the work can be done or how the work should be done.

The concept of job, process and task revolves around each other. Job, task and process may be considered the same or different in reference to the context they refer to. A process is an isolated entity of Operating System. A task may be called a process if it is a single task. A job may be called a task if the job to be performed is a single unit of work. A process or group of processes can be termed as a task and a group of tasks can be termed as a job.

10.3 JOB CONTROL

In Unix and Unix-like operating systems, job control refers to control of jobs in the shell. Basic job control features are suspending, resuming, or terminating of all processes in the job or process group. Job control is of particular interest in Unix due to its multiprocessing nature, and it is different from job control in process management.

Typically, the shell keeps a list of jobs in a job table. The jobs command lists the background jobs existing in the job table, along with their job number and job state (stopped or running). When the user exits from the shell, the shell sends SIGHUP to all jobs, and waits for the process groups to end before terminating itself.

The 'disown' command is used to remove link from the parent process and make the processes orphan, such that when the session ends,

shell should not wait for them to terminate nor shell required to send SIGHUP to those processes.

A job running in the foreground can be stopped by pressing the suspend command (Ctrl + Z). This sends the "terminal stop" signal (SIGTSTP) to the process group. By default, SIGTSTP causes processes receiving it to stop, and control is returned to the shell. A process can also be paused with the "stop" signal (SIGSTOP).

A job running in the foreground can be interrupted by typing the interruption command (Ctrl + C). This sends the "interrupt" signal (SIGINT), which defaults to terminating the process.

A stopped job can be resumed as a background job with the 'bg' command, or as the foreground job with 'fg' command. In either case, the shell redirects I/O appropriately, and sends the SIGCONT signal to the process, which causes the operating system to resume its execution.

A background process that attempts to read or write its sent terminal SIGTTIN (for input) or SIGTTOU (for output) signal. Shells often override the default stop action of SIGTTOU so that background processes deliver their output to the controlling terminal by default.

bg: Resume a suspended program without bringing it to the foreground.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs
[1]   Running                 gedit &
[2]-  Stopped                 vi
[4]+  Stopped                 vi
naba@naba-HP-Laptop-14q-cs0xxx:~$ bg %2
[2]- vi &

[2]+  Stopped                 vi
```

fg: Resumes a suspended job and brings it to the foreground.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs
[1]   Running                 gedit &
[2]+  Stopped                 vi
[4]-  Stopped                 vi
naba@naba-HP-Laptop-14q-cs0xxx:~$ fg %1
gedit
```

jobs: List the status of all running jobs

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs
[1]-  Running                 gedit &
[2]+  Stopped                 vi
```

Jobs command has following options

-l Lists process IDs in addition to the normal information.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -l
[1]-  2274 Running                 gedit &
[2]+  2352 Stopped (tty output)    vi
```

-n Displays information only about jobs that have changed status since the user was last notified of their status.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -n
[4]+  Stopped                 vi
```

-p Lists only the process ID of the job's process group leader.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -p
2274
2352
2664

[3]+  Stopped                 vi
```

-r Displays only running jobs.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -r
[1]   Running                 gedit &
```

-s Displays only stopped jobs.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -s
[2]   Stopped                 vi
[3]-  Stopped                 vi
[4]+  Stopped                 vi
```

kill: Send a signal to a process, affecting its behaviour or killing it.

wait- It waits for the process to change its state i.e. it waits for any running process to complete and returns the exit status.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ wait
bash: wait: warning: job 2[2352] stopped
bash: wait: warning: job 3[2664] stopped
bash: wait: warning: job 4[2684] stopped
```

suspend: Suspends the execution of this shell until it receives a SIGCONT signal.

10.4 CRONTAB FILE FORMAT

Cron is named after Greek word "Chronos" meant for time. Cron is a system process that will automatically perform tasks as per specific schedule. It is a set of commands that are used to run as per schedule. Crontab stands for "cron table". It allows to use job scheduler to execute tasks. Cron is a system daemon used to execute desired tasks (in the background) at designated times.

A crontab file is a simple text file containing a list of commands intended to be run at specified times. It is edited using the crontab command. The commands in the crontab file (and their run times) are checked by the cron daemon, which executes them in the background.

Each user (including root) has a crontab file. The cron daemon checks a user's crontab file regardless of whether the user is actually logged into the system or not.

Linux Crontab format

Crontab of Linux has six fields. The first five fields define the time and date of execution, and the 6th field is used for command execution.

Crontab syntax:

[MIN] [HOUR] [DOM] [MON] [DOW] [CMD]

For example

```
30 08 10 06 * gedit
31 06 * * * ps
01 * * * * jobs
```

Here, asterics (*) is used for matching, we can use range of values (for eg- 1-10 or 30-40 or jan-mar, mon-wed) or multiple ranges (apr-jun,oct-dec) in place of values. The values in each field are from the valid range of values.

Crontab file allowed ranges

Field	Description	Allowed Value
MIN	Minute field	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	1-31

MON	Month field	1-12
DOW	Day Of Week	0-6
CMD	Command	Any command to be executed.

Cron special keywords and its meaning

Keyword	Equivalent
@yearly	0 0 1 1 *
@daily	0 0 * * *
@hourly	0 * * * *
@reboot	Run at startup.

These special keywords are useful when we need to run a particular command after a definite interval.

Use of cronjobs

Reasons for using Cronjobs in Linux are considered as following-

- It helps OS to take a scheduled backup of log files or database.
- It deletes old log files.
- It archives and purges database tables.
- Send out any notification email such as Newsletters, Password expiration email.
- Regular clean-up of cached data.
- Crontab is an ideal option to automate Unix jobs.
- It is used to automate system maintenance.

10.5 LINUX BACKUP AND RECOVERY

In Linux, backup can be made using various ways such as tar, dd, SCP, etc. Tar, also known as tape archiving, is the most commonly used to create backup and restore. We can compress and archive files and directories using tar. Furthermore, we can use the tar command to create a single backup file for more than one file. The syntax of the tar command is:

tar <options> <file>

where we can use many options, which are discussed below:

c = creates a new archive

v = provides verbose output

f = specifies the archive file to use

t = lists the files in an archive

x = Extract the backup

z = Zipping

For example, if we want to create a backup file (say abc.txt) using tar, and if the backup file is to be compressed, then we can use the following command:

#tar -czvf abc.tar.gz abc.txt

If we want the system to run the backup automatically, we can use the crontab file. The crontab file contains a list of commands that can be run regularly. Every line in the crontab file is for performing a task at a particular time. By using this file, we can automate our task or schedule a task to perform at a particular time. If we want to take backup at 6 AM every morning then we need to add the following in the users crontab file.

* 6 * * * tar -czvf abc.tar.gz abc.txt

Incremental Backups with Tar:

Incremental backup is the advanced backup type using tar. Here, we do not have to perform the full back up every time. We need to back up only the data that has been created or modified since the last backup has been taken. So, using this type of backup, we can save backup time and disk space. Steps to perform the incremental backup is as follows:

At first, we have to perform the full backup. Let us consider an example where we will perform the full backup of directory data. This can be done using the following way:

#tar --verbose --verbose --create --gzip --listed-incremental=/backup/data.sngz --file=/backup/data.tgz data

where the option /backup/data.tgz will compress the data directory in tgz format and save it inside /backup directory and /backup/data.sngz option will create an initial incremental backup (or full backup) and save it at /backup/data.sngz.

After a few days and after modifying the data directory, if we want to perform the incremental backup in the data directory, we will use the following command:

#tar --verbose --verbose --create --gzip --listed-incremental=/backup/data.sngz --file=/backup/data1.tgz data

Restore the Backup with Tar Incremental Backup

To restore the incremental backup, we can use the following command:

tar --extract --listed-incremental=/dev/null --file data.tgz

Creating backup with Timeshift application

For creating system backup on Ubuntu 20.04 system, we need to install Timeshift backup utility as follows-

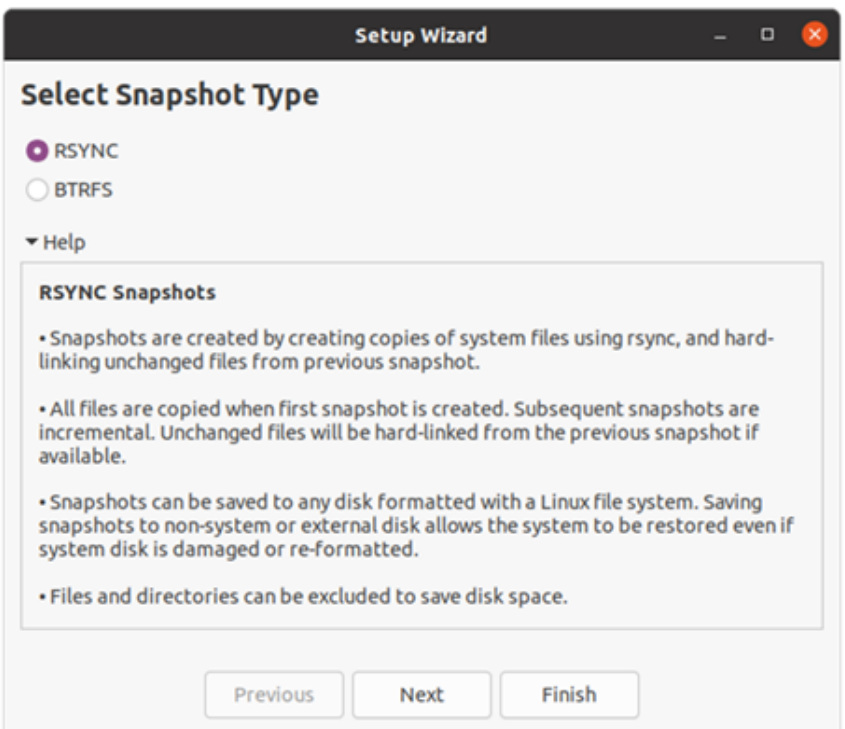
$ sudo apt install timeshift

Timeshift is a graphical user interface for creating backup and restore point on ubuntu 20.04.

Create Backup by using the Timeshift GUI

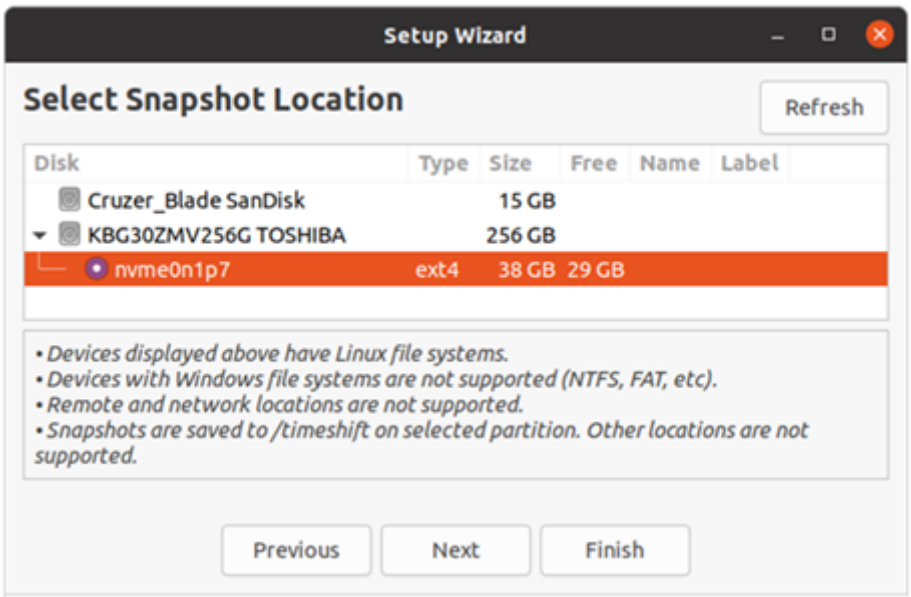
To create system backup

Step1: First to Open the timeshift application from Activities menu.



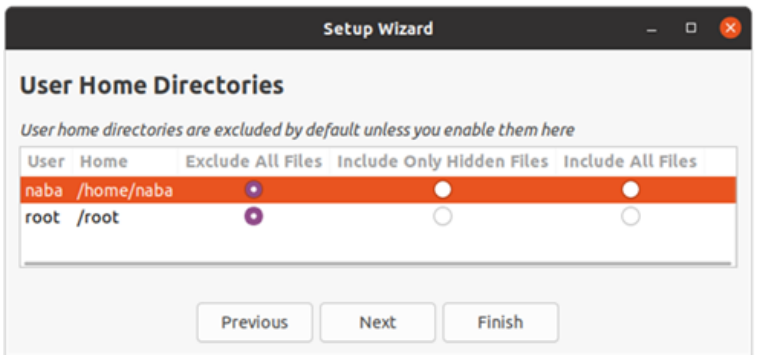
Upon opening the timeshift application user will be greeted with a wizard to help schedule the backups. Here there are two options. First is to use the rsync as a main backup tool or take an advantage of an inherent BRTFS built-in file system features. In this example, we will use rsync. Select RSYNC and hit the Next button.

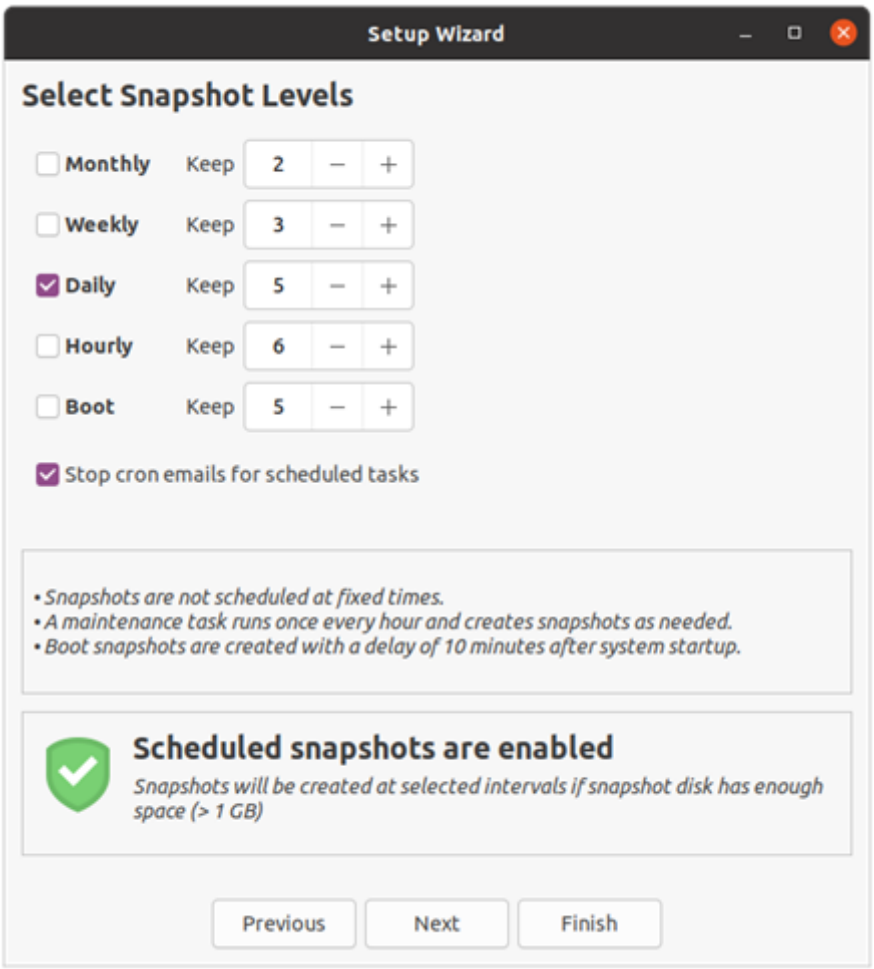
Step 2: Then, we have to select backup destination. timeshift will search our system for available file-system partition and provide us with an option to where to create backup file.



Step 3: Then to select how often we wish to perform the system backup and how many backup snapshots will be retrained before the first backup is overwritten.

Step 4: As the screenshot indicates the home directories are excluded by default. Depending on our work environment, we have to select whether we wish to include home directories into the backup.



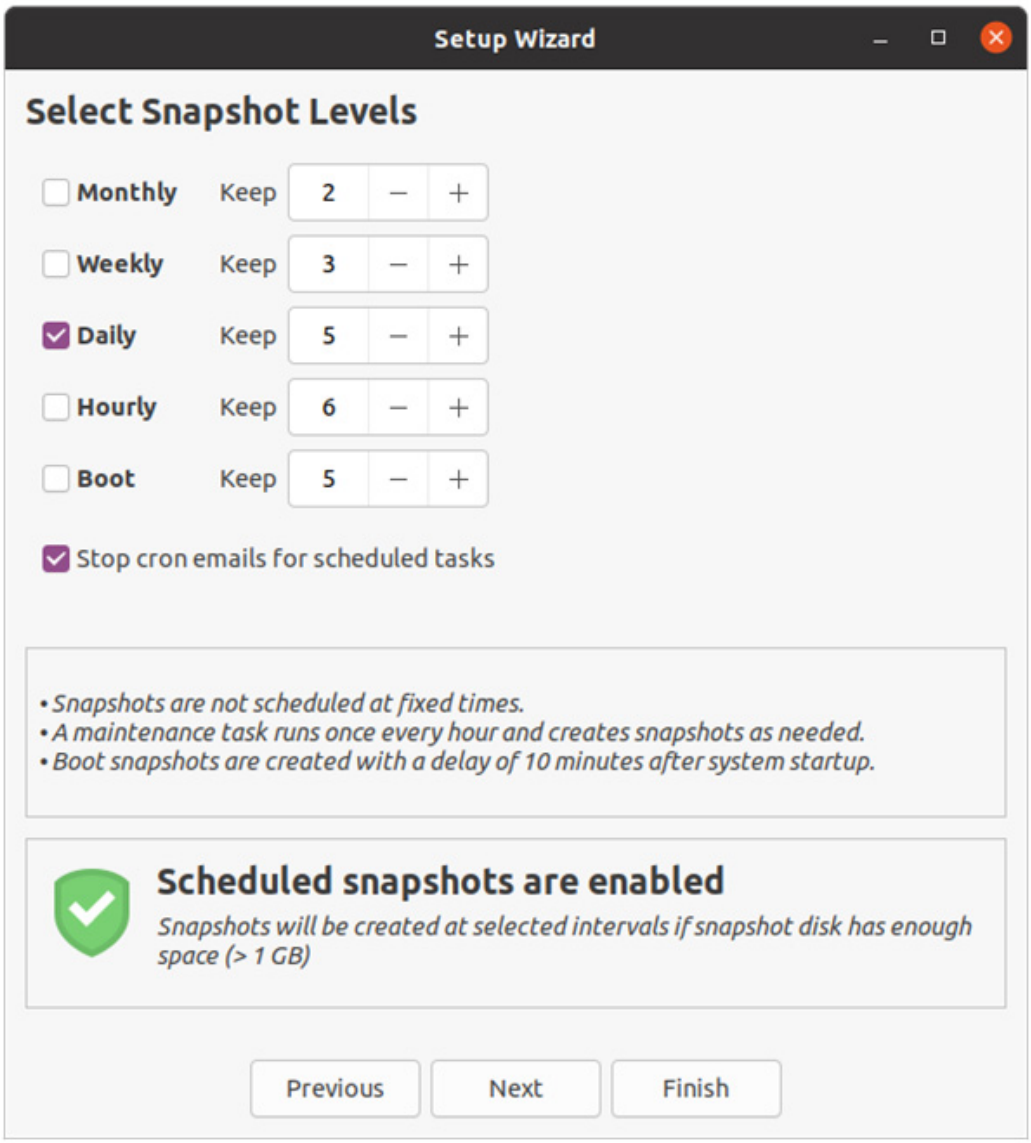


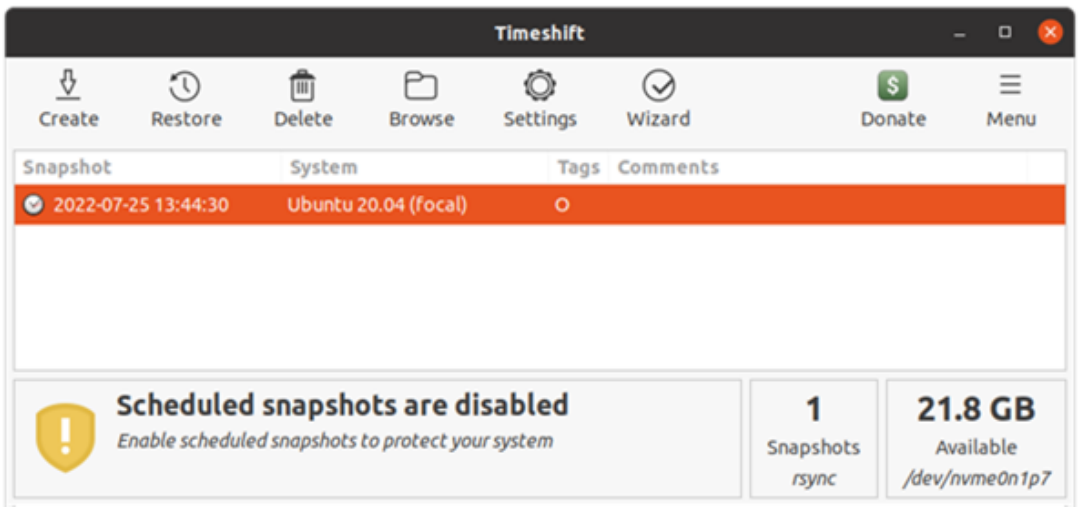
Step 5: This will conclude our initial backup schedule setup. We have to click on Finish button.

Step 6: The backup has not been created yet. We can either wait until the timeshift automatically triggers the backup or simply hit the Create button to perform the previously predefined backup now.

Step 7: We have to wait for the backup to complete.

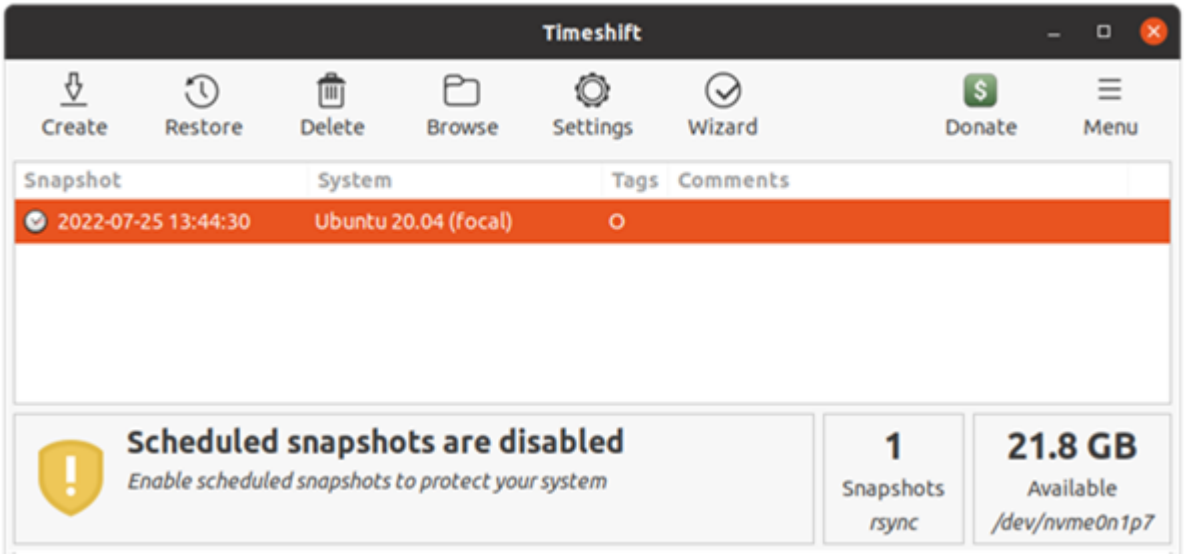
Step 8: If everything is done well, we should see our first backup snapshot listed.


Setup Wizard
Select Snapshot Levels
Monthly Keep 2
Weekly Keep 3
Daily Keep 5
Hourly Keep 6
Boot Keep 5
Stop cron emails for scheduled tasks
• Snapshots are not scheduled at fixed times.
• A maintenance task runs once every hour and creates snapshots as needed.
• Boot snapshots are created with a delay of 10 minutes after system startup.
Scheduled snapshots are enabled
Snapshots will be created at selected intervals if snapshot disk has enough space (> 1 GB)
Previous
Next
Finish

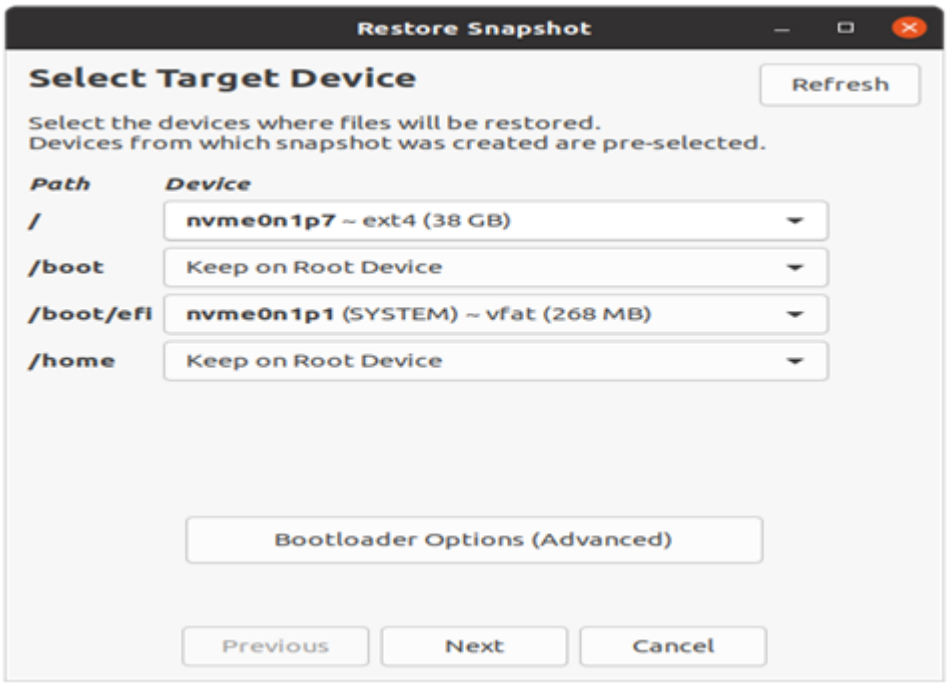

Timeshift
Create
Restore
Delete
Browse
Settings
Wizard
Donate
Menu
Snapshot
System
Tags
Comments
2022-07-25 13:44:30
Ubuntu 20.04 (focal)
Scheduled snapshots are disabled
Enable scheduled snapshots to protect your system
1
Snapshots
rsync
21.8 GB
Available
/dev/nvme0n1p7

Restore from Backup

Step 1: At this point we will restore the system from the previously created system backup snapshot. Select a backup snapshot from which we wish to restore and click on the Restore button.



Step2: Timeshift give us option on how to restore from the backup.



Step 3: The Timeshit will provide us with a list of changes it will take to restore from the backup just to make sure no data is lost in the process.

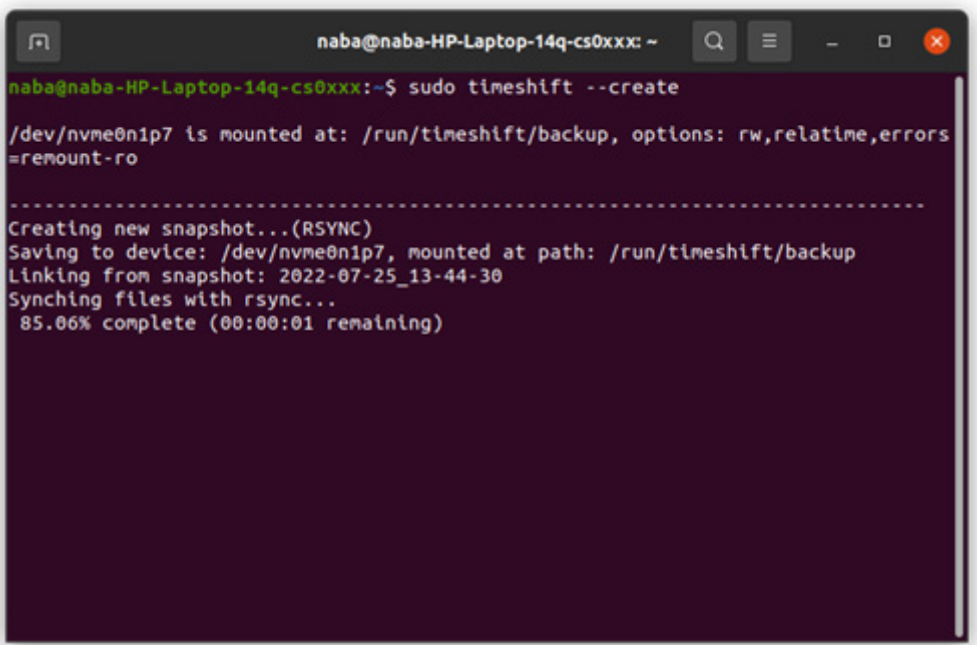
Step 4: Once we hit the Next button, the system will be restored

and restarted. All done.

Create and restore backup by using the Timeshift's command line

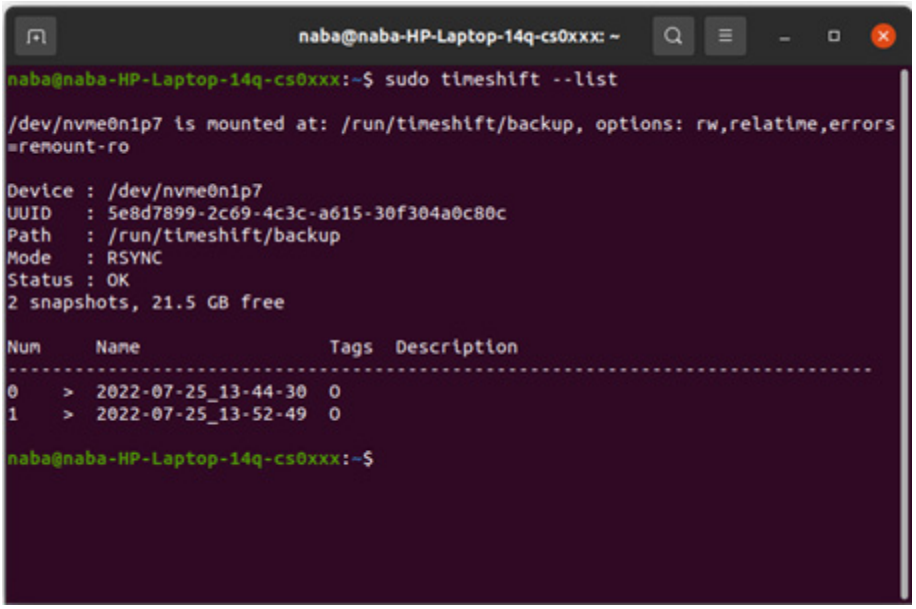
To create backup using command line, we need to execute the following command:

$ sudo timeshift --create



To List all the currently created system backup, following command is used which prints as follows:

$ sudo timeshift -list



Restore from the backup snapshot:

$ sudo timeshift --restore --snapshot "2022-07-25_13-52-49"

Here "2022-07-25_13-52-49" is the name of the backup file

Delete selected backup snapshot:

$ sudo timeshift --delete --snapshot "2022-07-25_13-52-49"



CHECK YOUR PROGRESS

Q 1: Which of the following command is used to make a background process to foreground?

GB / background / fg / bgIn

Q 2: How to run a process in the background of Linux process management?

Q 3: Process information in the current shell can be obtained by using

(gf / bg / gb / ps)

Q 4: A job is a set of task. (True / False)

Q 5: To stop a process by force, which command is used?

Q 6: Cron is named after Greek word "Chronos" means _____.

Q 7: PPID stands for _____.



10.6 LET US SUM UP

- A program in running state (execution) is called a process.
- When a program is executed in system, the system creates an environment for the program. This environment contains all the resources required by the program to ex-ecute.
- When a process starts executing, the operating system assigns a process ID (pid) to the process.
- A service is a program or application in Linux that runs in the background. A service runs without the need of user intervention and sometimes users are unaware of it.
- We can list the running processes started by current user by using the ps command.

- The top (table of processes) command shows a real-time view of running processes in Linux.
- Attributes of a process are: Process ID, Program counter, Process State, Priority, General Purpose Registers, List of open files, List of open devices.
- A job is a complete unit of work under execution. A job consists of many tasks which consist of many processes.
- In Linux, backup can be made using various ways such as tar, dd, SCP, etc.
- Backup can also be performed using Timeshift application.

10.7 FURTHER READINGS

1) Shell Scripting by Jason Cannon
2) Linux Command Line and Shell Scripting Bible by Richard Blum & Christine Bresnahan.
3) Mastering Linux Shell Scripting by Andrew Mallett & Mokhtar Ebrahim

10.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: fg

Ans to Q No 2: by adding & after the command

Ans to Q No 3: ps

Ans to Q No 4: True

Ans to Q No 5: kill -9

Ans to Q No 6: time

Ans to Q No 7: parent process ID

10.9 MODEL QUESTIONS

Q 1: Explain the different attributes of a process?

Q 2: Explain any one method of system backup and recovery in Linux.

Q 3: What is process? What are the different types of process? Explain them.

Q 4: Explain the crontab file format and use of it.

Q 5: Explain the different methods of starting a process.

Q 6: Explain the working of top command.

*** ***** ***

UNIT 11: PRINTING JOBS

UNIT STRUCTURE

11.1 Learning Objectives

11.2 Introduction

11.3 Set up a Local Printer

- 11.3.1 Installation of Printer
- 11.3.2 Set the Default Printer
- 11.3.3 Installation of Network Printer

11.4 Printing

- 11.4.1 Print Only Certain Pages
- 11.4.2 Print Pages in a Different Order
- 11.4.3 Change the Paper Size When Printing
- 11.4.4 Print a Booklet
- 11.4.5 Print a File

11.5 Troubleshooting a Printer

11.6 Let Us Sum Up

11.7 Further Readings

11.8 Answer to Check Your Progress

11.9 Model Questions

11.1 LEARNING OBJECTIVES

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

- Install printers in Linux Operating System
- Print documents in different printers
- Know different property settings of the printer

11.2 INTRODUCTION

In the previous unit, we have learnt about processes and the various tasks associated with processes. In this unit, we will discuss how to work with printer. In the subsequent sections, we will discuss the installation of a printer and the various activities to be carried out while printing documents.

11.3 SET UP A LOCAL PRINTER

Linux operating system recognizes many types of printers automatically once they are connected to the computer. Most of the printers are connected via Universal Serial Bus (USB) directly to the computer, but some printers are connected via network (wired or wireless).

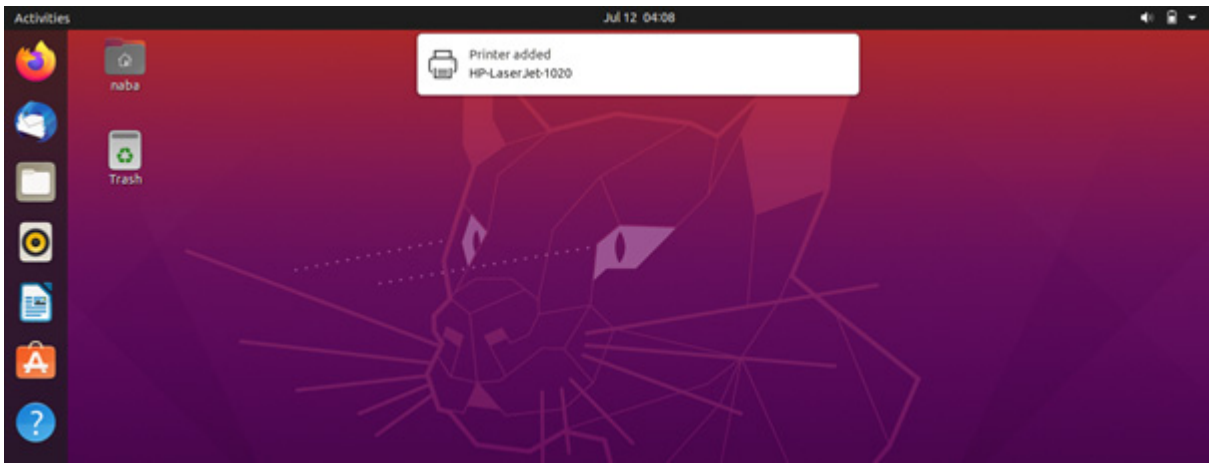
If the printer is connected to the network, then user should install it from the Printers panel in Settings.

11.3.1 Installation of Printer

Automatic installation of printer -

- Turn on the printer.
- Connect the printer to the computer via appropriate cable.
- Once the printer is connected to the computer, it will automatically search for drivers of the printer. If driver is found it will ask the user to authenticate it and it will automatically install the printer.
- Once installation is complete, a popup message will display that the system has finished the installing of printer.

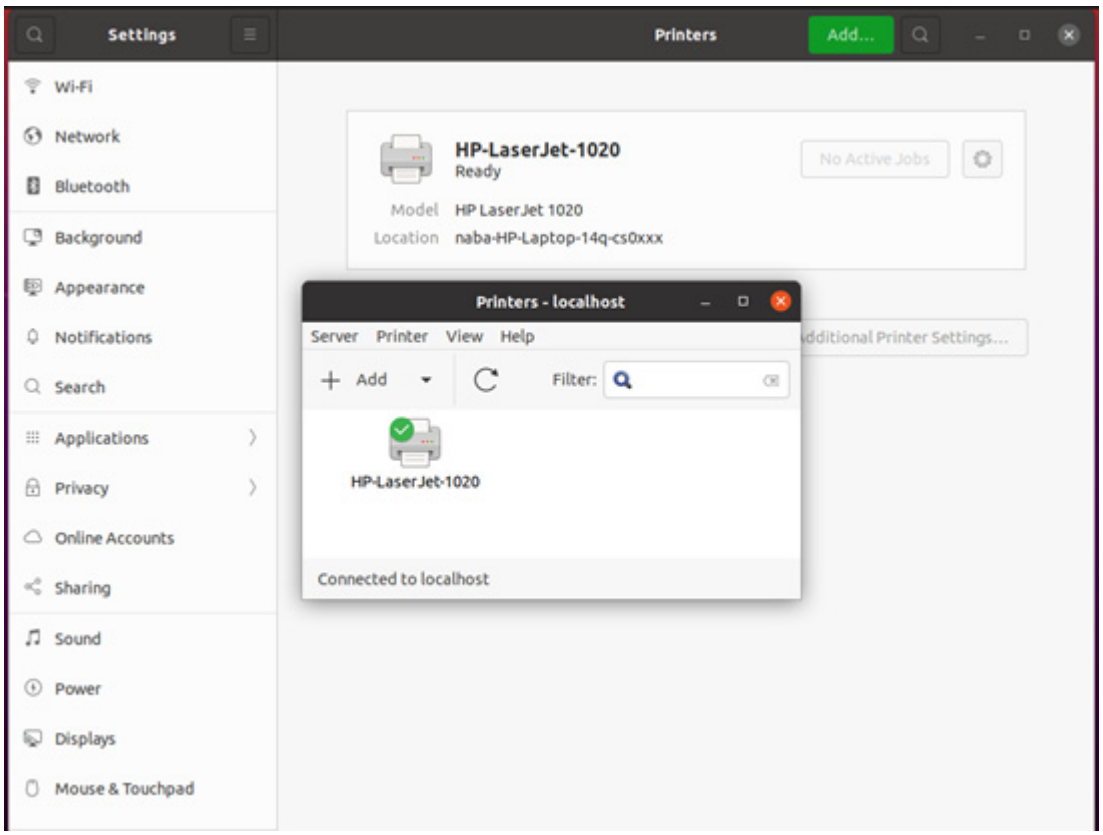
  Now, we can print the "Test Page" and can make additional changes in the printer setup.



Manual installation of printer -

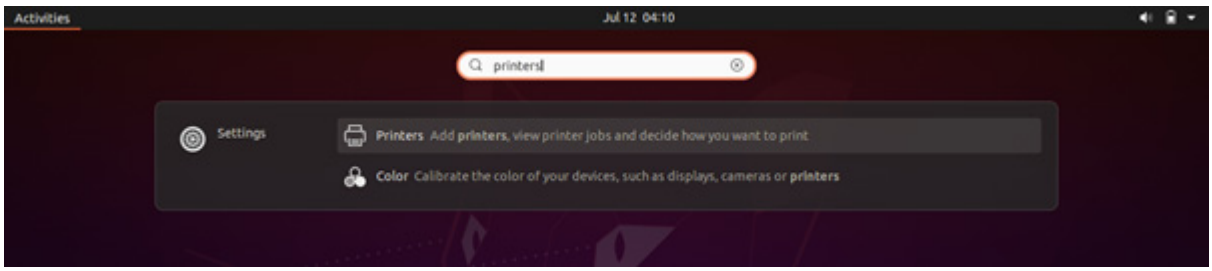
If the printer is not set up automatically, then we can add it from the printer settings. For manual installation of printer, we need to follow the steps given below:

- Open the Activities overview and type Printers.
- Select the "Printers" from menu.
- If the current user is not administrator (root user), it will ask for administrator permission.
- Select the Add from the menu. (Add / Add New / Add Printer differs in different OS).
- In the pop-up window, select the printer and press Add. (System will display preinstalled drivers in the list)
- If the printer is not discovered automatically, user needs to install the printer driver. Printer drivers can be downloaded from the printer's manufacturer (HP/ EPSON/ Canon/ Brother) website and we need to install it to the computer before further proceed.



What is Activities overview-

When we start GNOME, we automatically enter the Activities overview. The overview allows us to access the windows and its applications. In the overview, we can search a software/system just by typing name of the applications, files, folders, etc.



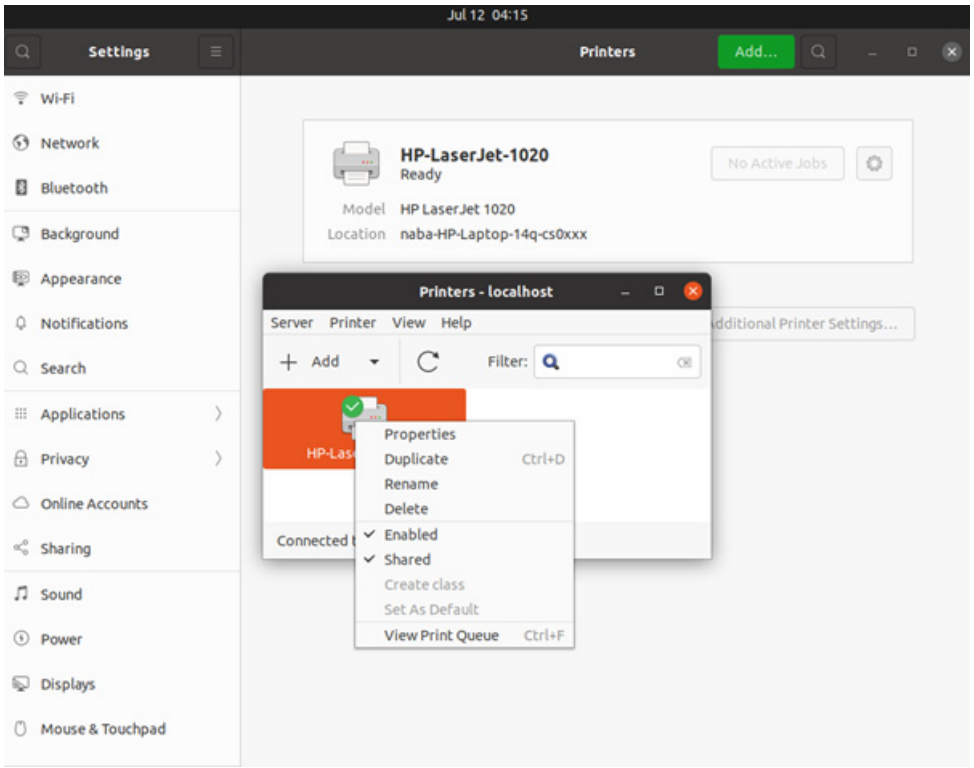
11.3.2 Set the Default Printer

If the system has more than one printer installed, then user needs to select the default printer to perform printing jobs. We select the printer used most often as the default printer.

To set the default printer, we need the administrative privileges (root user). To set the default printer we need to follow the following steps-

- To Open the Activities overview and start type of Printers.
- To Select Printers from the menu.
- To Click the settings button next to the printer.
- To Select the Use Printer by Default checkbox.

After setting of default printer, any job submitted for printing without selecting the printer will be printed by the different printer.



Change the name or location of a printer-

To change the name or location of a printer, we need the administrative privileges (root user). To set the name or location of the printer we need to follow the following steps-

- Select the printer from Printers by selecting the Printers from Activities overview.

Change printer name

- Click the settings button next to the printer.
- Select Printer Details.
- Enter the new name in the Name field.
- Close the dialog.

Change printer location

- Click the settings button next to the printer.
- Select Printer Details.
- Enter the new location for the printer in the Location field.
- Close the dialog.

11.3.3 Installation of Network Printer

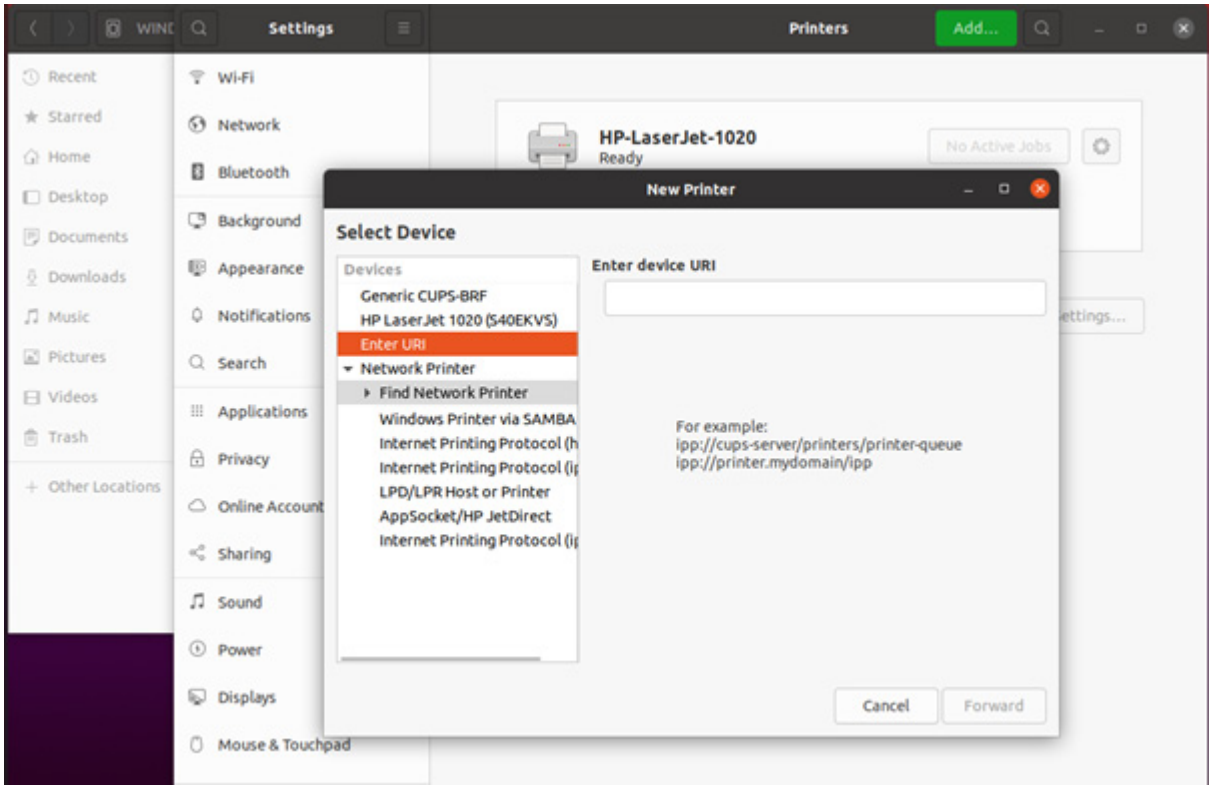
Ubuntu is a network-based operating system, it has built-in tools to help users to quickly locate and utilize the network resources like printers, databases, file servers etc. For installation of a network printer, we need to follow the basic steps-

- Open the Activities overview and type Printers.
- Select the "Printers" from menu.
- If the current user is not administrator (root user) it will ask for administrator permission.
- Select the Add from the menu. (Add / Add New / Add Printer differs in different OS)
- Select "Network Printer" under "Devices," then select "Find Network Printer."
- Type the network printer's IP address into the input box labelled "Host," then select the "Find" button.

- Select the printer that needs to install from the list under "Devices."
- Select the "Forward" button and select "Choose Driver" and select the appropriate driver.
- If system asks for authentication, then type the administrative password and select authenticate.
- Select the printer's manufacturer from the drop-down list, and select the printer's model from the drop-down list.
- Press "Apply"/ "OK" button to keep the setting.

Once Installation is complete, we can print the "Test Page" and can make additional changes in the printer setup.

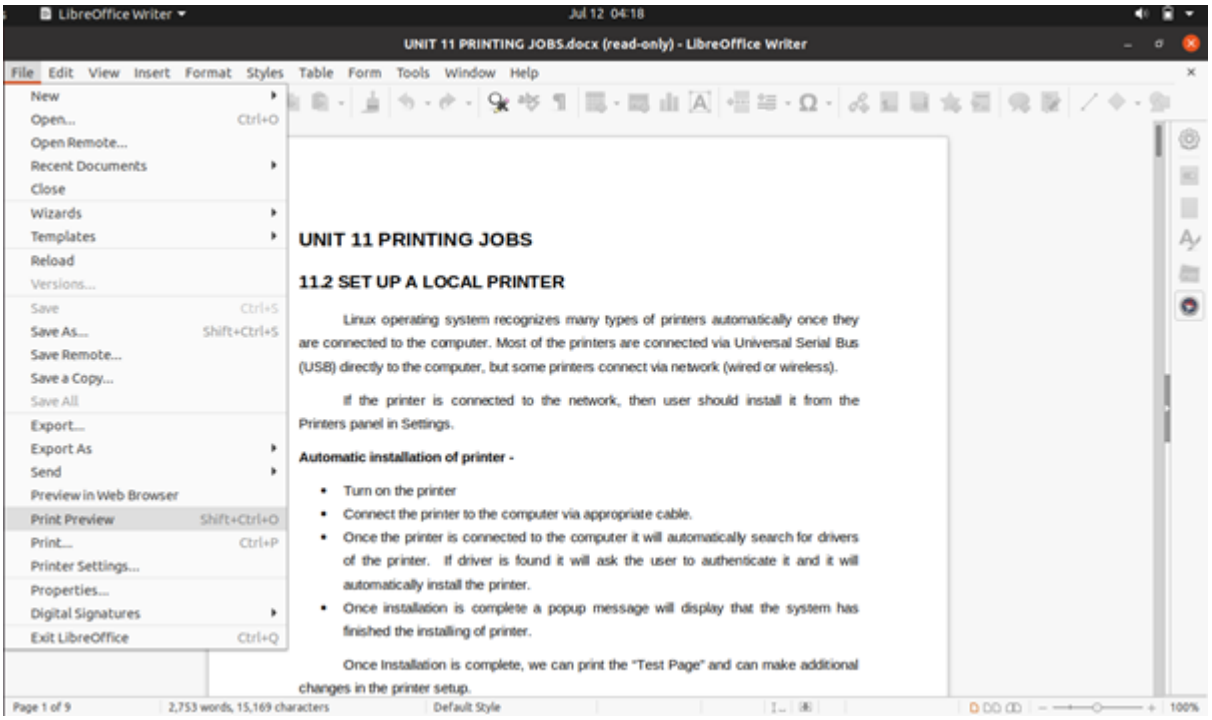
Note- If another Ubuntu or Windows PC on the network is already connected to the printer, the device may already be on the Devices list before we perform the search. In that case we may not have to install the printer we can just use it.

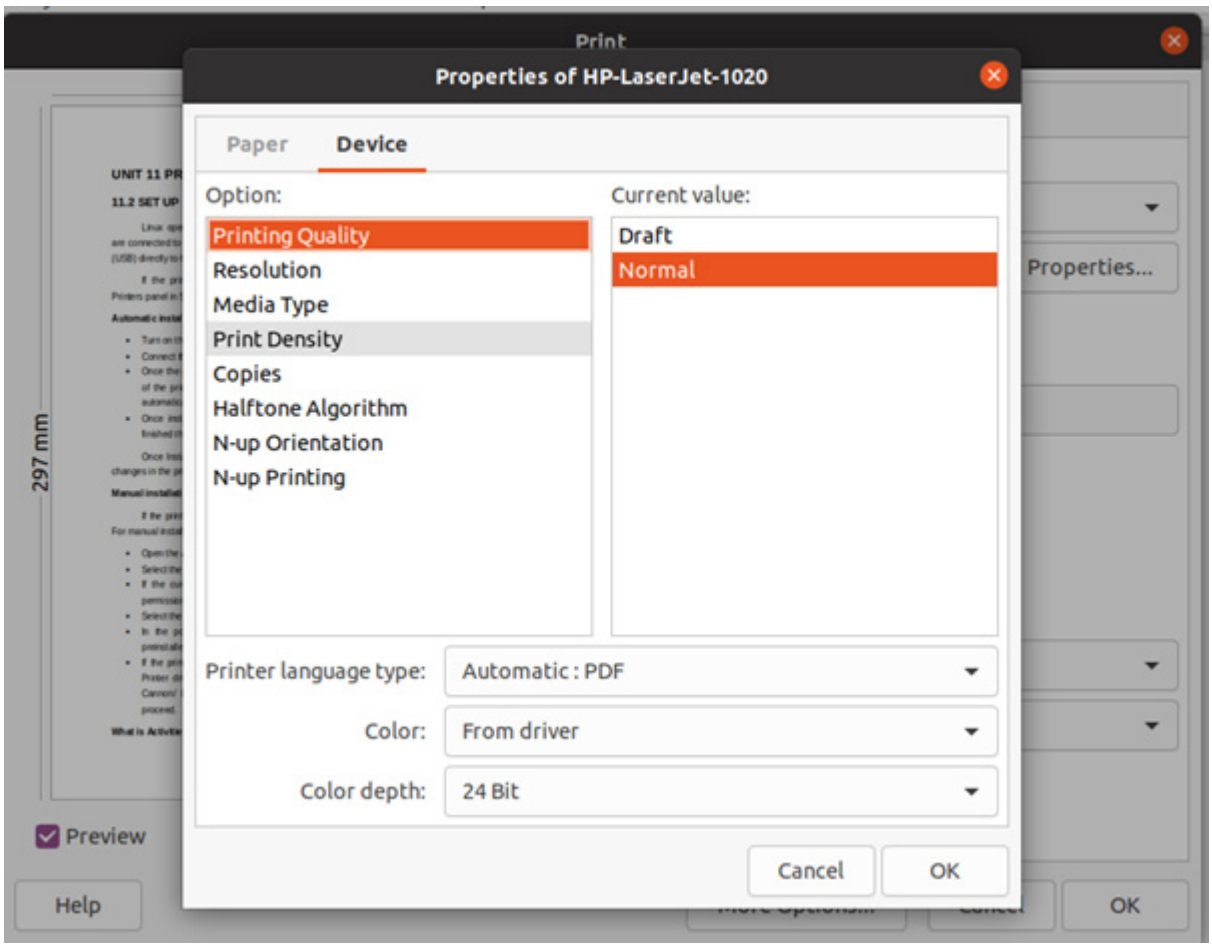


11.4 PRINTING

For print the document / file, we need to follow the steps cited below-

- Open the print dialog by pressing Ctrl+P or by selecting print dialog from file menu.
- Select the printer to print the document, if we don't select anyone, document will be printed by the default printer.
- We can change the Page type, page orientation, page order and few more setting in the print dialog menu as required to us.
- After finishing the setup, click the print button to get the printout copy.





11.4.1 Print Only Certain Pages

To only print certain pages from the document:

- Open the print dialog by pressing Ctrl+P.
- In the General tab, choose Pages from the Range section.
- Type the numbers of the pages to print in the text box, separated by commas. Use a dash to denote a range of pages.
  For example, if you enter "1,3,5-7" in the Pages text box to print pages 1,3,5,6,7

11.4.2 Print Pages in a Different Order

Reverse

- Printers usually print the first page first, and the last page last, so the pages end up in reverse order when we pick them up. If

required, we can reverse this printing order. To reverse printing order, follow the steps-

- Press Ctrl+P to open the Print dialog.
- In the General tab, under Copies, check Reverse. The last page will be printed first, and so on.

Collate

If we have to print more than one copy of the document, the print-outs will be grouped by page number by default (that is, all of the copies of page one come out, then the copies of page two, and so on). Collating will make each copy come out with its pages grouped together in the right order instead. To collate, -

- Press Ctrl+P to open the Print dialog.
- In the General tab, under Copies, check Collate.

Print envelopes

Most printers will allow us to print directly onto an envelope. This is especially useful if we have a lot of letters to send. There are two things we need to check when trying to print onto an envelope.

- The first one is that the printer knows the size of the envelope. To select the size in the Print dialog, go to the Page Setup tab and choose the Paper type as "Envelope" and then set the Paper size to the size of the envelope. Sometimes matching size is not available, in that case custom size needs to be set according to the requirement.
- Secondly, we need to make sure that the envelopes are loaded with the right side up in the printer's in-tray.

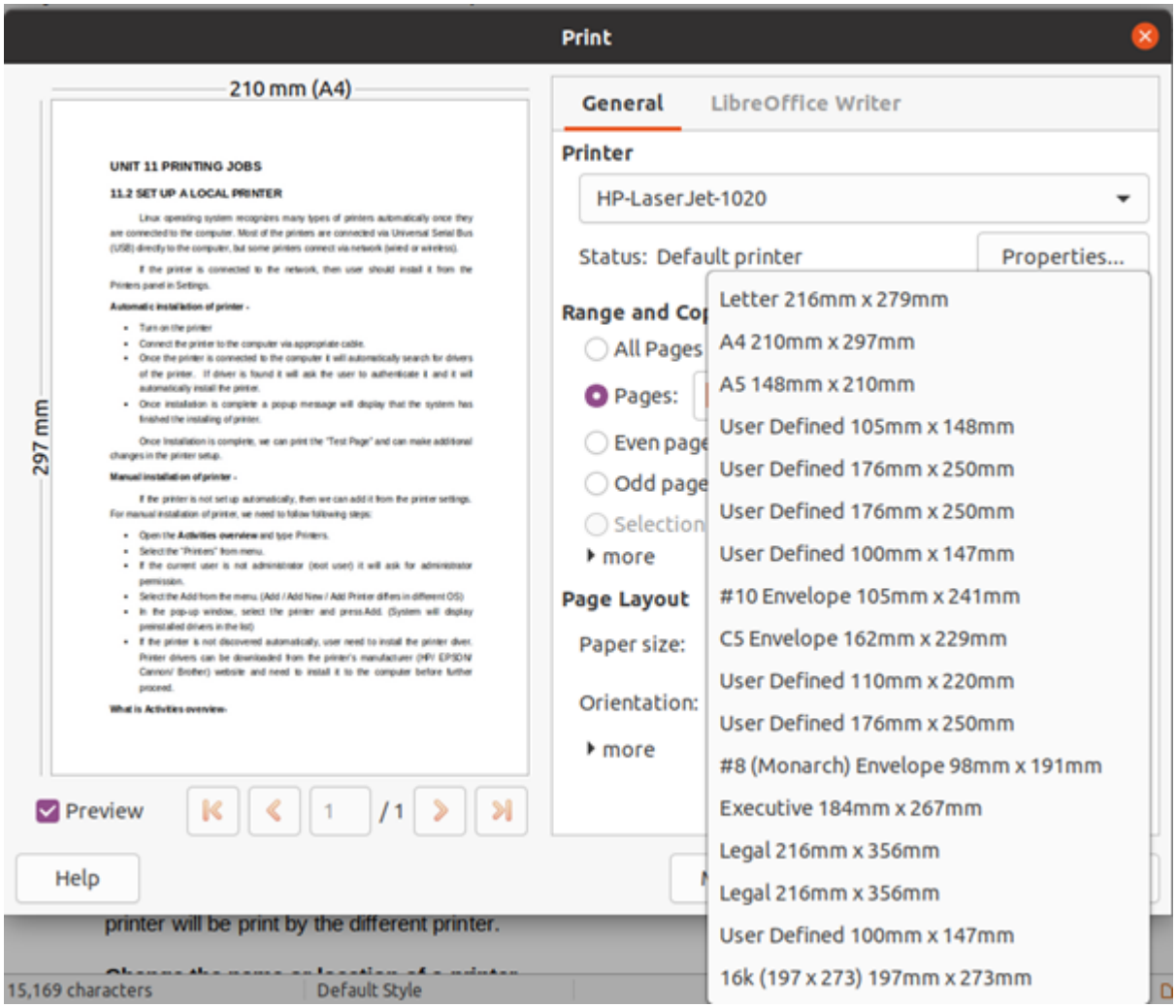
  Some printers are not designed to handle envelopes printing, especially some laser printers. If printer does not support envelop printing, it may damage the printer if we forcefully try to print it.

11.4.3 Change the Paper Size When Printing

If we want to change the paper size of the document (for example, print a US Letter-sized PDF on A4 paper), we can change the printing

format for the document as described below.

- Open the print dialog by pressing Ctrl+P.
- Select the Page Setup tab.
- Under the Paper column, choose the Paper size from the drop-down list.
- Click on Print to print your document.



we can also use the Orientation in the drop-down list to choose a different orientation:

- Portrait
- Landscape
- Reverse portrait
- Reverse landscape

Print two-sided and multi-page layouts

To print on both sides of each sheet of paper:

- Open the print dialog by pressing Ctrl+P.
- Go to the Page Setup tab of the Print window and choose an option from the Two-sided drop-down list. If the option is disabled, two-sided printing is not available for that printer.

Some printers do not have duplex printing facility. In that case we need to put back the one side printed page back to the printer as instructed by the printer.

Few printers do not support the both side printing facility; in those printers we can manually print first side of the page and then print the second side of the page, but we should be careful in selecting the pages to print on second side when we manually do that.

11.4.4 Print a Booklet

If we want to print a booklet from a LibreOffice document, first we need to export it to a PDF by choosing File -> Export as PDF. To print a booklet, the document needs to have multiple of 4 number of pages (4, 8, 12, 16,…). We may need to add blank pages to make it multiple of 4. After creating the PDF file, we can print the file as a booklet by selecting the printer

Print a booklet on a double-sided printer:

- Open the print dialog. Either by selecting Print from menu or by typing Ctrl+P.
- Click the Properties button of the print dialog
- In the Orientation drop-down list, make sure that Landscape is selected.
- In the Duplex drop-down list, select Short Edge.
- Click OK to go back to the print dialog.
- Under Range and Copies, choose Pages.
- Type the numbers of the pages in this order (n is the total number

of pages, and a multiple of 4):

n, 1, 2, n-1, n-2, 3, 4, n-3, n-4, 5, 6, n-5, n-6, 7, 8, n-7, n-8, 9, 10, n-9, n-10 …

Examples:

4 page booklet: Type 4,1,2,3

8 page booklet: Type 8,1,2,7,6,3,4,5

20pagebooklet: Type 20,1,2,19,18,3,4,17,16,5,6,15,14,7,8,13,12,9,10,11

- Choose the Page Layout tab.
- Under Layout, select Brochure.
- Under Page Sides, in the Include drop-down list, select All pages.
- Click Print.

Print a booklet on a single-sided printer:

- Open the print dialog. Either by selecting Print from menu or by typing Ctrl+P.
- Click the Properties button of the print dialog
- In the Orientation drop-down list, make sure that Landscape is selected.
- Click OK to go back to the print dialog.
- Under Range and Copies, choose Pages.
- Type the numbers of the pages in this order (n is the total number of pages, and a multiple of 4):

n, 1, 2, n-1, n-2, 3, 4, n-3, n-4, 5, 6, n-5, n-6, 7, 8, n-7, n-8, 9, 10, n-9, n-10…

Examples:

4 page booklet: Type 4,1,2,3

8 page booklet: Type 8,1,2,7,6,3,4,5

12 page booklet: Type 12,1,2,11,10,3,4,9,8,5,6,7

16 page booklet: Type 16,1,2,15,14,3,4,13,12,5,6,11,10,7,8,9

20 page booklet: Type 20,1,2,19,18,3,4,17,16,5,6,15,14,7,8,13,12,9,10,11

- Choose the Page Layout tab.
- Under Layout, select Brochure.
- Under Page Sides, in the Include drop-down list, select Front sides / right pages.

- Click Print.

When all the pages have been printed, flip the pages over and place them back in the printer.

- Open the print dialog. Either by selecting Print from menu or by typing Ctrl+P.
- Choose the Page Layout tab.
- Under Page Sides in the Include drop-down list, select Back sides / left pages.
- Click Print.

11.4.5 Print a File

Sometimes we need to print the document / data into a file instead of sending it to print from a printer. Printing to file will create a PDF, a PostScript or a SVG file that contains the document. This is useful if we want to send the document online or to another machine or to share it with someone. To print the document to a file we need to follow the steps as cited-

- Open the print dialog by pressing Ctrl+P or by selecting print dialog from file menu.
- Select Print to File in the Printer menu.
- To change the default filename and where the file to be saved, click the filename and set the new file name and select the location in the print dialog.
- After finishing the setup, click the print button to save the file.

PDF is the default file type for the document. If we want to use a different Output format, select either PostScript or SVG.

11.5 TROUBLESHOOTING A PRINTER

How to check printer's ink or toner level?

Checking of ink or toner level depends on the model and manufacturer of the printer, and the drivers and applications installed on the computer. Some printers have a built-in screen to display ink levels

and other information.

Some printers report toner or ink levels to the computer, which can be found in the Printers Settings panel. The ink level is shown with the printer details if it is available.

Alternatively, we can install applications to check or monitor ink levels. Inkblot shows ink status for many HP, Epson and Canon printers. Another ink levels application for Epson and some other printers is mtink.

Clearing a paper jam

Sometimes due to incorrect feeding of paper or low quality or damaged pages, printer get jammed. In the manual of the printer, detailed instructions are given about how to clear paper jams. In general, we need to open the printer's panels to find the jam inside and then firmly (but carefully!) pull the jammed paper out of the printer's feeding mechanism.

Once the jam has been cleared, we may need to press the printer's resume button to start printing again. In some printers, we may even need to turn the printer off and then on again, and then start the print job again.

Why are there streaks, lines or the wrong colors on my print-outs?

If the print-outs are streaky, faded, have lines on them that should not be there, or of poor quality, this may be due to a problem in the printer or low ink.

Fading text or images

This is mainly caused due to low ink. To solve this issue, check the ink levels and change the cartridge/ toner if necessary. If the ink levels are sufficient but still it showing the same problem then we need to clean the ink head. In the user manual, detail instructions for ink head cleaning is given.

Streaks and lines

This problem is mainly caused due to a dirty or partially blocked ink head. To solve this, we need to clean the ink head. In the user manual, detail instructions for ink head cleaning is given.

Wrong colors

This is mainly caused due to low ink or run out of one color from one of the cartridge / toners if we have multiple cartridges / toners. To

solve this issue, check the ink levels and change the cartridge/ toner if necessary.

Jagged lines, or lines are not straight

If lines are printed or writings are jagged not in straight line then we need to align print head. In the user manual detail instructions for aligning print head is given.

Cancel a print job

If we accidentally start printing a document that we don't require, we can cancel the print. To cancel a print job-

- Open the Activities overview and type Printers.
- Select Printers to open the list of printers.
- Click on the Show Jobs on the right-hand side of the Printers dialog.
- Cancel the print job by clicking the stop button.

If this does not cancel the print job then switch off and on the printer. But when we switch off the printer make sure that no pages are in the printing area otherwise it will cause a paper jam or damage the printer head.

Pause and release a print job

If we want to pause or release a print job, same procedure for cancel job need to be followed. Only instead of "Cancel the print job by clicking the stop button" we need to press pause printing or resume printing.



CHECK YOUR PROGRESS

Q 1: Shortcut key printing is _____

Q 2: We can set up a new printer from Printers in _____

Q 3: We cannot change the name of the printer. (TRUE/ FALSE)

Q 4: We can monitor ink level from Computer. (TRUE/ FALSE)



11.6 LET US SUM UP

- Linux operating system recognizes most of the printers automatically

once they are connected to the computer.

- If the OS recognizes the printer, it automatically installs the driver for the printer.
- If OS does not recognize the printer, we need to install from Printers by using Add New Printer window
- We can set a printer as default printer so that all the documents are get printed via that printer.
- In network-based operating system, it has built-in tools to locate and install network printers.
- We can print any document by using Ctrl + P. If we don't select the printer documents get printed in default printer.
- We can select certain pages to print instead of printing the whole document in the printing menu.
- We can set different printing properties such as page-size, page-orientation etc to get the printout as we wish.
- We can print to a file (as a soft copy) instead of printing to a printer (as hard copy)



11.7 FURTHER READINGS

1) Linux Administration Cookbook By Adam K. Dean
2) Linux Administration By Jason Cannon



11.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: Cntrl + P

Ans to Q No 2: Activities overview

Ans to Q No 3: FALSE

Ans to Q No 4: TRUE



11.9 MODEL QUESTIONS

Q 1: Write the steps for Manual Installation of printers in Linux environment.

Q 2: Explain the installation process of network printers.

Q 3: Describe the process of Booklet printing.

Q 4: Describe the steps of setting a default printer.

Q 5: Describe the steps for manual duplex printing.

*** ***** ***

UNIT 12: SYSTEM ADMINISTRATION TASKS

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 Getting Started With Your System

12.4 Users Account and Group

12.5 System Monitoring

12.6 Getting System Information With Uname

12.7 Backup and Restore Files

12.8 kudzu

12.9 X-windows

12.10 Gnome graphical interfaces

12.11 KDE Installation and Configuration

12.12 Let Us Sum Up

12.13 Further Readings

12.14 Answer to Check Your Progress

12.15 Model Questions

12.1 LEARNING OBJECTIVES

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

- Understand the login process of Linux
- create, modify and remove users and groups
- learn system monitoring tools
- learn to navigate X windows, GNOME and KDE

12.2 INTRODUCTION

In the previous unit, we have learnt how to install and configure a printer in Linux. In this unit, we will discuss some common system administration task. The job of a Linux system administrator is to manage the operations of a computer system like to maintain, enhance, create user account/report, taking backups using Linux tools and command-line

interface tools. So, in this unit we will mainly focus on system administrative tasks like managing user accounts, granting access, checking system performance, and taking corrective measures. At first, we will go through the login process.

12.3 GETTING STARTED WITH YOUR SYSTEM

In Linux, the login page is handled by the GNOME Display Manager (GDM) or the Light. When we press the power button, an electrical signal passes through the system's motherboard and awakens the entire hardware system. As we all know, the Linux kernel works in a very different way than the traditional booting system. In Linux, the boot process includes a few steps. The booting process activates the kernel user mode that subsequently activates the BIOS, MBR, boot menu, GRUB, and the login page. When the power button gets ON, it powers up the BIOS, and then the BIOS starts looking for a boot device to run the operating system. BIOS then invokes GRUB (Grand unified bootloader) from MBR (Master boot records). MBR partition holds not only GRUB, but it also stores the boot records, the booting related file and data about all the other storage devices and how they will be acting on the Linux system. The Initial RAM Disk (intrd) is designed to be used as a temporary filesystem for a Linux system to find the mounted EFI boot files. In most Linux distributions, the bootloader files are stored inside the /boot/efi directory. When the bootloader is executed, a login stage appears to enter into the Linux system. When the kernel is first loaded into memory, it initializes itself and the hardware that may be attached to the computer. The run level state of a Linux system is defined as when our Linux system has finished the booting process and is ready to be used. Simply, that immediate state of a computer where the power options, user-mode option, and the entire environment can be operated is known as the run level state.

Once the kernel is established enough to be able to run programs, it does. The first program is called "init"; its job is to function as the grandparent of all processes.

When init starts, it reads a file called inittab, usually located in /etc. This file tells init which programs should be run under which conditions. Not only does init run the startup scripts that bring the rest of the system up, but init also takes care of shutting the system down. After that init starts getty process which in turn displays login prompt. The entire login process can be summarized as:

Login process steps

Init starts getty process

getty process initiates login prompt on terminal

login command check user credentials from /etc/passwd

getty starts user shell process

shell reads the system wide files /etc/profile, /etc/bashrc

Shell reads user specific files .profile, .login

Now it reads shell specific configuration file .bashrc

Shell displays the default prompt

Init starts getty

When Linux system boots, it goes through various booting stages. In last stage it starts init, which reads a file called inittab which is located in /etc, where it finds out in which run level it has to execute. Once init process completes run-level execution and executing commands in /etc/rc.local, it will start a process called getty. Getty is the process which will take care of complete login process.

Getty shows login prompt

getty is short for "get terminal". A getty program initiates login command, it opens the terminal device, initializes it, prints login: and waits for a user name to be entered.

getty starts /etc/login

Once user enters his login name, getty starts /etc/login, this in-turn will prompt for user password. The password will be hidden and will not be shown on screen.

getty verifies the credential and starts users shell

In next stage getty checks the user credentials by verifying with /etc/passwd and /etc/shadow file; if password matches then it initiates user

properties gathering and starts user's shell. If password doesn't match then getty terminates login process and re-initiates again with new login: prompt. In next stage, the getty process reads the user properties (username, UID, GID, home directory, user shell etc.) from /etc/passwd file. After that, it reads /etc/motd file for displaying content banner message.

Shell reads system wide default files and specific default files

For getting shell related properties, user aliases and variables, getty process reads appropriate system wide default files /etc/profile or /etc/bashrc. After the system wide default files are read, the shell reads user specific login files.profile or.login.

Shell specific file read

At last stage, it reads shell specific configuration files (.bashrc,.bash_profile etc. for bash shell) of that user which it gets on the users home directory.

Shell Prompt

When all startup files are read, the shell displays the default prompt, normally PS1 prompt for user to execute their commands.

12.4 USERS ACCOUNT AND GROUP

To add a new user account, we can run either of the following two commands as root.

- adduser
- useradd

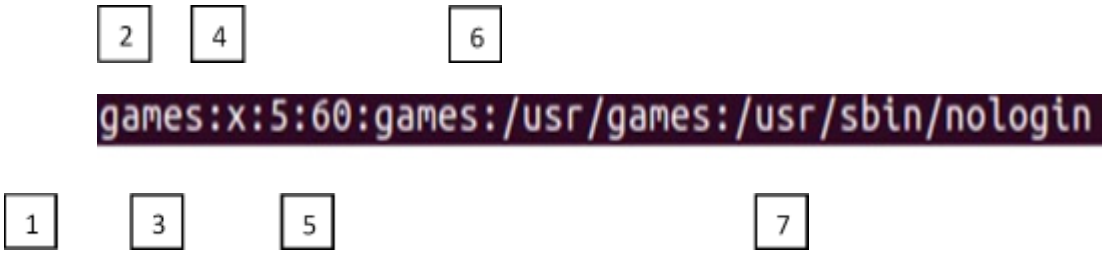
When a new user account is added to the system, the following operations are performed.

- His/her home directory is created (/home/username by default).
- The following hidden files are copied into the user's home directory, and will be used to provide environment variables for his/her user session.
- .bash_logout
- .bash_profile
- .bashrc

- A mail spool is created for the user at /var/spool/mail/username.
- A group is created and given the same name as the new user account.

Understanding /etc/passwd file format:

When user enters login ID and password, entries are verified against the information stored in /etc/passwd file. The entries in the /etc/passwd file have some information like username, userid, home directory and the first program that should be started after login succeeds.



- **Username:** It is used when user logs in. It should be between 1 and 32 characters in length.
- **Password:** An x character indicates that encrypted password is stored in /etc/shadow file. Please note that we need to use the passwd command to compute the hash of a password typed at the CLI or to store/update the hash of the password in /etc/shadow file.
- **User ID (UID):** Each user must be assigned a user ID (UID). UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups.
- **Group ID (GID):** The primary group ID (stored in /etc/group file)
- **User ID Info (GECOS):** As the comment field, it allows to add extra information about the users such as user's full name, phone number etc. This field is used by finger command.
- **Home directory:** The absolute path to the directory the user will be in when they log in. If this directory does not exist, then user's directory becomes /.
- **Command/shell:** The absolute path of a command or shell (/bin/bash). Typically, this is a shell. Please note that it does not have to be a shell. For example, sysadmin can use the nologin shell, which acts as a

replacement shell for the user accounts. If shell sets to /sbin/nologin and the user tries to log in to the Linux system directly, the /sbin/nologin shell closes the connection.

So general format is like username: password: UID:GID: comment: home: shell

Understanding /etc/group

Group information is stored in the /etc/group file. Each record has the following format.

[Group name]:[Group password]:[GID]:[Group members]

- [Group name] is the name of group.
- An x in [Group password] indicates group passwords are not being used.
- [GID]: same as in /etc/passwd.
- [Group members]: a comma separated list of users who are members of [Group name].

Creating users

For this, we will be making use of the useradd command. This command is pretty flexible and allows us to create users that can login or even users that cannot login (in the case of creating a user for a software installation only).

The basic syntax of the command is:

useradd [options] username

Suppose we want to create the user "test" such that it has a home directory and can log in. The command will be

$sudo useradd test

where sudo (super user do) is the command which allows us to be the super user. The user would be added, without a home directory and be locked out of logging in. If we want to create the user and also create the user's home directory to match the username then following command will do the magic.

$sudo useradd -m test

So, if we look in the /home directory, we will now see test. Now to add password to the account, either one of two ways can be used. If we

have already created the user, we can issue the command:

$sudo passwd test

We will be prompted to enter and verify the new password. At this point, the user account will be unlocked, and they can login. If we want to do this all in a single step, that command would look like this:

$sudo useradd -m test -p PASSWORD

Where PASSWORD is the password, we want to use for the user test. Once the user logs in, they can change their password by using the passwd command, entering their current password, and then entering/verifying their new password.

Deleting User Account

For deleting user account, we will use userdel command.

The basic syntax of the command is: $userdel [option] username

When invoked, the command reads the content of the /etc/login.defs file. Properties defined in this file override the default behaviour of userdel. If USERGROUPS_ENAB is set to yes in this file, userdel deletes the group with the same name as the user, only if no other user is a member of this group. We will discuss about user group in next section. The userdel command removes the user entries from the /etc/passwd and /etc/shadow files. In most Linux distributions, when removing a user account with userdel, the user home and mail spool directories are not removed. We will use the -r (--remove) option to force userdel to remove the user's home directory and mail spool. The command will look like $userdel -r username. Suppose we want to delete recently created test account, then command will be

$sudo userdel -r test

The user account named "test" creation with password "PASSWORD" and thereafter deleting it are shown here.

```
ubuntu@ubuntu:/home$ ls
ubuntu
ubuntu@ubuntu:/home$ sudo useradd -m test -p PASSWORD
ubuntu@ubuntu:/home$ ls
test  ubuntu
ubuntu@ubuntu:/home$ sudo userdel -r test
userdel: test mail spool (/var/mail/test) not found
ubuntu@ubuntu:/home$ ls
ubuntu
```

Groups make it easy to associate users with similar access-control requirements, so managing users and groups is a key responsibility for sysadmins.

Groups in Linux

While administering a Linux machine that houses multiple users, there might be times when he needs to take more control over those users than the basic user tools offer. This idea comes to the fore especially when the administrator needs to manage permissions for certain users. For example, we have a directory that needs to be accessed with read/write permissions by one group of users and only read permissions for another group. So, users can be listed in different groups. Group allows us to set permission on the group level instead of setting the permission on individual level. Every Linux distribution has a graphical tool to manage groups. Groups can be managed by graphical tools, command line tools and by vi editor depending upon the user's experience. Here we will discuss basic group commands

The three commands used to manage group accounts on most of the Linux distribution are:

- Add a group: groupadd
- Modify a group: groupmod
- Delete a group: groupdel

Creating Group

Sysadmins are likely to add groups that represent specific and varied needs to Linux servers. Most of these need focus on controlling access to files, directories, or other resources. To create a group, we can use following command:

$ sudo groupadd [option] group_name

This command without option adds group_name group to the /etc/group file. The entry of this file is already discussed. The group membership is empty by default. There are a few options for groupadd, but a common one is --gid (-g for short), to specify a group ID (GID) number.

Modification of Group

Creating a group is only the first step. We might also need to modify the group by using the groupmod command. The most common group modifications consists of name changes.

Group modification options include:

--gid (-g): Change the GID

--new-name (-n): Change the group name

Therefore, to update a group name from group_name to gropu_name1, use the following command:

$ sudo groupmod --new-name group_name1group_name

Adding User to Group

By default, groups are empty. The group members can be edited with usermod or useradd command. If a group is not listed then by default, usermod command will remove the user from every group of which he is a member. Here, -a (append) option is used to prevent this from happening Suppose we want to add a user named test to group group_name1, then the command will be as follows.

$sudo usermod -a -G group_name1 test

We can remove users from group using gpasswd command itself. If we remove test from group_name1 , then command will looks like

$sudo gpasswd -d test group_name1

We can also remove a user from a group by manually editing the file '/etc/group'. The effects of this method will apply to the user upon reboot.

Deleting Group

We may need to delete a group. Deleting a group does not delete the user accounts that are members of the group. To delete the group "group_name1", command is :

$ sudo groupdel group_name1

Run the command groups to see what user-groups the user belongs to.

Permission and Ownership in Linux

Linux is a multi-user operating system. There may be multiple users

accessing the same system. But if any user could access and modify all files belonging to other users or system files, this would certainly be a security risk. This is why UNIX and thus Linux (Linux is a Unix-like system) have built-in security measures in place. This ensures that a file or directory can be accessed, modified, or executed by only desired users. Which file would be accessed by which user is decided by two factors in Linux:

- File ownership
- File permission

File ownership in Linux

Every file and directory in Linux have three kinds of owners:

User

User is the owner of the file. When we create a file, we become the owner of the file. The ownership can be changed as well.

Group

Every user is part of a certain group(s). A group consists of several users, and this is one way to manage users in a multi-user environment.

Other

'Other' can be considered as a super group with all the users on the system. Basically, anyone with access to the system belongs to this group.

File permissions in Linux

Every file and directory in Linux have the following three permissions for all the three kinds of owners:

- Permissions for files

  Read - Can view or copy file contents

  Write - Can modify file content

  Execute - Can run the file (if its executable)

- Permissions for directories

  Read - Can list all files and copy the files from directory

  Write - Can add or delete files into directory (needs execute permission as well)

Execute - Can enter the directory

We have already discussed the output of ls-l command in Unit-4, for example we have taken this output as reference

-rw-rw-r--1 ubuntu ubuntu 19 May 20 06:09 test0.txt

Here we have found that the out has following fields

- File type: Denotes the type of file. d means directory, - means regular file, l means a symbolic link.
- Permissions: This field shows the permission set on a file.
- Hard link count: Shows if the file has hard links. Default count is one.
- User: The user who owns the files.
- Group: The group that has access to this file. Only one group can be the owner of a file at a time.
- File size: Size of the file in bytes.
- Modification time: The date and time the file was last modified.
- Filename: Obviously, the name of the file or directory.

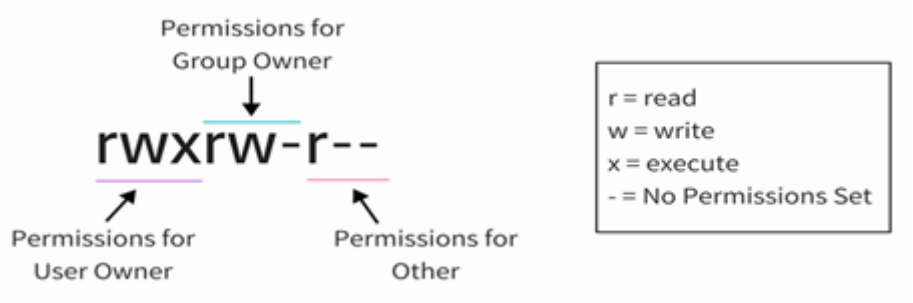
In the above line, we can see that Permission field has 9 characters:

-rw-rw-r-

Each letter denotes a particular permission:

- r: Read permission
- w: Write permission
- x: Execute permission
- -: No permission set

Permissions are always in the order of read, write and execute, i.e., rwx. And then these permissions are set for all three kinds of owners in the order of User, Group and Other.



Change file permissions in Linux

We can use chmod command for changing the permissions on a

file in Linux. There are two ways to use the chmod command: Absolute mode and Symbolic mode. In the absolute mode, permissions are represented in numeric form usually in octal form. In this system, each file permission is represented by a number.

- r (read) = 4
- w (write) = 2
- x (execute) = 1
- - (no permission) = 0

So, rw- permission is represented as 6 i.e. 4 for read permission and 2 for write permission. Similarly, rwx permission will be 7 (4+2+1) in absolute mode. So, if we want to assign read-write-execute (rwx) permission to user, read-write (rw-) permission to group owner and only read(r--) permission to other owner for test0.txt file then command will be like:

$sudo chmod 761 test0.txt

Symbolic mode

The problem with the absolute mode is that we should always provide three numbers for all the three owners even if we want to change the permission set for just one owner. This is where we can use the symbolic mode with chmod command. In symbolic mode, owners are denoted with the following symbols:

- u = user owner
- g = group owner
- o = other
- a = all (user + group + other)

The symbolic mode uses mathematical operators to perform the permission changes:

- + for adding permissions
- - for removing permissions
- = for overriding existing permissions with new value

In our previous example, if we want to perform it with symbolic mode, command will be look like:

$chmod u+rwx, g+rw, o+r test0.txt

To only add execute permission for other owner, we can use chmod command like this: $chmod o+x test0.txt

Change file ownership in Linux

To change the ownership of a file, we can use the command chown i.e. change owner. We can change the user owner of a file using chown command like:

$chown <new_user_name> <filename>

If we want to change the user as well as group, we can use chown command like this:

$chown <new_user_name>:<new_user_group> <filename>

If we just want to change the group, we can either use chown command in this manner:

$chown :<new_user_group> <filename>

or we can use chgrp (change group) command specifically used for changing group owner of a file or directory.

chgrp <new_user_group> <filename>

In our example so far, if we want to change the user owner and group to root, the chown command will be like this:

$sudo chown root:root test0.txt

The passwd Command

The passwd command in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.

Syntax:

passwd [options] [username]

```
ubuntu@ubuntu:~$ passwd ubuntu
New password:
Retype new password:
passwd: password updated successfully
```

Processing in passwd command:

- **Verify current user password:** Once the user enters passwd command, it prompts for current user password, which is verified against

the password stored in /etc/shadow file user. The root user can bypass this step and can directly change the password.

- **Verify password aging information:** In Linux, a user password can be set to expire after a given period of time. Also, a user can be prohibited to change his/her password for a period. This password aging information (and the password itself) is stored in a file /etc/shadow.

Change the password: After authentication, the user is prompted to enter the new password and verify it by retyping the password.

passwd options:

- **-d, -delete:** This option deletes the user password and makes the account password-less.
- **-e, -expire:** This option immediately expires the account password and forces the user to change password on their next login.
- **-h, -help:** Display help related information to the passwd command.
- **-i, -inactive INACTIVE_DAYS:** This option is followed by an integer, INACTIVE_DAYS, which is the number of days after the password expires that the account will be deactivated.
- **-k, -keep-tokens:** This option is used when we want to change the password if it is expired. It keeps the authentication tokens for the authentication if the password is not yet expired, even if it is requested to change it. Note that if the expiry period for a user is set to 99999, then this option will not keep tokens and the password will be changed.
- **-l, -lock:** Lock the password of user. This appends the encrypted password of the user with a character '!', and thus making it unable to match with any of input password combinations. This does not disable the account but prevents the user from logging in using a password like:

  $sudo passwd -l ubuntu Locking password for user ubuntu
- **-u , -unlock :** Unlock the password of the user, e.g.$sudo passwd -u ubuntu
- **-n, -mindays MIN_DAYS:** Change the minimum number of days between password changes to MIN_DAYS so that the user can't change the password for MIN_DAYS.

- **-q, -quiet:** This option is used for quiet mode. While using this option to change a password, the message "Changing password for $user.", usually gets printed before changing a password, does not get echoed.
- **-x, -maxdays MAX_DAYS:** Set the maximum number of days for which the password remains valid. After MAX_DAYS, the password will expire, and the user will be forced to change password.

  Example: Use the following flags to set minimum password lifetime (-n), maximum password lifetime (-x), warning before expiration (-w), and inactive to disabled (-i) in days for each. The order of the options doesn't matter.

  $ sudo passwd -n 1 -x 90 -w 3 -i 10 ubuntu

  The user ubuntu can not change the password for 1 day (-n 1), after 90 days (-x 90) password will expire and if the user remains inactive for 10 days (-i 10) then also password will expire.

  Root vs Sudo

  In a Linux system a root user can do anything, but normal user has no permissions. To run any command, they need to ask for permissions from the superuser. The easy and common way to grant administrative privileges to non-root users is that a user can use su command and temporarily become the root, but users must know the root's password. Sharing root password in multiuser system is dangerous. To overcome this risk, sudo command comes in trend. It allows a user to run a command as a root or as any other user after providing the user's own password for authentication. This information are defined in the /etc/sudoers file.

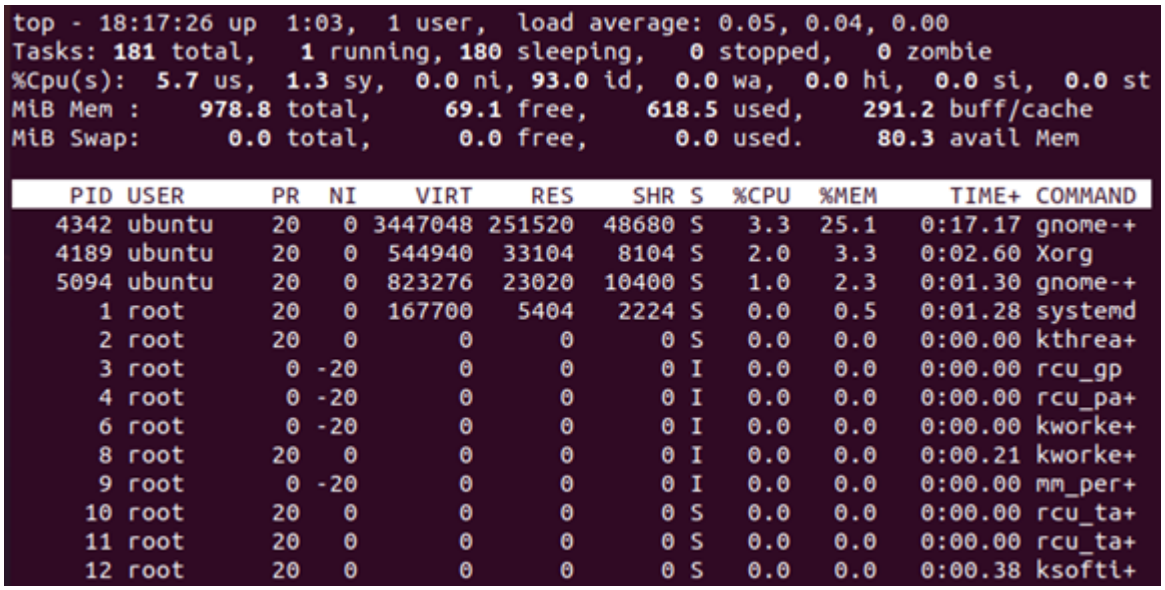
12.5 SYSTEM MONITORING

Some system monitoring tools available with Linux distribution which are used to check memory usage, disk space usage and I/O activity are described in the following.

TOP

TOP is part of every Linux distribution. Following screenshot shows

the result of top command. As we can see in the following screenshot running tasks, their CPU, memory and swap usage. We can do different specific task along with this output according to our need. It has a large number of options/switches which system administration can use easily for different purposes.



Hot keys /Switches associated with top command

Hot Key	Usage
t	Displays summary information off and on.
m	Displays memory information off and on.
A	Sorts the display by top consumers of various system resources. Useful for quick identification of performance-hungry tasks on a system.
f	Enters an interactive configuration screen for top. Helpful for setting up top for a specific task.
o	Enables you to interactively select the ordering within top.
r	Issues renice command.
k	Issues kill command.
z	Turn on or off color/mono

Free

Free also comes pre-installed with Linux distributions to check the memory usage. It shows us the buffers and cached memory as well. There are several formats of memory size like KB, MB and GB. We need to use 'm' and 'g' parameters with the free command.

Iostat

Iostat command reports us the CPU and disk I/O stats. Reads and writes are shown as block read and block writes. We can get the idle percentage of the CPU to check how much time it has not done any heavy task. Following is the output of iostat command. As we can see in the following screenshot that system is 97.7 percent idle while sda represents the hard disk.

```
ubuntu@ubuntu:~$ iostat
Linux 5.11.0-27-generic (ubuntu)        05/22/2022      _x86_64_        (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.59    0.00    1.01    0.03    0.00   97.37

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_
wrtn    kB_dscd
loop0             16.74       299.39         0.00         0.00    2318321
   0          0
loop1              0.05         0.81         0.00         0.00       6237
   0          0
loop2              0.02         0.43         0.00         0.00       3343
   0          0
loop3              0.03         0.30         0.00         0.00       2309
   0          0
loop4              0.02         0.30         0.00         0.00       2318
   0          0
loop5              0.25         8.62         0.00         0.00      66780
   0          0
loop6              0.00         0.00         0.00         0.00         14
   0          0
sda                0.11         2.70         0.00         0.00      20879
   0          0
scd0              11.57       745.90         0.00         0.00    5775834
   0          0
```

lsof: The job of lsof command is to "list open files" in the system. An open file does not mean pdf or a text file, it includes disk files or pipes used by processes in the background.

ps - Displays the Linux processes, ps command will report a snapshot of the current processes. To select all processes, use the -A or -e option.

12.6 GETTING SYSTEM INFORMATION WITH UNAME

The uname is a command line tool used to determine the processor architecture, the system hostname and the version of the kernel running

on the system. Syntax of this command is uname -options

Different options available with uname include:

-a, --all: print all information, in the following order, except omit -p and -i if unknown

-s, --kernel-name: print the kernel name

-n, --nodename: print the network node hostname

-r, --kernel-release: print the kernel release

-v, --kernel-version: print the kernel version

-m, --machine: print the machine hardware name

-p, --processor: print the processor type (non-portable)

-i, --hardware-platform: print the hardware platform (non-portable)

-o, --operating-system: print the operating system

--version: output version information and exit

When we run uname command without any option, it just prints the core operating system name. For example, here the core operating system name is Linux

```
ubuntu@ubuntu:~$ uname
Linux
ubuntu@ubuntu:~$
```

If we use -a option to display all the information in this order.

- Kernel name
- Nodename/hostname
- Kernel release
- Kernel version
- Machine
- Processor
- Hardware platform
- Operating System

For example, after using uname -a command the following output has been shown:

```
ubuntu@ubuntu:~$ uname -a
Linux ubuntu 5.11.0-27-generic #29~20.04.1-Ubuntu SMP Wed Aug 11 15:58:17 UTC 2
021 x86_64 x86_64 x86_64 GNU/Linux
```

Here

- Kernel name: Linux
- Nodename/hostname: ubuntu
- Kernel release: 5.11.0-27-generic
- Kernel version: #29~20.04.1-Ubuntu SMP Wed Aug 11 15:58:17 UTC 2021
- Machine: x86_64
- Processor: x86_64
- Hardware platform: x86_64
- Operating System: GNU/Linux

12.7 BACKUP AND RESTORE FILES

Unix and Linux backup and restore can be done using backup commands tar, cpio ufsdump, dump and restore. Though these commands may be sufficient for small setups in order to take an enterprise backup, we need to go for custom backup solutions provided by different software solutions.

Backup Restore using tar command

The tar (tape archive) command is used for single or multiple files backup and restore on/from a tape or file. The tar cannot backup special character & block device files, showing as 0 byte files with first letter of permissions as b or c for block or character and it works only on mounted file system. It cannot access the files on unmounted file system.

Backing up all files in a directory including subdirectories to a tape device (/dev/rmt/0) or a file.

$tar cvf /dev/rmt/0 *

Where Options cvf indicates create, verbose and file or archive device respectively; * refers to all files and directories. Together the command means creating a tar file on /dev/rmt/0 from all file and directories in the current directory.

$tar cvf /home/backup *

Create a tar called backup in home directory, from all file and

directories in the current directory.

Viewing a tar backup on a tape or file:

t option is used to see the table of content in a tar file. So

$tar tvf /dev/rmt/0 ## view files backed up on a tape device.

$tar tvf /home/backup ## view files backed up inside the backup

Extracting tar backup from the tape:

x option is used to extract the files from tar file.

$tar xvf /dev/rmt/0 ##extract/restore files into current directory.

$tar xvf /home/backup ## extract/restore files into current directory

Some useful tar options are-

-p, -preserve-permissions : extract & keep the file permissions (default for superuser)

-z, -gzip : Filter the archive through gzip

-j, -bzip2 : Filter the archive through bzip2

-u, -update : Only append files newer than copy in archive

-r, -append : Append files to the end of an archive

-delete : Delete from the archive (not on mag tapes!)

-d, -diff, -compare : Find differences between archive and file system

-test-label : Test the archive volume label and exit

-C, -directory=DIR Change to directory DIR

Backup/restore using cpio command

The cpio command is used for single or multiple files backup. It can backup special character & block device files which tar cannot. Like tar, it also works only on mounted file system. The cpio command needs a list of files to be backed up. It preserves hard links and time stamps of the files.

Using cpio command to backup all the files in current directory to tape, we can use:

#find . -depth -print l cpio -ovcB > /dev/rmt/0

Where cpio expects a list of files and find command provides the list, cpio has to put these files on some destination and '>' sign redirects these files to tape. This can be a file as well.

Viewing cpio files on a tape:

cpio -ivtB < /dev/rmt/0 ## Options i -> input ; v->verbose; t-table of content; B-> set I/O block size to 5120 bytes

Restoring a cpio backup:

cpio -ivcB < /dev/rmt/0 ## Options i -> input ; v->verbose; t-table of content; B-> set I/O block size to 5120 bytes

Dump and Restore:

The dump command in Linux is used for backup the filesystem to some storage device. It backs up the complete file system and not the individual files. But it is not come with a preinstall. We have to install it first to use its utilities. It works only with ext2/ext3 filesystem and not with others like FAT and ReiserFS. It allows user to take incremental backup like weekly basis.

Syntax for dump command is: **dump -[options]**

-level # : The dump level which is an integer ranging from 0-9. If the user has asked to perform a full backup or a backup of only those new files added after the last dump of a lower level.

- f file : This specifies the file where the backup will be written to. The file can be a tape drive, a floppy disk, ordinary file or standard output.

-u : This records and updates the backup in /etc/dumpdates file.

sudo dump 0uf /dev/qft0 /dev/sda6

-B records : It displays the number of dump records per volumes. In other words, it shows the amount of data that can fit in the tape. This is not always required as DUMP can detect the end of media or storage. This takes a numerical value and is used in combination with -b as mentioned below.

-b blocksize : The block size specifies the number of kilobytes per dump record. The default value of block size is 10. Notice here how in the 4th line of DUMP the block size is 20 instead of 10.

-W : Lists the file systems that need to be backed up as given below:

dump -W

Restore Command

restore command in Linux system is used for restoring files from a backup created using dump. The restore command performs the exact inverse function of dump. A full backup of a file system is being restored and subsequent incremental backups layered is being kept on top of it. Single files and directory subtrees can easily be restored from full or partial backups. Restore simply works across a network. Other arguments which need to be passed to the command are file or directory names specifying the files that need to be restored.

Options:

-a: In -i o -x mode, restore will ask the user for the volume number on which the files need to be extracted. The -a option simply disables this behaviour and reads all the volumes starting with 1.

-c: Normally, restore will try to determine dynamically whether the dump was made out of an old or from a new format file system. The -c flag simply disables this check, and it only allows reading a dump with the method of the old format.

-d: The -d (debug) flag causes the restore to print all the debug information.

-f: This option reads the backup from the file that can be a special device file like /devfffrrr5/st0, /dev/sda1 (a disk drive) etc.

-h: Extract the actual directory, rather than the files which is being referencing. This prevents the hierarchical restoration of complete subtrees from the dump.

-C: This flag allows comparison of files from a dump. Restore command reads the backup and compares it with the contents of files present on the disk.

-i: This flag allows the interactive restoration of files from a dump. After reading the directory information from the dump, restore command provides a shell-like interface which simply allows the user to move around the directory tree selecting files to be extracted.

-P file: Restore creates a new Quickly File Access file from an existing dump file without any kind of restoring its contents.

-R: Restore requests a particular tape of the multi-volume set to restart a full restore This is proven useful if the restore has been interrupted.

-r: Restore a file system. The target file system needs to be made pristine with mke2fs(8), mounted, and the user cd'd into a pristine file system before starting the restoration of the initial level 0 backup.

-t: The names of the specified files are being listed if they occur on the backup. If no file argument is already given, the root directory is listed by default.

-x: The names of the files are read from the given media. If a named file matches a particular directory whose contents are on the backup and the -h flag is not specified, the directory is then recursively extracted.

-k: Uses the Kerberos authentication when contacting the remote tape server (Only available if this option was already enabled when restore was compiled).

-l: When we are doing remote restores, then assume the remote file is a regular file (instead of a tape device). If we are supposed to be restoring a remote compressed file, then we should need to specify this option or restore will fail to access it correctly.

-m: Extracted by the inode numbers rather than by file name. This is proven useful if only a few files are being extracted, and one wants to avoid the regenerating to be completed pathname to the file.

-N: The -N flag causes the restore to perform a full execution as requested by one of the -i, -R, -r, t or x command without actually writing any file on the disk.

-o: The -o flag causes the restore to automatically restore the current directory permissions without asking for the operator whether he needs to do so in one of -i or -x modes.

-u: Whenever certain types of files are created, restore may generate a warning diagnostic if they already exist in the target directory. To prevent this, the -u (unlink) flag can cause restore to remove the old entries before attempting to create new ones.

-v: Normally restore does this work silently under the hood. The -v (verbose) flag causes it to type the name each of file it simply treats preceded

by its file type.

-V: Enables reading a multi-volume non-tape medium like CDROMs.

-y: Do not ask for the user whether to abort the restore in the event of an error. It always tries to skip over the bad block and then continues.

Dumping and Restoring

Dump can do full back up or incremental backup. Numerical argument as 0 means full backup and subsequent number represents the incremental backup corresponding to full back.

Syntax : dump - < level number > -f < source filesystem device >

sudo dump -0u -f /dev/st0 /dev/sda9 //this command will take full backup(as shown by -0)

of /dev/sda9

to restore the same file, command will be:

sudo restore -f /dev/st0

Important Files related to backup

/dev/st0: default tape unit to dump to

/etc/dumpdates: dump date records

/etc/fstab: dump table: file systems

/etc/mtab: dump table: mounted file systems

/etc/group: to find group operator

12.8 KUDZU

When we add or remove hardware from our Red Hat linux systems and reboot it, a window appears (New system may not show it) during the reboot process suggesting that hardware has either been added or removed and asking whether we want to reconfigure it. The code that detects and reconfigures hardware is called kudzu.

The kudzu script is a hardware autodetection and cut tool that runs automatically at boot time. We can start kudzu while Linux is running without rebooting also. In either case, here is what kudzu does:

- It checks the hardware connected to the computer.
- It compares the hardware it finds to the database of hardware

information stored in the /etc/sysconfig/hwconf file.

- It prompts to modify the system configuration, based on new or removed hardware that has been detected.
- If no previous database exists, kudzu attempts to determine what devices have already been configured, by looking at /etc/modprobe.conf, /etc/sysconfig/network-scripts/, and /etc/X11/xorg.conf.

Files Related to kudzu

/etc/sysconfig/hwconf : Lists the current installed hardware.

/etc/sysconfig/kudzu: Configuration for the boot-time hardware probe. Set 'SAFE' to something other than 'no' to force only safe probes.

/etc/modprobe.conf: Module configuration file.

/etc/sysconfig/network-scripts/ifcfg-*: Network interface configuration files.

The following is a list of hardware that kudzu can detect (according to the kudzu README file), followed by a version of what kudzu does to configure the device. Other devices may be detected as well (such as USB devices).

Network devices - Adds an Ethernet interface alias (eth0, eth1, etc.) if necessary and either migrates the old device configuration or creates a new one.

SCSI - Adds an alias for scsi_hostadapter.

Video card - Runs the Xconfigurator command to configure the video card.

Sound card - Runs the sndconfig command to configure and test the sound card.

Mouse - Links the new mouse device to /dev/mouse and runs the mouseconfig command to

configure and test the mouse.

Modem - Links the new modem device to /dev/modem.

Scanner - Links the new scanner device to /dev/scanner.

Keyboard - Runs the kbdconfig command to reconfigure the keyboard. Also, if you are using a serial console, it makes sure /etc/inittab and /etc/securetty are configured to be used by a serial console.

The following is a list of actions kudzu takes when a device is removed:

Network - Removes the alias for the Ethernet interface (eth0, ethl, etc.).

SCSI - Removes the alias for the SCSI host adapter (scsi_hostadapter).

Mouse - Removes the link to /dev/mouse.

Modem - Removes the link to /dev/modem.

CD-ROM - Removes the link to /dev/cdrom.

Scanner - Removes the link to /dev/scanner.

The only required problems with kudzu is its serial probe which disturbs any currently in-use devices and returns odd results if used on machines acting as serial consoles. On some older graphics cards, the DDC probe can do strange things.

In many cases, kudzu will find the right modules needed by the hardware and configure the system to load them automatically observed and identified separately. If we want to ownership kudzu without rebooting, we should first shut down the graphical desktop interface. We can do that by changing to run level 3 (init 3). Then, as root user, simply type kudzu.

12.9 X-WINDOWS

The X Window System, aka "X" is a GUI which sits on top of Linux. Unlike Microsoft Windows, the X Window System can look and operate in a large variety of different ways. It can operate very primitively or very advanced. Getting X working properly can range from simple to very complicated. Fortunately, such configuration is becoming easier and more automated in the newer distributions of Linux. Although in a majority of cases, X can be configured automatically.

There are two main methods of getting X for Linux. The first and easiest method, is to use ``Xconfigurator'' utility. The utility tries to detect the system's hardware and installs the applicable X software with the appropriate configuration settings.

If we are still unsuccessful after trying out various settings with Xconfigurator, we may use the "xf86config" utility. Although certainly not as user-friendly or attractive as Xconfigurator is, but it gives us finer control over the configuration process.

Finally, if it is still not working, we may go for editing the "/etc/X11/XF86Config" file by hand and tweaking various settings. If this is the case, we need help from experts which may be found in Linux communities.

After getting X working properly, it may be the lack of rich colours. This is because X uses a default 8-bit per pixel (``bpp'') colour depth. We can use higher colour depths, if video hardware will support them.

The various colour depths are listed in "/etc/X11/XF86Config" file, and look like this:

```
Subsection "Display"
    Depth       24
    Modes       "800x600" "1024x768"
    ViewPort    0 0
    Virtual     1024 768
EndSubsection
```

The above section shows the possible resolutions which are available when using the 24-bit colour depth (800x600 and 1024x768, as listed in the "Modes" line); these resolutions can be switched between "on-the-fly" using the <Alt><+> and <Alt><-> keys.

We can test each colour depth manually by typing, "startx -- -bpp 24" (for the 24-bit depth) and make sure X is working properly for the colour depth we wish to use.

If we are able to successfully use a higher colour depth and wish to use it as the default, we will need to create a "/etc/X11/xinit/xserverrc" file as follows:

```
exec X :0 -bpp 24
```

The above change will allow X to use 24 bits per pixel (if we have problems with this, we can try 16 or 32 instead of 24).

Assuming we have configured X properly, starting it is a simple matter of typing "startx" as any user. The X GUI will start, and after we

have finished our session and quit X, we will be returned to the regular Linux console.

We may also use the X Desktop Manager ("xdm") to start up the X Window System automatically at system boot time. This allows Linux system to always run under X (although we can switch from the GUI to the regular consoles with <Ctrl>-<Alt>-<F1>, and then back again to the GUI with <Alt>-<F7> as needed). This is a nice way of providing an attractive and friendly environment for users, and avoid having to type "startx" all the time.

To enable xdm, simply edit the "/etc/inittab" file and change the line that reads "id:3:initdefault:" to the following:

id:5:initdefault:

The above change will switch Linux to run level 5 upon system boot up; this run level, by default, will start xdm. We may also wish to check our "/etc/inittab" file, probably near the bottom, to ensure whether the following line is present:

x:5:respawn:/usr/bin/X11/xdm -nodaemon

If we have enabled xdm and wish to use a higher "bpp" value, then the default of 8 (and video card and monitor will support it), we will need to modify the "/etc/X11/xdm/Xservers" file as follows:

:0 local /usr/X11R6/bin/X -bpp 24

The above change will start the xdm at 24 bits per pixel.

We may also wish to edit the "/etc/X11/xdm/Xsetup_0" file and with a "#" character, comment out the line that starts "xbanner" as shown:

#/usr/X11R6/bin/xbanner

This will prevent the default xdm banner screen from displaying for a fragmented second between KDE sessions.

Choosing a Window Manager for X:

Now, we should decide on a window manager. The X Window System is simply the environment which allows graphics to be displayed on our system's hardware; the window manager is responsible for how X looks and how it interacts with us and our applications. The Red Hat distribution of Linux contains several window managers, including fvwm,

olvm, twm, AfterStep, and others. The default one we see when starting up X for the first time is fvwm95, a Win95-like environment.

12.10 GNOME GRAPHICAL INTERFACES

The GNU Network Object Model Environment (GNOME) is a windowing environment that enhances X window environment. It is fully featured containing a large selection of applications.

Generally, GNOME is included with the distribution. Otherwise, we will need to download the latest RPM distribution of the package. After we have all the necessary files, the GNOME package can be installed with a simple command, typed as "root":

#rpm -Uvh gtk*.rpm *.rpm

(The above command ensures the GTK libraries are installed first, to avoid dependency errors from occurring).

GNOME is actually not a Window manager, but provides added functionality to it. Therefore, once we have installed GNOME, we should decide which window manager to use, and create a ".xinitrc" file in our directory which loads the appropriate window manager and starts GNOME. The file should look something like this:

afterstep &

exec gnome-session

The above file will load AfterStep for the window manager, and then run GNOME on top of it.

More information on the GNU Network Object Model Environment can be found on the GNOME web page at http://www.gnome.org/. Don't forget to check out the screen shots, located at http://www.gnome.org/screenshots/.

12.11 KDE INSTALLATION AND CONFIGURATION

The K Desktop Package (KDE) is another popular window manager that is somewhat more mature. However, it seems to require a bit more memory resources than GNOME.

The first step for installing KDE is to download the latest RPM distribution of the package. To do so, locate an FTP mirror at http://www.kde.org/mirrors.html. Try to choose a mirror that is close to our geographic location, but make sure whichever one we choose is updated often.

When we have found a suitable mirror, download all the RPM files which are applicable to Linux version.

Once we have all the necessary files, the KDE package can be installed with the following simple commands, typed as "root" (make sure you are in the directory where all your KDE rpm files are):

#rpm -Uvh qt*.rpm

#install-kde-1.1-base

The above commands will install the Qt libraries first, and then install the KDE base package. Once this is done, we should log off and log back in so that our path environment is set appropriately. Then type:

#install-kde-1.1-apps

The above command will install the applications programs.

This installation procedure is discussed in more detail in the file "readme-redhat-rpms.txt" that should have been included with the KDE files which we downloaded.

If all goes well, and KDE has been installed without any error messages, we may have to configure KDE to be the default window manager for any of our users (the one they will see immediately after typing "startx"), by typing the following, again as "root":

/opt/kde/bin/usekde userid

More information on the K Desktop Environment can be found on the KDE web page at http://www.kde.org/. Don't forget to check out the screen shots, located at http://www.kde.org/kde2shots.html.



CHECK YOUR PROGRESS

Q 1: Which command is used to add a new user?

Q 2: Which command is used to delete an existing user account?

Q 3: Which command is used to create a group?

Q 4: Which command is used to modify a group?

Q 5: Which command is used to delete an existing group?

Q 6: What is the use of top command?

Q 7: What is the use of free command?

Q8: What is the use of tar command?



12.12 LET US SUM UP

- In system administration, we have to perform tasks like managing user accounts, granting access, checking system performance, and taking corrective measures. Absolute path names are full path name to the file whereas relative pathnames are relative to the working directory.
- Linux login process is not a single process, it has different sub processes along with parent process together.
- Linux is a multiuser system, so in same system we are able generate different users and different groups and accordingly we can give authorization for file access.
- We have lots of system monitoring tools to check system health in Linux.
- There is no inbuilt backup system for Linux, but we can use dump and restore package for it.
- The X Window System, aka "X" (commonly and incorrectly known by many as "X-Windows") is a GUI which sits on top of Linux. GNOME and KDE sits top of X windows and make it more easier to use.



12.13 FURTHER READINGS

1) Linux Administration Cookbook By Adam K. Dean

2) Linux Administration By Jason Cannon



12.14 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: adduser

Ans to Q No 2: userdel

Ans to Q No 3: groupadd

Ans to Q No 4: groupmod

Ans to Q No 5: groupdel

Ans to Q No 6: Top command is used to view the running tasks, their CPU, memory and swap usage.

Ans to Q No 7: Free command is used to check the memory usage along with buffers and cached memory.

Ans to Q No 8: tar command is used for single or multiple files backup and restore on/from a tape or file.



12.15 MODEL QUESTIONS

Q 1: Explain login process in Linux in details.

Q 2: Create a user account with password in your system add it to a specific group and modify it to only active for 3days from creation.

Q 3: Explain different system monitoring tools and their uses.

Q 4: Explain backup and restore process of Linux system

Q 5: What is kudzu? Why it is useful?

Q 6: Write short note on X-Windows, GNOME, KDE.

*** ***** ***

UNIT 13: NETWORKING IN LINUX

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 Installation and Configuration of a Simple LAN

13.4 Installation and Configuration of Proxy Server (Squid)

13.5 Installation and Configuration of Webserver Apache

13.6 Installation and Configuration of a SSH Server and Client

13.7 Installation and Configuration of File Server (Samba)

13.8 Rules Governing IP address Classes and Netmasks

13.9 The Ping Command

13.10 The netstat Command

13.11 The Traceroute Command

13.12 The telnet Command

13.13 The /etc/services File

13.14 Port

13.15 Let Us Sum Up

13.16 Further Readings

13.17 Answers to Check Your Progress

13.18 Model Questions

13.1 LEARNING OBJECTIVES

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

- learn to install and configure various Linux services
- understand Network services rules and IP address
- use network commands to configure and trouble shoot
- understand the concept of port

13.2 INTRODUCTION

In the previous unit, we have learnt the login process of Linux. We discussed how to create, modify and remove users and groups. The system monitoring tools and how to navigate X windows, GNOME and KDE are

also discussed. In this unit, we will discuss the installation and configuration of various Linux services like proxy server, Apache web server etc.

13.3 INSTALLATION AND CONFIGURATION OF A SIMPLE LAN

Linux has very powerful and robust networking capabilities. Linux's networking setup is open to review and completely configurable. In Linux everything is transparent to user and no parameter is forced upon. The basic networking principles are same both in Windows and Linux. This chapter starts with an overview of networking, and then looks in more detail at Linux networking on a Local Area Network (LAN).

By default Linux distro install network capabilities so we don't need to install anything for simple LAN .

To configure a IPv4 network we need three fundamental pieces of numerical, for IPv6 we need some more information's but here we will only discuss about IPv4. Those three fundamental numerical pieces are:

IP Address: It is an unique number defining the access point to the network interface. It has the form: xxx.xxx.xxx.xxx, where "xxx" are three, or fewer, numbers between 0 and 255.This address can be manually assigned or can be automatically assigned by DHCP.

Gateway: It is the unique number assigned to the network interface through which it communicate with other networks. It has the general xxx.xxx.xxx.xxx format and takes a form that also works with the other two numbers.

Netmask: The non-unique number that defines the network itself. This number can be automatically generated but is sometimes requested by the method use to configure the interface. It also has the format xxx.xxx.xxx.xxx.

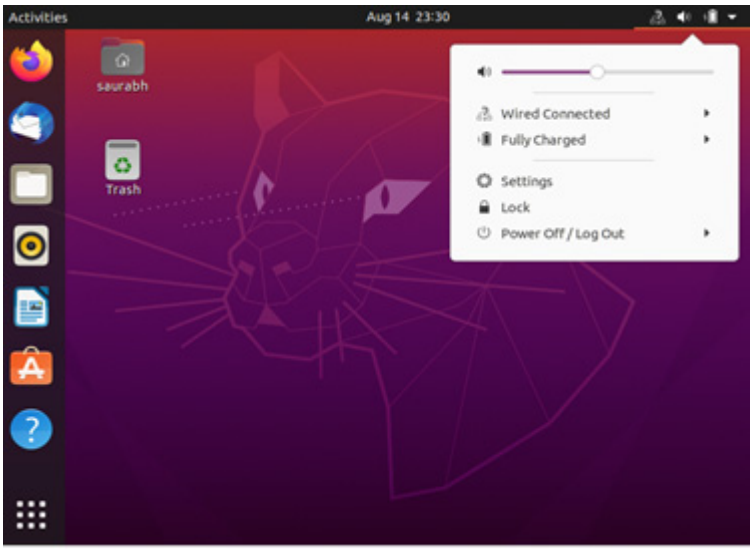
At installation time

Some of the installer ask users to provide network configuration during installation time itself , we can skip this part if not , then installer prompts with "Network & Host Name" , The installer has automatically

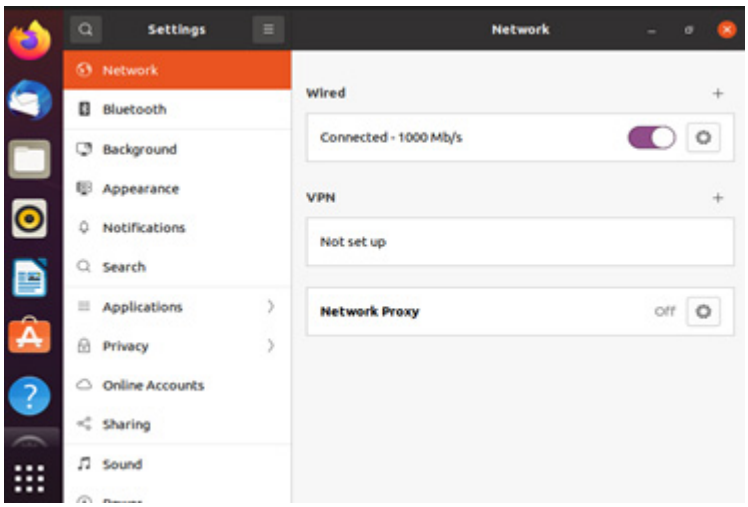
identified the one ethernet interface available and selected it for configuration. The text next to the selected interface is the system name of the interface, connection state but hardware address and speed are found from the system.

From GUI of the System

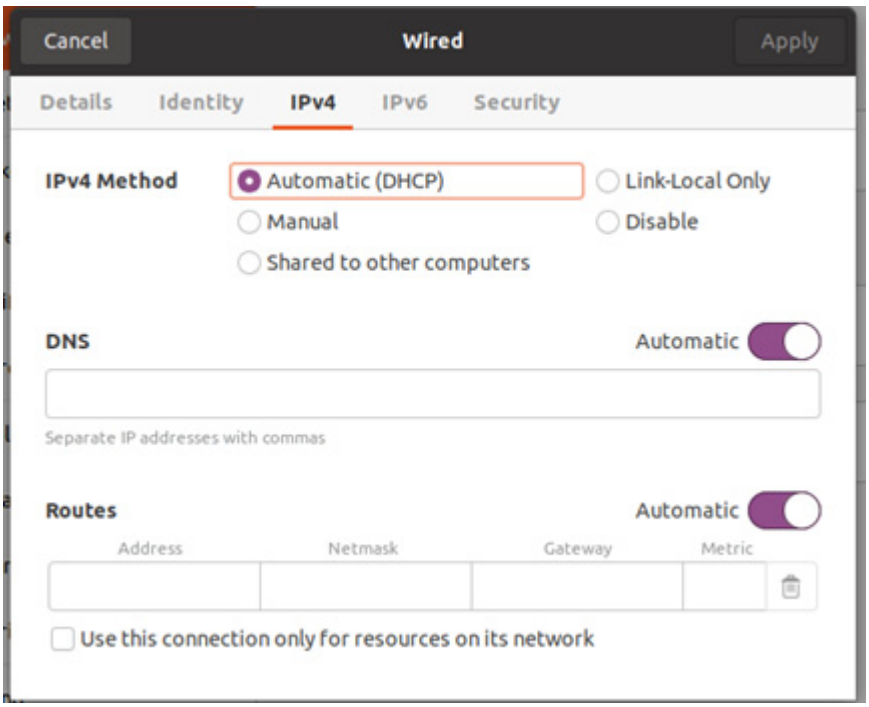
We can open the settings and go to network settings.

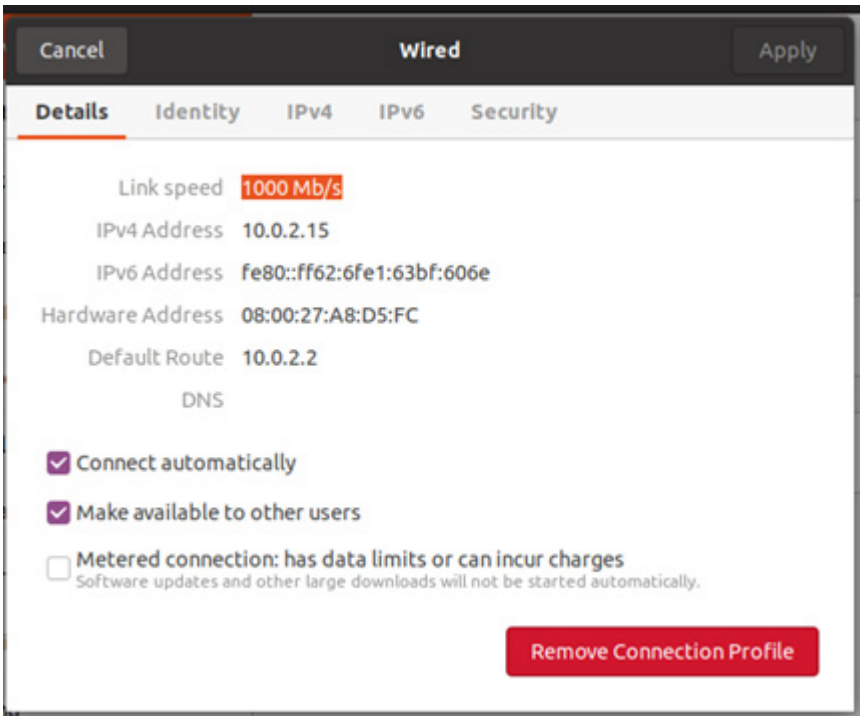


If we click on the setting it will open network settings it will show you Wired connection. There's the ON/OFF button and in this case if it's in the ON position. Choosing the gear button displays the network numbers that have been assigned (still DHCP) to the network interface.



If we click on IPv4, you can see familiar numbers we already discussed were given to your network interface by DHCP.





If we want to choose our network addresses manually, we can apply manual settings instead of DHCP

By Command Line

Suppose we are at totally new environment, and we need to setup an ethernet. While dealing with Live CD distros we can type these

commands for network setup in command line.

Setting up the hostname and DNS stuff

```
# Setup hostname
hostname mycomputerhostname
#  DNS setting
echo "nameserver 1.2.3.4" > /etc/resolv.conf
# Adding 1.2.3.4 as nameserver
```

When dealing with a live CD distro we don't care about /etc/resolv.conf file but if we are running from our own HD then taking a backup will be a good idea

Now let's flush the eth card and setup the new ip address (let's assume we've a /dev/eth0 card to setup). Here a little bit about naming convention for network interfaces. It uses ethX. The "eth " for ethernet and "X" a single digit. No some new convention uses "enoX"

```
# Flush network address for the device /dev/eth0
ip addr flush dev eth0
# Add a new IP address to /dev/eth0 (use the form you want, that's just mine)
ip addr add 192.168.0.2/24 dev eth0
# adding 192.168.0.2, subnet mask (255.255.255.0) to my eth card
# Let's it up and running
ip link set eth0 up
# Now add a default route, 192.168.0.1
ip route add default via 192.168.0.1
```

The Ifconfig command

ifconfig (interface configurator) command can be used to initialize an interface, assign IP Address to interface and enable or disable interface. With this command, we can view IP Address and Hardware / MAC address assign to interface and also MTU (Maximum transmission unit) size.

```
saurabh@saurabh-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::ff62:6fe1:63bf:606e  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a8:d5:fc  txqueuelen 1000  (Ethernet)
        RX packets 4991  bytes 6769792 (6.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2177  bytes 168681 (168.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 17481  bytes 1250413 (1.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17481  bytes 1250413 (1.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

To check details of a specific interface we can use ifconfig with interface (enp0S3) command. with -a option will display all available interface details if it is disabled also.

```
saurabh@saurabh-VirtualBox:~$ ifconfig enp0s3
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::ff62:6fe1:63bf:606e  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:a8:d5:fc  txqueuelen 1000  (Ethernet)
        RX packets 5061  bytes 6775838 (6.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2264  bytes 175659 (175.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

To set IP Address and Gateway in Linux we can use ifconfig command but this setting will be removed in case of a system reboot.

```
saurabh@saurabh-VirtualBox:~$ ifconfig enp0s3 192.168.20.10 netmask 255.255.255
.0
```

To enable or disable a specific Interface we can use this command like this .

Enable enp0s3 : ifup enp0s3

Disable enp0s3: ifdown enp0s3

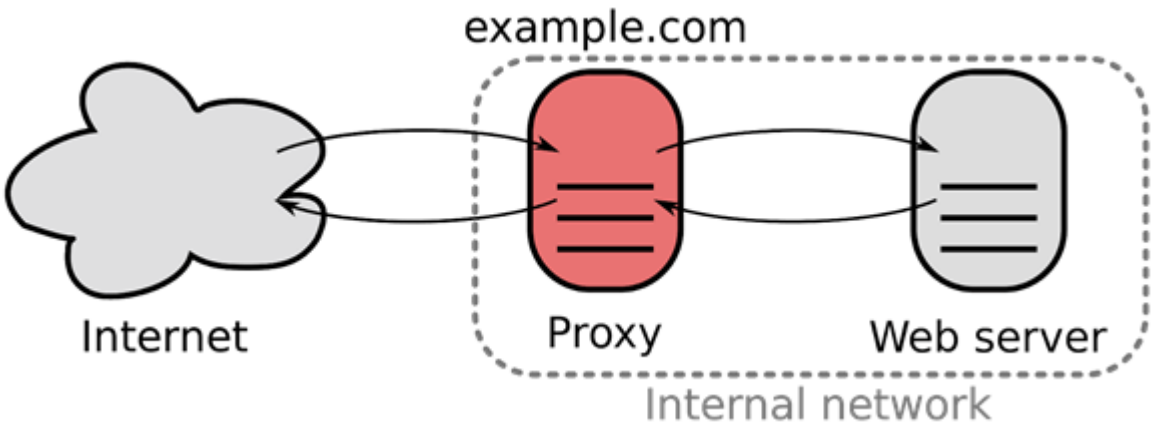
Setting MTU Size

By default MTU size is 1500. I we required size less than this we can replace XXXX with size.

ifconfig enp0s3 mtu XXX

13.4 INSTALLATION AND CONFIGURATION OF PROXY SERVER (SQUID)

Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently visited web pages. Squid has access controls and a great server accelerator. It runs on most available operating systems, including Windows and is licensed under the GNU GPL.



Squids reverse proxy is a service that sits between the Internet and the webserver (usually within a private network) that redirects inbound client requests to a server where data is stored for easier retrieval. If the caching server (proxy) does not have the cached data, it then forwards the request on to the web server where the data is stored. This type of caching allows for the collection of data and reproducing the original data values stored in a different location to provide for easier access.

A reverse proxy typically provides an additional layer of control to smooth the flow of inbound network traffic between your clients and the webserver.

Installation and configuration of: Proxy server (Squid) has been discussed in Unit 14 in details, here we have provided a basic idea about it.

Installation and configuration of: DNS server (BIND)

A DNS is a computer server that stores a IP addresses and their associated domain names. It serves to translate a domain name to its IP address, so that the computer knows which IP address to connect to for

the requested contents. For a human mind it is easier to remember a domain name as myexample.com, rather than a string of numbers, 104.20.73.209 (IP address). A DNS plays an important role in helping us to conveniently use the Internet as its translate domain name to particular IP address.

A DNS starts working as soon as user types a domain name in the address bar of a browser. It will search the Internet to find the correct IP address that is associated with the domain name. After successfully identifying the IP address, it then guides the user's browser to connect to it, which will then serve the requested website contents.

At first a DNS send a DNS query to several other DNS servers. A DNS is not just a single server responding to over billions of domain name requests, but instead it is distributed globally across a network of DNS. All the DNS servers work together to attend to the billions of domain names requests worldwide. It reduces the response for their requests.

When a website address is entered by a user in an Internet browser, a DNS query is initiated and a DNS server sends the query to several other DNS servers, each tasked with translating a different part of the domain name the user entered. There are mainly four servers which work together to translate the website address into a computer readable IP address, namely the DNS Resolver server, the root server, the Top-level Domain (TLD) server and the Authoritative Name Server. The DNS Resolver is the server who does most of the process in translating a domain name to an IP address. It receives the DNS query and in turn acts like a client to query the three other DNS servers mentioned above to translate the domain name.

It first queries the root server, and the root server responds to the query by returning the IP address of a TLD server (like .com, .net, .org etc.). A TLD server stores the information for its domains and will return the IP address of the Authoritative Name Server to the DNS Resolver. This is where the requested website is located. It then returns the actual IP address of the requested website to the DNS Resolver, which in turn response to the initial DNS query the actual IP address. Check out the illustration below to have a clear picture of the step-by-step processes of how a DNS work.

Installation and configuration of Mail server

If anyone wants to send or receive an email from its Linux system, it must have a mail sever. A mail server works according to SMTP (Simple Mail Transfer Protocol) works as well as other mail-related protocols, like Post Office Protocol (POP) and Internet Message Access Protocol (IMAP).

SMTP protocol defines how to send mail from one host to another; it is also system independent, which means the sender and receiver may have different operating systems. SMTP server can send ASCII text to another server by its default port 25.

Most Linux distribution today by default has two of the most common implementations of SMTP : sendmail and Postfix. Sendmail is a famous and free mail server, but it has a little complex design and less secure. The Postfix took mail server implementation is easy and has security . So here we will try Postfix server.

The mail service on any mail server has three components:

Mail user agent (MUA): this component that the user sees and interacts with like Thunderbird and Microsoft Outlook, these user agents are responsible for reading mail and allowing you to compose mail.

Mail transport agent (MTA): this component is responsible for getting the mail from one site to another like Sendmail and Postfix.

Mail delivery agent (MDA): this component is responsible for distributing received messages on the local machine to the appropriate user mailbox like postfix-maildrop and Procmail.

Setup Email server

Postfix is a default mail server in most of the Linux distribution. But to be sure first check is it present or not in your system. We can check that with following command

$ rpm -qa l grep postfix

```
saurabh@saurabh-VirtualBox:~$ rpm -qa | grep postfix
saurabh@saurabh-VirtualBox:~$
```

The rpm package may not be installed at your system so you may have to install rpm package before using it . If the output of this command returns nothing that means Postfix is not installed at your system.

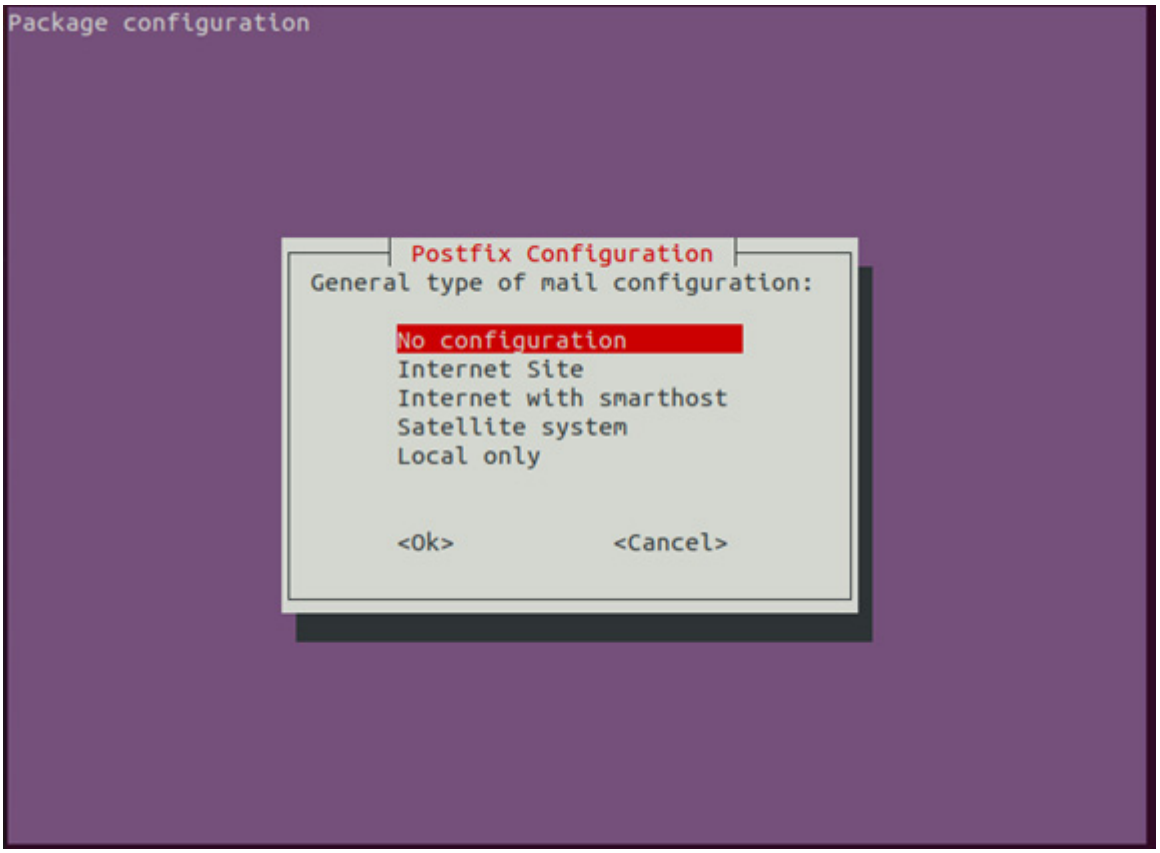
On Debian based distros like Ubuntu, we can install it with following command

$ sudo apt-get -y install postfix

```
saurabh@saurabh-VirtualBox:~$ sudo apt-get -y install postfix
[sudo] password for saurabh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

It will ask you to select the type of configuration of the Postfix mail server during the installation process.

There are four choices No configuration, Internet site, Internet with smarthost, Satellite system, and Local only, we will choose No configuration option as we are going to configure it our own



Now start the postfix service and enable it on system start-up:

```
saurabh@saurabh-VirtualBox:/$ systemctl start postfix
saurabh@saurabh-VirtualBox:/$ systemctl enable postfix
```

After installation of the Postfix mail server, we need to configure it; we can find most of its configuration files under the /etc/postfix/ directory.

You can find the main configuration for Postfix mail server in /etc/postfix/main.cf file. It is always good practice to take a backup before doing any change.

This file contains a lot of options like:

myhostname

You can use this option for specifying the hostname of the mail server. This is the Internet hostname, which Postfix will receive emails on it. The hostnames could be like mail.example.com, smtp.example.com.

You incorporate the hostname this way:

myhostname = mail.example.com

```
89 # The myhostname parameter specifies the internet hostname of this
90 # mail system. The default is to use the fully-qualified domain name
91 # from gethostname(). $myhostname is used as a default value for many
92 # other configuration parameters.
93 #
94 #myhostname = host.domain.tld
95 #myhostname = virtual.domain.tld
96 myhostname = mail.example.com
```

mydomain

This option is the mail domain that pwe will be servicing, like example.com The syntax is like this: mydomain = example.com

```
 99 # The mydomain parameter specifies the local internet domain name.
100 # The default is to use $myhostname minus the first component.
101 # $mydomain is used as a default value for many other configuration
102 # parameters.
103 #
104 #mydomain = domain.tld
105 mydomain = example.com
106
```

myorigin

All emails sent from this mail server will look as though it came from this option. We can set this to $mydomain value.

myorigin = $mydomain

we can use any option value,but we neet to put a $ like $mydomain.

```
107 # SENDING MAIL
108 #
109 # The myorigin parameter specifies the domain that locally-posted
110 # mail appears to come from. The default is to append $myhostname,
111 # which is fine for small sites.  If you run a domain with multiple
112 # machines, you should (1) change this to $mydomain and (2) set up
113 # a domain-wide alias database that aliases each user to
114 # user@that.users.mailhost.
115 #
116 # For the sake of consistency between sender and recipient addresses,
117 # myorigin also specifies the default domain name that is appended
118 # to recipient addresses that have no @domain part.
119 #
120 # Debian GNU/Linux specific:  Specifying a file name will cause the
121 # first line of that file to be used as the name.  The Debian default
122 # is /etc/mailname.
123 #
124 #myorigin = /etc/mailname
125 #myorigin = $myhostname
126 myorigin = $mydomain
```

mydestination

This option lists the domains that the Postfix server uses for incoming emails.

It can take values like this:

mydestination = $myhostname, localhost.$mydomain, $mydomain, mail.$mydomain, www.$mydomain

```
175 # The local machine is always the final destination for mail addressed
176 # to user@[the.net.work.address] of an interface that the mail system
177 # receives mail on (see the inet_interfaces parameter).
178 #
179 # Specify a list of host or domain names, /file/name or type:table
180 # patterns, separated by commas and/or whitespace. A /file/name
181 # pattern is replaced by its contents; a type:table is matched when
182 # a name matches a lookup key (the right-hand side is ignored).
183 # Continue long lines by starting the next line with whitespace.
184 #
185 # See also below, section "REJECTING MAIL FOR UNKNOWN LOCAL USERS".
186 #
187 #mydestination = $myhostname, localhost.$mydomain, localhost
188 #mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
189 mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain,
190         mail.$mydomain, www.$mydomain, ftp.$mydomain
```

mail_spool_directory

There are two modes of delivery that Postfix mail server can use:

- Directly to a user's mailbox.
- To a central spool directory, this way, the mail will be in /var/spool/mail with a file for each user.

mail_spool_directory = /var/spool/mail

```
445 # The mail_spool_directory parameter specifies the directory where
446 # UNIX-style mailboxes are kept. The default setting depends on the
447 # system type.
448 #
449 #mail_spool_directory = /var/mail
450 mail_spool_directory = /var/spool/mail
451
```

mynetworks

This will let you arrange which servers can relay through your Postfix server.It should only take local addresses like local mail scripts on your server.If this isn't the case, then spammers can piggyback on your Linux mail server. That means your lovely shiny server will be doing the heavy lifting for some bad guys and it will also end up getting banned.

Here's the syntax for this option:

mynetworks = 127.0.0.0/8, 192.168.1.0/24

smtpd_banner

This one determines what message is sent after the client connects successfully. Consider changing the banner so it doesn't give away any potentially compromising information about your server.

inet_protocols

This option designates which IP protocol version is used for server connections.

inet_protocols = ipv4

When we change any of files used to configure Linux mail server for Postfix, you must reload the service, with this directive:

$ systemctl reload postfix

Of course, we all get distracted and typing things in can often result in mistakes, but you can track down any misspellings that might compromise your Linux mail server using this command:

$ postfix check

Checking the Mail Queue

Things like network failure (and many other reasons) can mean that the mail queue on your Linux email server can end up getting full, but you can check the Postfix mail queue with this command:

$ mailq

If that reveals that its full then you can flush the queue using this

command:

$ postfix flush

Look at it again and you should see that your Linux email server queue is clear.

Test Linux Mail Server

Once your configuration is done you need to test your Linux mail server.

The first thing to do is use a local mail user agent such as mailx or mail which is a symlink to mailx.

Send your first test to someone on the Linux mail server and if that works then send the next one to somewhere external.

$ echo "This is the body of the message" I mailx -s "Here we have a Subject" -r "for instance <small example@mydomain.com>" -a /path/to/attachment someone@mydomain.com

Then check if your Linux email server can pick up external mail.

If you run into any snags, have a peek at the logs. The Red Hat log file can be found in /var/log/maillog and for Debian versions in /var/log/mail.log, or wherever else the rsyslogd configuration specifies.

I would suggest you review the Linux syslog server for an in-depth clarification on logs and how to set up rsyslogd.

13.5 INSTALLATION AND CONFIGURATION OF WEBSERVER APACHE

A web server is a system which serves the requests via the HTTP protocol, if you request a file from the server, and it responds with the requested file, that's why web server are not only for web. Web servers also serves devices such as printers, routers, when you open your router configuration page, there is a web server behind it.

When we type something in a browser it sends a request to the server. The server takes the requested file or page from browser and maps it to the corresponding file from the server. The server sends the file back to the browser with some information such as the MIME type, the length of

the content, and other useful information. Sometimes the requested file is a static page like HTML pages or dynamic pages like PHP, Java, Perl, or any other server-side language.

There are many Linux web server implementations available for you to use:

- Apache server
- Nginx
- Lighttpd
- Apache Tomcat

But we will use Apache server here as it is stable, flexible and secure.

We can install apache using following command :

$ apt-get -y install apache2

The Apache web server service is called apache 2 in ubuntu. Apache works on http protocol which uses port 80. so we need to open port 80. In linux firewall like iptables, we can add a rule to open port 80.

$ iptables -I INPUT 1 -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT

To start your service and enable it on boot:

$ systemctl start httpd

$ systemctl enable httpd

To check if your service is running or not, use the following command

$ systemctl status httpd

Configuring Apache web server

You can add files to Apache in the

/var/www/html

directory for top-level pages. Just remember to make sure that any files or directories placed in that directory are world-readable. The default index page is index.html.

The Apache configuration files are present in

/etc/apache2/apache2.conf

To discuss every options for conf file is beyond scope of this article, but we will discuss the most important options.

ServerRoot option

This option specifies the configuration folder for the Apache web server.. On Debian distros the ServerRoot option is /etc/apache2/.

ServerRoot /etc/apache2

Listen option

The Listen option specifies the port that the Apache web server will use to wait for incoming connections. The default value for this option is 80 for nonsecure connections and 443 for secured connections. If you have multiple IP addresses on your server, you can assign which IP should listen for connection using the Listen option.You can specify a different port other than 80, just make sure that it's not in use.You can run many HTTP servers on the same hardware every one on a unique port.When a server runs on a non-standard port such as port 8080, it will require the port number to be explicitly stated like this:

www.example.com:8080

Listen 80

ServerName option

This option specifies the hostname of the web server that appears to the visitors.

ServerName FQDN

DocumentRoot option

The DocumentRoot option defines the path that will contain your files that will be served.

The default path is /var/www/html.

DocumentRoot /var/www/html

MaxRequestWorkers option

This option sets the maximum number of concurrent connections that the server will receive.

LoadModule option

We use this option to load modules into the Apache web server.

There are a lot of Apache modules like these:

mod_cgid: This module runs CGI scripts using the Apache web server.

mod_ssl: Provides secure connections via SSL and TLS protocols.

mod_userdir: This module allows you to serve content from users' specific directories.

If you want to disable loading a specific module, you can use these commands:

$ a2enmod modulename

The a2enmod enables the module.

$ a2dismod modulename

This command to disable the module.

All these commands do is create a symlink under

/etc/apache2/mods-enabled

And if you use a2dismod, the symlink will be removed.If you enable or disable a module, you have to reload or restart the Apache web server.

LoadModule mod_cgid.so

Include option

This option allows you to include other configuration files.You can store all the configuration for different virtual domains, and Apache will include them at runtime.

Include filePath

UserDir option

This option specifies the directory that will contain the files that will be accessible via the web server. This directory is usually public_html and its location in the user's home directory.For example, if you have a user adam who wants to make his web content available via the Apache web server.

First, we make a public_html folder under his home directory.

Then set the permission for the public_html folder:

$ chmod 644 public_html

Now if we put an index.html file, it will be accessible via the browser like this:

http://YOURHOSTNAME/~adam

UserDir public_html

Alias option

This option specifies the location of the files that are outside the DocumentRoot location and need to be served by the Apache web server.Like you have files outside DocumentRoot, and you want them to be available to the visitors.

Alias URL_Path Actual_Path

ErrorLog option

This option specifies the error log file for the Apache web server.

ErrorLog /var/log/httpd/error_log

VirtualHost option

This option allows you to host multiple websites on the same server.

The idea is that the content is served based on the requested hostname.

To setup a virtual host for the host www.example.com. First, create a VirtualHost option in /etc/httpd/conf/httpd.conf file.

And specify the DocumentRoot and ServerName like this:

<VirtualHost www.example.com>
ServerAdmin admin@example.com
DocumentRoot /home/adam/public_html
ServerName www.example.com
ErrorLog /var/log/users/adam/error_log
</VirtualHost>

Keep in mind that the ServerName option must be resolvable via DNS.

These are the most used Apache options.

Virtual Host types

There are two types of virtual hosts that you can define in Apache web server:

- Name-based virtual hosts
- IP-based virtual hosts

The NameVirtualHost directive defines which addresses can be virtual hosts; the asterisk (*) means any name or address on this server.

You can write them like this:

```
NameVirtualHost *
<VirtualHost *>
 ServerName www.example.com
 DocumentRoot "/home/user1/public_html/"
</VirtualHost>
<VirtualHost *>
 ServerName www.example2.com
 DocumentRoot "/ home/user2/public_html/"
</VirtualHost>
```

If you have more than one IP address and you want to use an SSL certificate, the website must be on a dedicated IP address. You can write IP-based virtual hosts like this:

```
<VirtualHost 192.168.1.2>
 ServerName www.example.com
 DocumentRoot "/home/user1/public_html/"
</VirtualHost>
<VirtualHost 192.168.1.3>
 ServerName www.example2.com
 DocumentRoot "/ home/user2/public_html/"
</VirtualHost>
```

Apache process ownership

We know from the Linux process management that each process inherits its permissions from its parent process.This fact is true for all processes except for applications with the SETUID bit set; they inherit permissions from the owner, not the parent process. A good example is the /bin/su.If a normal user runs /bin/su program, it does not inherit the permission from adam, but it acts as a root user running it.Since Apache web server needs to bind port 80, and this needs root privileges.

After binding to port 80, Apache can run as a normal user and only read files that have permissions to read them.

User option

The User option specifies the user ID which the web server will use

to answer requests.

User www-data

Group option

The Group option specifies the group that the Apache web server will use to read files.

Group www-data

Security is very important for sites that use executable scripts such as CGI or PHP scripts.

The user you will use will have the permission to read and write the content of all sites on the server. But we want to ensure that only the members of a particular site can read their site only.

This is very important because if an attacker compromises your website, he will be able to read all files since the Apache user has permission to do that.

So how to solve this problem?

suEXEC support

A popular method is to use suEXEC. suEXEC is a program that runs with root permissions and makes CGI programs run as the user and group IDs of a specific user, not the Apache server user.

You can specify the user on each virtual host like this:

<VirtualHost www.example.com>

SuExecUserGroup adam adamGroup

</VirtualHost>

Just that simple.

Apache authentication

You may want to restrict some parts to specific visitors. It's like a password-protected directory.

In Apache, you can store the authentication information file called .htpasswd file.

You can use the htpasswd command to do that.

First, create the .htpasswd file using the htpasswd command:

$ htpasswd -c /home/adam/.htpassswd myuser

You need the -c option the first time you run htpasswd, but when you need to add more users, you shouldn't use -c because it will overwrite the file.

Then create a .htaccess file in the public_html folder and write the following:

<Location /vip>
AuthName "test"
AuthType Basic
AuthUserFile /home/adam/.htpasswd
Order deny,allow
require valid-user
</Location>

AuthName is required, you can use any string you want.

AuthType Basic says that you're using an htpasswd style user file.

AuthUserFile points to the file that contains the generated password from the htpasswd command.

The Order line indicates that Apache must deny access by default, and only allow access for users specified in the htpasswd file.The require directive means any user in the .htpasswd file is allowed.

13.6 INSTALLATION AND CONFIGURATION OF A SSH SERVER AND CLIENT

SSH (Secure Shell framework) is a framework which is powerful and efficient system for accessing. It helps to establish connection and communication between two computing devices or routers remotely. It is a secured protocol; it encrypts all connections to ensure that all data or information does not land in the wrong hands. Thus, attackers are unable to eavesdrop or hijack details in the connection. It also offers an array of authentication modes and configuration options.

Setting Up Linux SSH on Ubuntu Linux

sshd or OpenSSH Daemon or server is an open source (free) ssh server which functions as the utility program for the ssh client. Ubuntu

distro do not come with sshd set up and running. So we need to install it .

First update your system

```
# sudo apt update

# sudo apt upgrade
```

Then install it

```
# sudo apt-get install openssh-server
```

```
saurabh@saurabh-VirtualBox:/$ sudo apt-get install openssh-server
[sudo] password for saurabh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

First, we will install the server than we will try with client. Once installed, verify the status of your ssh service using the below command.

```
# sudo systemctl status ssh
```

The above command should give you a result similar to the one in the image below. Note that the "Active (running)" is highlighted to indicate that SSH installation is successful and the protocol is already running in your systems.

```
saurabh@saurabh-VirtualBox:/$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: >
     Active: active (running) since Thu 2022-08-18 12:30:27 IST; 2min 25s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 6812 (sshd)
      Tasks: 1 (limit: 1082)
     Memory: 1.0M
     CGroup: /system.slice/ssh.service
             └─6812 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Aug 18 12:30:27 saurabh-VirtualBox systemd[1]: Starting OpenBSD Secure Shell s>
Aug 18 12:30:27 saurabh-VirtualBox sshd[6812]: Server listening on 0.0.0.0 por>
Aug 18 12:30:27 saurabh-VirtualBox sshd[6812]: Server listening on :: port 22.
Aug 18 12:30:27 saurabh-VirtualBox systemd[1]: Started OpenBSD Secure Shell se>
```

If SSH does not return an active result, enable the SSH protocol in your systems by using following commands

```
# sudo systemctl enable ssh

# sudo systemctl start ssh
```

Ubuntu comes with the Uncomplicated Firewall (UFW), which manages network rules by default. An active UFW will potentially prevent

the connection of SSH to your server. You can change UFW settings to allow connection to the SSH server. You should also use this command t open port 22. These commands will help you configure UFW to allow connection;

```
# sudo ufw allow ssh
# sudo ufw enable
# sudo ufw status
```

To check ssh is working or not use following command : ssh user@localhost

You can get the user name by using command: whoami

```
saurabh@saurabh-VirtualBox:/$ ssh saurabh@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:spE1egz0U5a/EbGmqmyg40biDFddVDSE+rRvAXCOZ9o.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
saurabh@localhost's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

366 updates can be applied immediately.
259 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Setting Up SSH Client on Linux

Now we can configure the SSH client. Notably, this allows connection to the remote server from your local machine. The procedure for setting up an SSH client is as follows;

The local machine should have the latest SSH version installed. But in cases where no SSH client is already installed, you can use the below command to install the client on Ubuntu;

```
# sudo apt install openssh-client
```

Connect to the Remote System

You can establish a connection to your remote system from the local computer. To achieve this, you should know the IP address of your local command and use this command.

```
# ssh username@IP_address
```

After installing and setting up SSH on both your server and client machines, you should be able to access your server remotely once you establish a connection.

13.7 INSTALLATION AND CONFIGURATION OF FILE SERVER (SAMBA)

Samba is a free and open-source software that allows files to be shared across Windows and Linux systems. It works on the (SMB) Server Message Block Protocol which is a client-server communication protocol that is used for sharing access to files, printers, serial ports, and other resources on a network. The (CIFS) Common Internet File System Protocol is a dialect of the SMB protocol. A collection of message bundles that describes a distinct variant of a protocol is called a dialect.While using Samba, we can share files and printing services, use authentication and authorization, name resolution, and service announcements between Linux/Unix servers and Windows clients.

The basic tasks that can be performed using Samba are

- share a Linux drive with Windows machines.
- access an SMB share with Linux machines.
- share a Linux printer with Windows machines.
- share a Windows printer with Linux machines.
- set up a domain controller on a Unix/Linux server and integrate Windows clients to the Domain controller.

Installation and configuration part of Sama has been put as a Case Study for the learners.

13.8 RULES GOVERNING IP ADDRESS CLASSES AND NETMASKS

An IP (Internet Protocol) address is a numerical label assigned to the devices connected to a computer network that uses the IP for communication. IP address act as an identifier for a specific machine on a particular network. It also helps you to develop a virtual connection between a destination and a source. The IP address is also called IP number or internet address. It helps you to specify the technical format of the addressing and packets scheme. Most networks combine TCP with IP. An IP address consists of four numbers, each number contains one to three digits, with a single dot (.) separates each number or set of digits.

IP Address is divided into two parts:

- Prefix: The prefix part of IP address identifies the physical network to which the computer is attached. Prefix is also known as a network address.
- Suffix: The suffix part identifies the individual computer on the network. The suffix is also called the host address.

For example IP address 192.168.30.20 first 3 octave that is 192.168.30 is network address and last 20 part is host address

IP address works in an IP network like a postal address. For example, a postal address combines two addresses, address, or your area your house address.The address or your area is a group address of all houses that belong to a specific area. The house address is the unique address of your homes in that area. Here, your area is represented by a PIN code number.In this example, the network address comprises all hosts which belong to a specific network. The host address is the unique address of a particular host in that network.

Classful Addressing

Classful addressing is a network addressing the Internet's architecture from 1981 till Classless Inter-Domain Routing was introduced in 1993.This addressing method divides the IP address into five separate classes based on four address bits. Here, classes A, B, C offers addresses

for networks of three distinct network sizes. Class D is only used for multicast, and class E reserved exclusively for experimental purposes.

Let's see each of the network classes in detail:

Class A Network

This IP address class is used when there are a large number of hosts. In a Class A type of network, the first 8 bits (also called the first octet) identify the network, and the remaining have 24 bits for the host into that network.

An example of a Class A address is 102.168.212.226. Here, "102" helps you identify the network and 168.212.226 identify the host.

Class A addresses 127.0.0.0 to 127.255.255.255 cannot be used and is reserved for loopback and diagnostic functions.

Class B Network

In a B class IP address, the binary addresses start with 10. In this IP address, the class decimal number that can be between 128 to 191. The number 127 is reserved for loopback, which is used for internal testing on the local machine. The first 16 bits (known as two octets) help you identify the network. The other remaining 16 bits indicate the host within the network.

An example of Class B IP address is 168.212.226.204, where *168 212* identifies the network and *226.204* helps you identify the Hut network host.

Class C Network

Class C is a type of IP address that is used for the small network. In this class, three octets are used to indent the network. This IP ranges between 192 to 223.

In this type of network addressing method, the first two bits are set to be 1, and the third bit is set to 0, which makes the first 24 bits of the address them and the remaining bit as the host address. Mostly local area network used Class C IP address to connect with the network.

Example for a Class C IP address:

192.168.178.1

Class D Network

Class D addresses are only used for multicasting applications. Class

D is never used for regular networking operations. This class addresses the first three bits set to "1" and their fourth bit set to use for "0". Class D addresses are 32-bit network addresses. All the values within the range are used to identify multicast groups uniquely.

Therefore, there is no requirement to extract the host address from the IP address, so Class D does not have any subnet mask.

Example for a Class D IP address:

227.21.6.173

Class E Network

Class E IP address is defined by including the starting four network address bits as 1, which allows you two to incorporate addresses from 240.0.0.0 to 255.255.255.255. However, E class is reserved, and its usage is never defined. Therefore, many network implementations discard these addresses as undefined or illegal.

Example for a Class E IP address: 243.164.89.28

Network Mask

Mask - It is a 32-bit binary number that gives the network address in the address block when AND operation is bitwise applied on the mask and any IP address of the block.

The default mask in different classes are :

Class A - 255.0.0.0

Class B - 255.255.0.0

Class C - 255.255.255.0

Example : Given IP address 132.6.17.85 and default class B mask, find the beginning address (network address).

Solution : The default mask is 255.255.0.0, which means that the only the first 2 bytes are preserved and the other 2 bytes are set to 0. Therefore, the network address is 132.6.0.0.

Subnetting: Dividing a large block of addresses into several contiguous sub-blocks and assigning these sub-blocks to different smaller networks is called subnetting. It is a practice that is widely used when classless addressing is done.

Classless Addressing

To reduce the wastage of IP addresses in a block, we use sub-netting. What we do is that we use host id bits as net id bits of a classful IP address. We give the IP address and define the number of bits for mask along with it (usually followed by a '/' symbol), like, 192.168.1.1/28. Here, subnet mask is found by putting the given number of bits out of 32 as 1, like, in the given address, we need to put 28 out of 32 bits as 1 and the rest as 0, and so, the subnet mask would be 255.255.255.240.

Some values calculated in subnetting:

- Number of subnets : Given bits for mask - No. of bits in default mask
- Subnet address : AND result of subnet mask and the given IP address
- Broadcast address : By putting the host bits as 1 and retaining the network bits as in the IP address
- Number of hosts per subnet : 2(32 - Given bits for mask) - 2
- First Host ID : Subnet address + 1 (adding one to the binary representation of the subnet address)
- Last Host ID : Subnet address + Number of Hosts

Example : Given IP Address - 172.16.0.0/25, find the number of subnets and the number of hosts per subnet. Also, for the first subnet block, find the subnet address, first host ID, last host ID and broadcast address.

Solution : This is a class B address. So, no. of subnets = 2(25-16) = 29 = 512.

No. of hosts per subnet = 2(32-25) - 2 = 27 - 2 = 128 - 2 = 126

For the first subnet block, we have subnet address = 0.0, first host id = 0.1, last host id = 0.126 and broadcast address = 0.127

Rules for assigning Network ID:

The network ID will be assigned based on the below-given rules:

The network ID cannot start with 127 because 127 belongs to class A address and is reserved for internal loopback functions.

All bits of network ID set to 1 are reserved for use as an IP broadcast address and cannot be used.

All bits of network ID are set to 0. They are used to denote a particular host on the local network and should not be routed.

13.9 THE PING COMMAND

Ping is short for Packet Internet Groper used for checking the network connectivity among host/server and host. The ping command can take a URL or IP address as input and transfers the data packet to a specified address along with a "PING" message. Then, it will get a reply from the host/server. This time is known as "latency".

The ping command permits us to:

- Test our Internet connection.
- Check if the remote machine is active.
- Analyse when there are network problems such as high latency or dropped packages.

When we use "ping" command to a remote host, our machine starts transferring ICMP echo requests and wait for a response. An echo response for all requests has been transferred when the connection is established. The result includes the times it takes for all packets for reaching its target and return. Besides, the terminal prints the reply until interrupted which will be followed by some ping statistics

The basic syntax of the ping command contains ping followed by the hostname, a website name, or exact IP address.

Syntax: ping [options] hostname or IP address

```
saurabh@saurabh-VirtualBox:~$ ping google.com
PING google.com (142.250.183.110) 56(84) bytes of data.
64 bytes from bom12s13-in-f14.1e100.net (142.250.183.110): icmp_seq=1 ttl=116 t
ime=58.7 ms
64 bytes from bom12s13-in-f14.1e100.net (142.250.183.110): icmp_seq=4 ttl=116 t
ime=60.3 ms
64 bytes from bom12s13-in-f14.1e100.net (142.250.183.110): icmp_seq=7 ttl=116 t
ime=57.7 ms
```

This reply will goes until we manually stop it by pressing ctrl+c. The detail of the reply as follows

- **from:** The destination and its IP address. This IP address may be different for a website depending on geographical location.

- **icmp_seq=1:** The sequence number of each ICMP packet. Increases by one for every subsequent echo request.
- **ttl=52:** The Time to Live value from 1 to 255. It represents the number of networks hops a packet can take before a router discards it.
- **time=58.7 ms:** The time it took a packet to reach the destination and come back to the source. Expressed in milliseconds.

Ping has many other utilities also which can be explored with its different options like :

Option	Description
a	Use this option for a beep sound when the peer is reachable
b	Use this option to allow pinging a broadcast address
B	Use this option if you do not want to allow the ping to change the source address of the probe
c (count)	Use this option to set the number of times to send the ping request
d	Use this option to set the SO-DEBUG option on the socket being used
f	Use this option to flood the network by sending hundred or more packets per second
i	(interval) Use this option to specify an interval between successive packet transmissions. The default value of interval is 1 second
I	(interface address) Use this option to set source address to the specified interface address. This option is required when pinging IPv6 link local address. Its argument can be an IP address or name of the device.
l	(preload) Use this option to set the number of packets to send without waiting for a reply. For selecting a value more than 3, you need to be a super user.
n	Use this option to display network addresses as numbers rather than as hostnames
q	Use this option to display a quiet output. It means that only the summary is displayed at startup and finish time

T (ttl) Use this option to set the Time To Live

v Use this option for verbose output

V Use this option to display the version and exit

w (deadline) Use this option to specify a timeout, in seconds, before ping exits, regardless of how many packets have been sent or received.

W (timeout) Use this option to set the time(seconds) to wait for a response

Generally ping will not stop until we stop it forcefully. To make the ping command automatically stop after it sends a certain number of packets we can use -c options and a number. This sets the desired amount of ping requests, for example if we want to stop ping after two replies

ping -c 2 google.com

```
saurabh@saurabh-VirtualBox:~$ ping -c 2 google.com
PING google.com (142.250.199.142) 56(84) bytes of data.
64 bytes from bom07s36-in-f14.1e100.net (142.250.199.142): icmp_seq=1 ttl=57 ti
me=62.2 ms
64 bytes from bom07s36-in-f14.1e100.net (142.250.199.142): icmp_seq=2 ttl=57 ti
me=61.6 ms

--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 61.603/61.914/62.226/0.311 ms
```

Suppose we do not want to fill our screen with information for every packet, then we can use the -q option (Quiet output) to display the summary only. For example if we want the summary of 10 packets

ping -c 10 -q google.com

```
saurabh@saurabh-VirtualBox:~$ ping -c 10 -q  google.com
PING google.com (142.250.199.174) 56(84) bytes of data.

--- google.com ping statistics ---
10 packets transmitted, 9 received, 10% packet loss, time 9043ms
rtt min/avg/max/mdev = 67.443/70.100/73.764/2.181 ms
```

13.10 THE NETSTAT COMMAND

The netstat command abbreviated for network statistics used to detailed information about the networking information about your machine such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc. By default Netstat

is not installed in Linux distro , we can install it using

sudo apt install net-tools

A simple execution of the netstat command alone to show a relatively simple list of all active TCP connections which, for each one, will show the local IP address (your computer), the foreign IP address (the other computer or network device), along with their respective port numbers, as well as the TCP state. There are different options available also with netstat

```
saurabh@saurabh-VirtualBox:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 saurabh-VirtualB:bootpc _gateway:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    33934    /run/user/1000/syste
md/notify
unix  3      [ ]         DGRAM      CONNECTED     15634    /run/systemd/notify
unix  2      [ ]         DGRAM                    15668    /run/systemd/journal
/syslog
unix  16     [ ]         DGRAM      CONNECTED     15678    /run/systemd/journal
/dev-log
unix  8      [ ]         DGRAM      CONNECTED     15682    /run/systemd/journal
/socket
```

Option	Explanation
-a	This switch displays active TCP connections, TCP connections with the listening state, as well as UDP ports that are being listened to.
-b	This netstat switch is very similar to the **-o** switch listed below, but instead of displaying the PID, will display the process's actual file name. Using **-b** over **-o** might seem like it's saving you a step or two but using it can sometimes greatly extend the time it takes netstat to fully execute.
-e	Use this switch with the netstat command to show statistics about your network connection. This data includes bytes, unicast packets, non-unicast packets, discards, errors, and unknown protocols received and sent since the connection was established.
-f	The **-f** switch will force the netstat command to display the Fully Qualified Domain Name (FQDN) for each foreign IP addresses when possible.
-n	Use the **-n** switch to prevent netstat from attempting to determine host names for foreign IP addresses. Depending on your current network connections, using this switch could considerably reduce the time it takes for netstat to fully execute.

-o	A handy option for many troubleshooting tasks, the **-o** switch displays the process identifier (PID) associated with each displayed connection. See the example below for more about using **netstat -o**.
-p	Use the **-p** switch to show connections or statistics only for a particular *protocol*. You can not define more than one *protocol* at once, nor can you execute netstat with **-p** without defining a *protocol*.
protocol	When specifying a *protocol* with the **-p** option, you can use **tcp**, **udp**, **tcpv6**, or **udpv6**. If you use **-s** with **-p** to view statistics by protocol, you can use **icmp**, **ip**, **icmpv6**, or **ipv6** in addition to the first four I mentioned.
-r	Execute netstat with **-r** to show the IP routing table. This is the same as using the route command to execute **route print**.
-s	The **-s** option can be used with the netstat command to show detailed statistics by protocol. You can limit the statistics shown to a particular protocol by using the **-s** option and specifying that *protocol* but be sure to use **-s** before **-p** *protocol* when using the switches together.
-t	Use the **-t** switch to show the current TCP chimney offload state in place of the typically displayed TCP state.
-x	Use the **-x** option to show all NetworkDirect listeners, connections, and shared endpoints.
-y	The **-y** switch can be used to show the TCP connection template for all connection. You cannot use **-y** with any other netstat option.
time_interval	This is the time, in seconds, that you'd like the netstat command to re-execute automatically, stopping only when you use Ctrl-C to end the loop.

For example

To List all tcp ports.

 # netstat -at

```
saurabh@saurabh-VirtualBox:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:sunrpc          0.0.0.0:*               LISTEN
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp6       0      0 [::]:sunrpc             [::]:*                  LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
```

To List all udp ports.

 # netstat -au

```
saurabh@saurabh-VirtualBox:~$ netstat -au
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 localhost:domain        0.0.0.0:*
udp        0      0 saurabh-VirtualB:bootpc _gateway:bootps         ESTABLISHED
udp        0      0 0.0.0.0:sunrpc          0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 0.0.0.0:42931           0.0.0.0:*
udp6       0      0 [::]:49179              [::]:*
udp6       0      0 [::]:sunrpc             [::]:*
udp6       0      0 [::]:mdns               [::]:*
```

List only listening TCP ports.

 # netstat -lt

```
saurabh@saurabh-VirtualBox:~$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:sunrpc          0.0.0.0:*               LISTEN
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp6       0      0 [::]:sunrpc             [::]:*                  LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
```

13.11 THE TRACEROUTE COMMAND

Linux traceroute command is used as network troubleshooting utility that helps in determining the number of hops and packets traveling path required to reach a destination. The traceroute is not a default utility of the Linux system so we need to install it manually. To install it, execute the following commands:

sudo apt install traceroute

The basic syntax of traceroute is : traceroute [OPTION...] HOST . For example, to trace the route of the server 'google.com,' execute the below command:

traceroute google.com

This command will display hops, packets, and much other information on the given route. Its output has been shown bellow

```
saurabh@saurabh-VirtualBox:~$ traceroute google.com
traceroute to google.com (142.250.192.238), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.547 ms  0.515 ms  0.500 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
```

- The first line is displaying the hostname and IP address (which has not been resolved, so send it to gateway address) that is to be reached, hops that are going to be attempted by the traceroute command, and the size of the packets to be sent.
- By default, it sends three packets for each host, so three response times are listed.
- The '*' symbol represents the packet loss.

The following command-line options are supported by the traceroute command:

-f, --first-hop=NUM: It is used to set the initial hop distance.

-g, --gateways=GATES: It is used to display a list of gateways for loose source routing.

-I, --icmp: It is specified to use ICMP ECHO as a probe.

-m, --max-hop=NUM: It is used to set maximal hop count, the default is 64.

-M, --type=METHOD: It specifies the METHOD (icmp or udp) for traceroute operations, the default method is udp.

-p, --port=PORT: It is defined to use destination PORT port, the default PORT is 33434.

-q, --tries=NUM: It is used to forward NUM probe packets per hop, the default is 3.

-- resolve-hostnames: It is used to resolve the hostnames.

-t, --tos=NUM: It is used to set the type of service (TOS) to NUM.

-w, --wait=NUM: It is used to wait in seconds for a response, the default is 3.

For example, if we want to trace the Route Using IPv6 than the '6' option is used to trace the route to a host network using the Ipv6 protocol. Consider the below command:

traceroute6 google.com

To set the Number of Queries per hop we use the '-q' option. Consider the below command:

traceroute -q 2 google.com

13.12 THE TELNET COMMAND

In Linux, the telnet command is used to establish the connections between different machines. This command allows us to manage the remote devices using the CLI (command-line interface). It uses TCP port 23 which is assigned to the telnet protocol. By default, telnet utility is not present in Linux. In Ubuntu and Debian based distributions it can be installed using the command:

$ sudo apt install telnetd -y

After the installation is completed, check the status of telnet protocol using the systemctl command:

$ sudo systemctl status inetd

The telnet protocol is installed and is in running status. We know that it uses port 23, so we will enable port 23 using the ufw command:

$ sudo ufw allow 23/tcp

Reload the ufw using the command:

$ sudo ufw reload

To connect to the devices we need the IP address of the particular machine and also make sure the telnet protocol is installed in that machine as well and port 23 is enabled. In our case, the IP address of the machine to which we want to connect is 192.168.18.135, we will run the command:

$ telnet host_ipr_adress

To connect to the devices we need the IP address of the particular machine and also make sure the telnet protocol is installed in that machine as well and port 23 is enabled. In our case, the IP address of the machine

to which we want to connect is 192.168.18.135, we will run the command:

$ telnet 192.168.10.10

After connecting it will prompt you to provide user name and password. after successfully login you can use its services.

13.13 THE /ETC/SERVICES FILE

UNIX operating systems store what's called a services file at /etc/services. It stores information about numerous services that client applications might use on the computer. Within the file are the service name, port number, and protocol it uses, and any applicable aliases. /etc/services file contains a list of network services and ports mapped to them. inetd or xinetd looks at these details so that it can call a particular program when a packet hits respective port and demand for service.

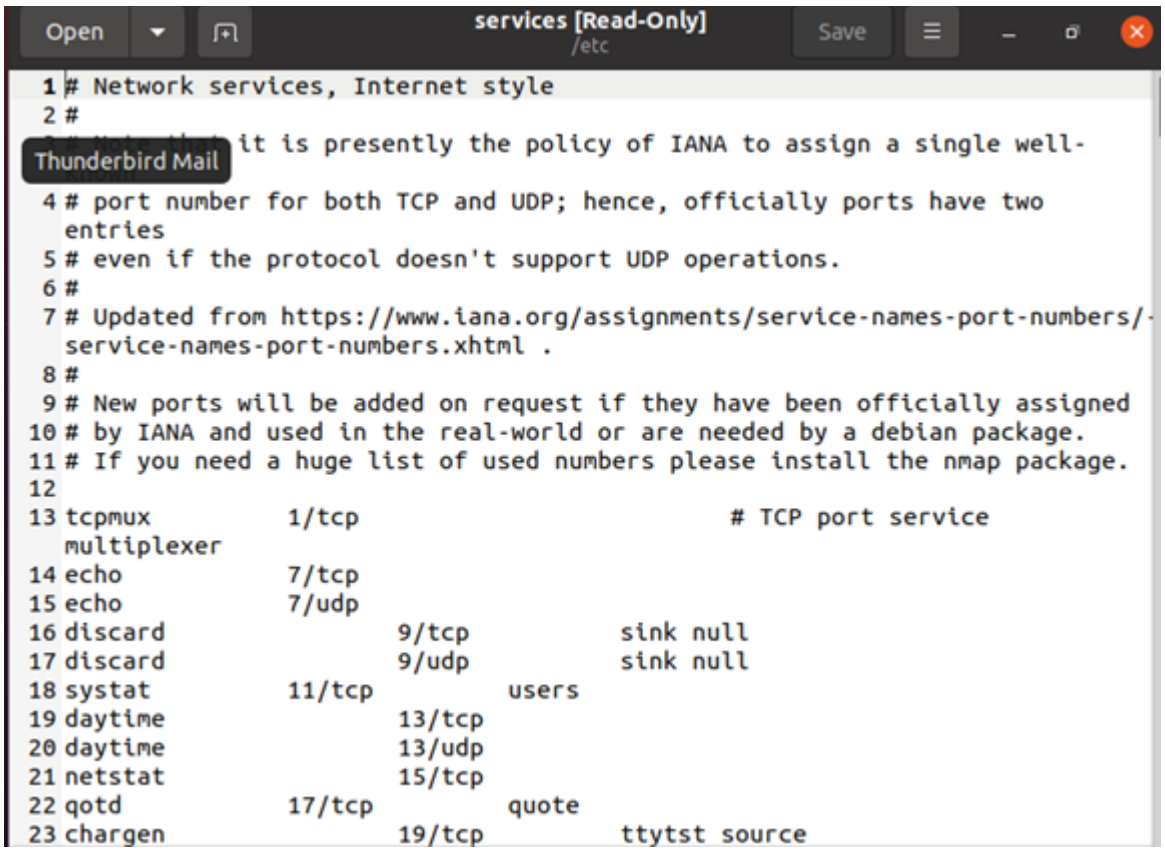
As a normal user, we can view this file since the file is world readable. To edit this file we need to have root privileges.

/etc/services file format

service-name port/protocol [aliases..] [#comment]

Last two fields are optional hence denoted in [] where -

- service-name is the name of the network service. e.g. telnet, FTP, SMTPetc.
- port/protocol is the port being used by that network service (numerical value) and protocol (TCP/UDP) used for communication by service.
- alias is an alternate name for service.
- a comment is a note or description you can add to service. Starts with # mark



13.14 PORT

Port number is a 16-bit numerical value that ranges from 0 to 65535. Well-known port (0-1023), registered port (1024-49151), and dynamic port is three types of port number space. (49152-65535). These ports can be opened and used by software applications and operating system services to send and receive data over networks (LAN or WAN) that employ certain protocols (eg TCP, UDP).For example, we use 80 for HTTP-web-based plain-text surfing and 443 for HTTPS-web-based encrypted websites in our daily work. To conclude, a port is a logical form to identify system activities or various network services used to create local or network-based communications.

Port Number	Service name	Transport protocol	Description
7	Echo	TCP, UDP	Echo service

20	FTP-data	TCP, SCTP	File Transfer Protocol data transfer
21	FTP	TCP, UDP, SCTP	File Transfer Protocol (FTP) control connection
22	SSH-SCP	TCP, UDP, SCTP	Secure Shell, secure logins, file transfers (scp, sftp), and port forwarding
23	Telnet	TCP	Telnet protocol—unencrypted text communications
25	SMTP	TCP	Simple Mail Transfer Protocol, used for email routing between mail servers
53	DNS	TCP, UDP	Domain Name System name resolver
69	TFTP	UDP	Trivial File Transfer Protocol
80	HTTP	TCP, UDP, SCTP	Hypertext Transfer Protocol (HTTP) uses TCP in versions 1.x and 2. HTTP/3 uses QUIC, a transport protocol on top of UDP
88	Kerberos	TCP, UDP	Network authentication system
102	Iso-tsap	TCP	ISO Transport Service Access Point (TSAP) Class 0 protocol
110	POP3	TCP	Post Office Protocol, version 3 (POP3)
135	Microsoft EPMAP	TCP, UDP	Microsoft EPMAP (End Point Mapper), also known as DCE/RPC Locator service, used to remotely manage services including DHCP server, DNS server, and WINS. Also used by DCOM
137	NetBIOS-ns	TCP, UDP	NetBIOS Name Service, used for name registration and resolution
139	NetBIOS-ssn	TCP, UDP	NetBIOS Session Service
143	IMAP4	TCP, UDP	Internet Message Access Protocol (IMAP), management of electronic mail messages on a server



CHECK YOUR PROGRESS

Q 1: Fill in the blanks:

(a) _____ is an unique number defining the access point to the network interface.

(b) _____ is the unique number assigned to the network interface through which it communicate with other networks.

(c) The non-unique number that defines the network itself is termed as _____

(d) _____ reduces bandwidth and improves response times by caching and reusing frequently visited web pages.

(e) A _____ is a computer server that stores a IP addresses and their associated domain names.

(f) In a _____ type of network, the first 8 bits (also called the first octet) identify the network, and the remaining have 24 bits for the host into that network.



13.15 LET US SUM UP

- ifconfig (interface configurator) command can be used to initialize an interface, assign IP Address to interface and enable or disable interface.
- Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently visited web pages.
- A DNS is a computer server that stores a IP addresses and their associated domain names. It serves to translate a domain name to its IP address, so that the computer knows which IP address to connect to for the requested contents.
- Postfix is a default mail server in most of the Linux distribution.
- Samba is a free and open-source software that allows files to be shared across Windows and Linux systems.
- An IP (Internet Protocol) address is a numerical label assigned to the devices connected to a computer network that uses the IP for

communication. IP address act as an identifier for a specific machine on a particular network.

- Ping is short for Packet Internet Groper used for checking the network connectivity among host/server and host.
- The netstat command abbreviated for network statistics which is used to get the detailed information about the networking information about a machine such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc.



13.16 FURTHER READINGS

1) Linux Administration Cookbook By Adam K. Dean
2) Linux Administration By Jason Cannon



13.17 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: (a) ip address

Ans to Q No 1: (b) gateway

Ans to Q No 1: (c) netmask

Ans to Q No 1: (d) squid

Ans to Q No 1: (e) DNS

Ans to Q No 1: (f) Class A



13.18 MODEL QUESTIONS

Q 1: Configure IP address and netmask of a system using ifconfig .

Q 2: Why we use proxy sever like Squid?

Q 3: Explain how Apache server works.

Q 4: Explain different classes of IP address.

Q 5: Explain the procedure to connect SSH client to its server

Q 6: Explain at least 5 options available with ping command.

Q 7: What are the contents of /etc/services file?

*** ***** ***

UNIT 14: NETWORK PROTOCOLS

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Configuring NFS

14.4 Configuring NIS

14.5 Configuring DNS

14.6 Configuring FTP

14.7 Configuring Squid Proxy

14.8 Contiguring Firewall

14.9 Configuring DHCP server

14.10 ip Tables and Firewall

14.11 Let Us Sum Up

14.11 Further Readings

14.13 Answers to Check Your Progress

14.14 Model Questions

14.1 LEARNING OBJECTIVES

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

- understand the basic configuration of NFS service
- understand NIS service and its configuration
- learn how to set up FTP and DHCP server
- learn to setup basic ip table rules to use as firewall
- browse using Squid Proxy

14.2 INTRODUCTION

In the previous unit, we have learnt how to install and configure various Linux services like proxy server, Apache web server etc. In this unit, we will discuss how to configure Network File System(NFS), Network Information Service(NIS) and Domain Name Service(DNS). We will also discuss how to set up FTP and DHCP server. In the last part of the unit, we will discuss how to use the ip table and browse using squid proxy.

14.3 CONFIGURING NFS

Sharing files between computers and servers is an essential networking task. Network File System (NFS) is a way of mounting Linux directories over a network. It was basically developed to share files and folders between two Linux systems faster and better. Sun Microsystems developed NFS in 1980 for this sole purpose. NFS uses Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) to provide a stateless connection between client and server. NFS is one of the best methods to share between to two Linux systems. The main purpose of NFS is to back up data into a central server that is accessible to all PCs in a network. To use NFS, first we need to install it.

NFS works on server/client model where one machine works as server and another one works as client. We need to install server part and client part in the respective machine separately. For installation first we need to update our systems with command **sudo apt update**.

```
saurabh@saurabh-VirtualBox:~$ sudo apt update
[sudo] password for saurabh:
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease
```

Then we have to install NFS server using the command **sudo apt install nfs-kernel-server.**

```
saurabh@saurabh-VirtualBox:~$ sudo apt install nfs-kernel-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

Here nfs-kernel-server is a package name.

Once the packages are finished installing, it's time to configure the exports file. To configure the export file, we can open it with any text editor. If we are not in root mode, we have to use sudo while editing. The exports file specifies which directories the server will export to the network. Once we open /etc/exports file, it displays as follows.

```
1 # /etc/exports: the access control list for filesystems which may be exported
2 #               to NFS clients.  See exports(5).
3 #
4 # Example for NFSv2 and NFSv3:
5 # /srv/homes       hostname1(rw,sync,no_subtree_check)
  hostname2(ro,sync,no_subtree_check)
6 #
7 # Example for NFSv4:
8 # /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
9 # /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
10 #
```

The structure of the file is simple. To the left are the directories which are to be shared, and to the right are the IP addresses and subnets of the machines which are also to be shared along with any specific options. Suppose we want to share nfs _share folder to the machine having IP address 192.168.87.158, then we have to add following line to exports file at last.

/nfs_share 192.168.87.158(rw,sync,no_root_squash,no_subtree_check)

Explanation

- /nfs_share - is the share folder which server wants to share.
- 192.168.87.158 - is the IP address of the client to whom we want to share.
- rw - This means all the clients to read and write the files to the share directory.
- sync - which will confirm the shared directory once the changes are committed.
- no_subtree_check - will prevent from the scanning the shared directory, as nfs performs the scans of every share directory, Disabling the subtree check will increase the reliability, but reduces the security.
- no_root_squash - This helps all the root user to connect to the designated directory.

Alternate parameter

- subtree_check: specifies that, in the case of a directory is exported instead of an entire filesystem, the host should verify the location of files and directories on the host filesystem.
- async: ignores synchronization checks in favour of increased speed

```
1 # /etc/exports: the access control list for filesystems which may be exported
2 #               to NFS clients.  See exports(5).
3 #
4 # Example for NFSv2 and NFSv3:
5 # /srv/homes       hostname1(rw,sync,no_subtree_check)
  hostname2(ro,sync,no_subtree_check)
6 #
7 # Example for NFSv4:
8 # /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
9 # /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
10 # /nfs_share 192.168.87.158(rw,sync,no_root_squash,no_subtree_check)
```

After we make the necessary edits in /etc/exports, use the exportfs command to export all shared directories registered in that file:

sudo exportfs -a

Once the exports file is ready to go, the server can be started and made to start on boot with Systemd. There are two parts of starting the NFS server, rpcbind and nfs-server. In Ubuntu, the server will have to start automatically, so we may need to run:

$ sudo systemctl restart rpcbind

$ sudo systemctl restart nfs-server

After that, the server will be up and running. Once again, since Debian and Ubuntu run services at boot time by default, nothing is required to make NFS run automatically at boot.

Connecting The Client

Setting up a client to connect to an NFS share requires much less effort. Again, for both Debian/Ubuntu, we need to install nfs-common package.

$ sudo apt-get install nfs-common rpcbind

Once it is started, or in the case of Debian/Ubuntu, restart using rpcbind.

$ sudo systemctl (re)start rpcbind

Once the packages are installed on the client, we need to create the directory in client side to mount point the shared folder.

 # mkdir -p /mnt/share

Mounting the Share Directory

 # mount 192.168.87.156:/nfs_share /mnt/share/

To confirm if the share is mounted or not, run the command 'df -h',

this will show the list of mounted folders. Here nfs_share folder from server 192.168.87.156 has been mounted and we can observe that the server and client are in same network (192.168.87.0).

14.4 CONFIGURING NIS

Network Information Service (NIS) is a distributed naming service based on Remote Procedure Call (RPC). It enables easy sharing of various information across the cluster including username, password, hosts, and service ports. Such a centralized user management system is also a prerequisite for setting up cluster management system including SLURM, which requires user & group synchronization across the cluster.

NIS architecture overview

NIS uses a client-server arrangement. By running NIS, the system administrator can provide administrative databases, called maps, among a variety of servers. Servers are further divided into master and slave servers: the master server is the true single owner of the map data. Slave NIS servers handle client requests, but they do not modify the NIS maps. The master server is responsible for all map maintenance and distribution to its slave servers. Once an NIS map is built on the master to include a change, the new map file is distributed to all slave servers.

Clients are hosts that request information from these maps. NIS clients observe these changes when they perform queries on the map file - it doesn't matter whether the clients are talking to a master or a slave server, because once the map data is distributed, all NIS servers have the same information. NIS architecture is depicted in figure 1.

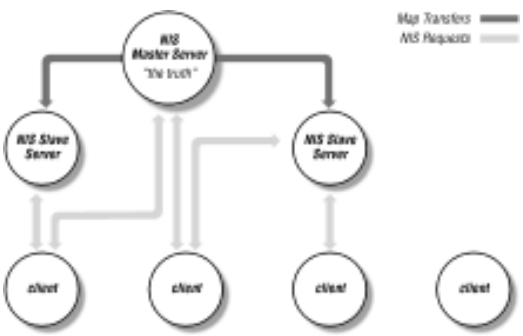


Figure 1. NIS Architecture

NIS uses domains to arrange the machines, users, and networks in its namespace. However, it does not use a domain hierarchy as the NIS namespace is flat.

Set Up NIS

NIS Master

- **Installing NIS**

First update the system then install the NIS package.

$ sudo apt install nis

System will prompt to enter preferred domain name during the installation process. Here, the domain name is test.nis.

```
Preconfiguring packages ...
```

```
# input your domain name
+---------------------------| Configuring nis |---------------------------+
| Please choose the NIS "domainname" for this system. If you want this     |
| machine to just be a client, you should enter the name of the NIS domain |
| you wish to join.                                                        |
|                                                                          |
| Alternatively, if this machine is to be a NIS server, you can either     |
| enter a new NIS "domainname" or the name of an existing NIS domain.      |
|                                                                          |
| NIS domain:                                                              |
|                                                                          |
| test.nis_____________________________________________________________    |
|                                                                          |
|                          <Ok>                                            |
|                                                                          |
+--------------------------------------------------------------------------+
```

2. Edit /etc/default/nis

$ sudo vim /etc/default/nis

Specifically, edit NISSERVER to be master

```
#
# /etc/defaults/nis     Configuration settings for the NIS daemons.
#
# Are we a NIS server and if so what kind (values: false, slave, master)?
NISSERVER=false # Should be master
```

3. Edit /etc/hosts

In master's /etc/hosts, there should be at least all the slave servers.

$ sudo vim /etc/hosts

SHELL

192.168.0.233 test # master

147.46.xx.xxx slave_test # slave

4. **Edit /etc/ypserv/securenets**

```
$ sudo vim /etc/ypserv.securenets # This line
gives access to everybody. PLEASE ADJUST!
# comment out
# 0.0.0.0 0.0.0.0
# add to the end: IP range you allow to access
# Allows IP in the form of: 192.168.0.* to access
255.255.255.0   192.168.0.0
# Allows certain internet IP to access:
255.255.255.255 147.46.xx.xxx
```

5. **Run ypinit -m**

By running ypinit -m, NIS will utilize the local user system as its cornerstone to build the network user information.

$ sudo /usr/lib/yp/ypinit -m

We will be prompted to enter the list of servers (master and slave). The first host added should be the hostname of the master server, followed by all the slave servers in the system. At this point, we have to construct a list of the hosts which will run NIS servers. test is in the list of NIS server hosts. It is to continue to add the names for the other hosts, one per line. When we have done with the list, type <control D>.

next host to add: test # master

next host to add: slave_test # slave

next host to add: # <- Ctrl + D

The current list of NIS servers looks like this:

test

slave_test

Is this correct? [y/n: y] y

We need a few minutes to build the databases...

Building /var/yp/test.nis/ypservers...

Running /var/yp/Makefile...

make[1]: Entering directory '/var/yp/test.nis'

Updating passwd.byname...

Updating passwd.byuid...

Updating group.byname...

Updating group.bygid...

Updating hosts.byname...

Updating hosts.byaddr...

Updating rpc.byname...

Updating rpc.bynumber...

Updating services.byname...

Updating services.byservicename...

Updating netid.byname...

Updating protocols.bynumber...

Updating protocols.byname...

Updating netgroup...

Updating netgroup.byhost...

Updating netgroup.byuser...

Updating shadow.byname...

make[1]: Leaving directory '/var/yp/test.nis'

test has been set up as a NIS master server.

Now we can run ypinit -s test on all slave server.

NIS client

1. **Install NIS**

   $ sudo apt install nis

   During the process it will also prompt to enter domain name. Use the same domain as the NIS master.

2. **Edit /etc/yp.conf**

   $ sudo vim /etc/yp.conf

   To Add the master / slave server, we want to request info from.

   ### FORMAT:

   # domain <domainname> server <IP or hostname>

   ### OR:

   # ypserver <IP or hostname>

   domain test.nis server 192.168.1.233

   If we want to use hostname instead of server IP, we make sure to specify it in /etc/hosts of the client.

3. Edit /etc/nsswitch.conf

The name service switch (named nsswitch.conf) controls how a client machine or application obtains network information. Each machine has a switch file in its /etc directory. Each line of that file identifies a particular type of network information, such as host, password, and group, followed by one or more locations of that information. A client can obtain naming information from one or more of the switch's sources. For example, an NIS client could obtain its hosts information from an NIS map and its password information from a local /etc file. In addition, the client could specify the conditions under which the switch must use each source.

The available information sources are listed in the following table:

Information Sources	Description
files	A file stored in the client's /etc directory. For example, /etc/passwd
nisplus	An NIS+ table. For example, the hosts table.
nis	An NIS map. For example, the hosts map.
compat	compat can be used for password and group information to support old-style + or - syntax in /etc/passwd, /etc/shadow, and /etc/group files.
dns	Can be used to specify that host information be obtained from DNS.
ldap	Can be used to specify entries be obtained from the LDAP directory.

Open /etc/nsswitch.conf to add nis as an information source.

$ sudo gedit /etc/nsswitch.conf

Add nis to the end of passwd, group, shadow, gshadow and hosts in the following way.

```
# ...
passwd:    files systemd nis  # add nis
group:     files systemd nis  # add nis
shadow:    files nis          # add nis
```

gshadow: files nis # add nis

hosts: files dns nis # add nis

...

If we want to look up nis first instead of local files, we have to put nis [NOTFOUND=return] in front of files. The [NOTFOUND=return] search criterion instructs the switch to stop searching the NIS tables if the switch gets a "No such entry" message. The switch searches through local files only if the NIS server is unavailable.

...

passwd: nis [NOTFOUND=return] files systemd # add nis

...

4. **Edit /etc/pam.d/common-session for creating home directory automatically.**

   $ gedit /etc/pam.d/common-session

   # add to the end

   session optional pam_mkhomedir.so =/etc/skel umask=000

5. **Restart NIS to apply changes**

   $ sudo systemctl restart rpcbind

   $ sudo systemctl restart nis

NIS Slave

The set-up process of slave server is an approximate combination of client set up and master set up.

1. **Go through NIS client set-up steps**

   Specifically through Step 1 to 4. This is for establishing RPC connection with master.

2. **Edit /etc/default/nis**

   $ sudo gedit /etc/default/nis

   Specifically, edit NISSERVER to be slave.

   #

   # /etc/defaults/nis Configuration settings for the NIS daemons.

   #

   # Are we a NIS server and if so what kind (values: false, slave, master)?

   NISSERVER=false # Should be slave

3. **Edit /etc/ypserv.securenets**

   $ gedit /etc/ypserv.securenets

   It is important to allow the master server to access.

   # This line gives access to everybody. PLEASE ADJUST!

   # comment out

   # 0.0.0.0 0.0.0.0

   # add to the end: IP range you allow to access

   # Allows IP in the form of: 192.168.100.* to access

   255.255.255.0 192.168.100.0

   # Allows certain internet IP to access:

   255.255.255.255 147.46.xx.xxx

4. **Edit /etc/hosts**

   $ sudo gedit /etc/hosts

   In slave's /etc/hosts, there should be at least the IP of master server.

   147.47.xxx.xx test # master

5. **Run ypinit -s <master>**

   $ sudo /usr/lib/yp/ypinit -s tes

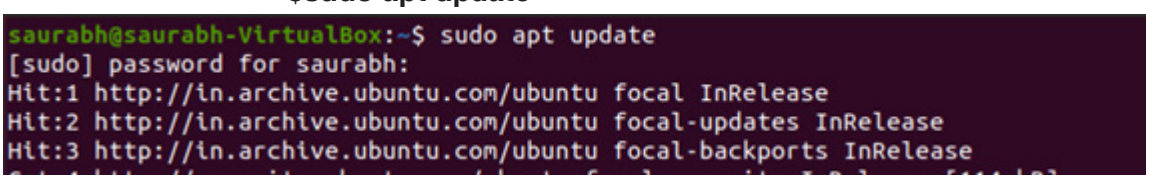
   This operation will pull information from the master to the slave.

14.5 CONFIGURING DNS

Domain Name Service (DNS) is an Internet service that maps IP addresses and fully qualified domain names (FQDN) to one another. In this way, DNS alleviates the need to remember IP addresses. Computers that run DNS are called name servers. Ubuntu ships with BIND (Berkley Internet Naming Daemon), the most common program used for maintaining a name server on Linux.

To use domain name server, first we need to install it. To install DNS, we must update our system as:

$sudo apt update



Now we will install the DNS server by using the command bind9:

$sudo apt install bind9

```
saurabh@saurabh-VirtualBox:~$ sudo apt install bind9
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bind9-dnsutils bind9-libs bind9-utils python3-ply
Suggested packages:
```

Next step is to install the utilities of the DNS by using the command of "dnsutils":

$sudo apt install dnsutils

```
saurabh@saurabh-VirtualBox:~$ sudo apt install dnsutils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  dnsutils
```

There are many ways to configure BIND9. Some of the most common configurations are a caching nameserver, primary server, and secondary server.

- When configured as a caching nameserver, BIND9 will find the answer to name queries and remember the answer when the domain is queried again.
- As a primary server, BIND9 reads the data for a zone from a file on its host and is authoritative for that zone.
- As a secondary server, BIND9 gets the zone data from another nameserver that is authoritative for the zone.

The DNS configuration files are stored in the /etc/bind directory. The primary configuration file is /etc/bind/named.conf, which in the layout provided by the package just includes these files.

- /etc/bind/named.conf.options: global DNS options
- /etc/bind/named.conf.local: for your zones
- /etc/bind/named.conf.default-zones: default zones such as localhost, its reverse, and the root hints

The root nameservers is used to be described in the file /etc/bind/db.root. This is now provided instead by the /usr/share/dns/root.hints file shipped with the dns-root-data package and is referenced in the

named.conf.default-zones configuration file above.

Now to configure the DNS, we will first go to the address /etc/bind/named.conf.options and add the Google DNS for just understanding. We can edit it by gedit

sudo gedit /etc/bind/named.conf.options

Replace the forwarders text in config file as, 8.8.8.8 i.e. is Google's DNS:

```
12
13          // forwarders {
14          //      0.0.0.0;
15          // };
```

Restart the DNS service using systemctl

$ sudo systemctl restart bind9

Now check the status of bind using the command

$ sudo systemctl status bind9

```
● named.service - BIND Domain Name Server
     Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset>
     Active: active (running) since Sat 2022-07-23 22:57:31 IST; 34min ago
       Docs: man:named(8)
   Main PID: 12421 (named)
      Tasks: 5 (limit: 1105)
     Memory: 12.7M
     CGroup: /system.slice/named.service
             └─12421 /usr/sbin/named -f -u bind

Jul 23 22:58:36 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:36 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:41 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:41 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:43 saurabh-VirtualBox named[12421]: network unreachable resolving>
Jul 23 22:58:44 saurabh-VirtualBox named[12421]: resolver priming query comple>
Jul 23 22:58:44 saurabh-VirtualBox named[12421]: managed-keys-zone: Unable to >
```

As the bind9 is running now, we will test the domain which we edit in the configuration file as:

$ dig google.com

```
saurabh@saurabh-VirtualBox:~$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 55443
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sat Jul 23 23:41:58 IST 2022
;; MSG SIZE  rcvd: 39
```

Now we will do primary zone configuration, but before configuration let's have an idea what is going on. At bind directory, we have two zone files: one is global DNS conf file named.conf & second is local DNS config file named.conf.local. We need to create forward and reverse zones. We can store some files at some host and in this configuration, the DNS gets its data from that specific file for its zone. Now for such primary master configuration, we will run the editor and will confirm that the following three commands are there and not commented on.

include "/etc/bind/named/.conf.option";

include "/etc/bind/named/.conf.local";

include "/etc/bind/named/.conf.default-zones";

To verify this, we will open named.conf file with the editor:

$ sudo gedit /etc/bind/named.conf

```
1 // This is the primary configuration file for the BIND DNS server named.
2 //
3 // Please read /usr/share/doc/bind9/README.Debian.gz for information on the
4 // structure of BIND configuration files in Debian, *BEFORE* you customize
5 // this configuration file.
6 //
7 // If you are just adding zones, please do that in /etc/bind/named.conf.local
8
9 include "/etc/bind/named.conf.options";
10 include "/etc/bind/named.conf.local";
11 include "/etc/bind/named.conf.default-zones";
```

All the three lines are present in the output. Now for example, we have a domain "example.com" for which we are going to configure.

Primary Server

In this section, BIND9 will be configured as the Primary server for the domain example.com.

Forward Zone File

To add a DNS zone to BIND9, turning BIND9 into a Primary server, first we need to edit /etc/bind/named.conf.local: so we will edit the file named.conf.local by gedit and add following lines :

```
zone "gamer.com" {
type master;
file "/etc/bind/db.example.com";
};
```

$ sudo gedit /etc/bind/named.conf.local

```
1 //
2 // Do any local configuration here
3 //
4
5 // Consider adding the 1918 zones here, if they are not used in your
6 // organization
7 //include "/etc/bind/zones.rfc1918";
8 zone "example.com" {
9 type master;
10 file "/etc/bind/db.example.com";
11 };
```

We will copy the contents from the db.local to the db.example.com:

```
saurabh@saurabh-VirtualBox:~$ sudo cp /etc/bind/db.local /etc/bind/db.example.c
om
[sudo] password for saurabh:
saurabh@saurabh-VirtualBox:~$
```

Now we are opening the newly created file example.com

```
saurabh@saurabh-VirtualBox:~$ sudo gedit /etc/bind/db.example.com
```

```
1 ;
2 ; BIND data file for local loopback interface
3 ;
4 $TTL    604800
5 @       IN      SOA     localhost. root.localhost. (
6                               2               ; Serial
7                          604800               ; Refresh
8                           86400               ; Retry
9                         2419200               ; Expire
10                         604800 )       ; Negative Cache TTL
11 ;
12 @       IN      NS      localhost.
13 @       IN      A       127.0.0.1
14 @       IN      AAAA    ::1
```

Edit the new zone file /etc/bind/db.example.com and change localhost. to the name of your server, leaving the additional . at the end. Change 127.0.0.1 to the nameserver's IP Address and root.localhost to a valid email address, but with a . instead of the usual @ symbol, again leaving the . at the end. Change the comment to indicate the domain that this file is for.

```
;
; BIND data file for example.com
;
$TTL    604800
@       IN      SOA     example.com. root.example.com. (
                              2                 ; Serial
                         604800                 ; Refresh
                          86400                 ; Retry
                        2419200                 ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.example.com.
@       IN      A       10.0.2.15
@       IN      AAAA    ::1
ns      IN      A       10.0.2.15
```

Create an A record for the base domain, example.com. Also, create an A record for ns.example.com, the name server in this example:

Here 10.0.2.15 is my nameserver's IP address. If DNS server and nameserver are same then no need to change anything, localhost bind data file is enough. Now we need to restart the bind services to make effect of all changes

$ sudo systemctl restart bind9

Reverse Zone File

Now that the zone is setup and resolving names to IP Addresses, a Reverse zone needs to be added to allow DNS to resolve an address to a name.

Edit /etc/bind/named.conf.local and add the following:

```
zone "10.0.2.in-addr.arpa" {
    type master;
    file "/etc/bind/db.10";
};
```

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
zone "example.com" {
type master;
file "/etc/bind/db.example.com";
};
zone "10.0.2.in-addr.arpa" {
type master;
file "/etc/bind/db.10";
};
```

10.0.2 this one will be the first three octet of IP address in the used network. Now, create the /etc/bind/db.10 file:

sudo cp /etc/bind/db.127 /etc/bind/db.10

Next edit /etc/bind/db.10 for changing the same options as /etc/bind/db.example.com:

```
;
; BIND reverse data file for local 10.0.2.xxx net
;
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.
1.0.0   IN      PTR     ns.example.com.
```

In this last part, we will confirm the configurations first by executing all these commands and check whether they generate errors or not:

$ named-checkzone gamer.com /etc/bind/db.gamer.com

$ named-checkzone your_IP /32 /etc/bind/db.10

$ named-checkconf /etc/bind/named.conf.local

$ named-checkconf /etc/bind/named.conf

The results will be somewhat like this if all is correct.

```
saurabh@saurabh-VirtualBox:~$ named-checkzone example.com /etc/bind/db.example.
com
zone example.com/IN: loaded serial 2
OK
saurabh@saurabh-VirtualBox:~$
```

14.6 CONFIGURING FTP

FTP stands for "File Transfer Protocol" and is a great protocol for downloading files from a remote or local server, or uploading files onto the server. It works by having a server that is listening for connections (on port 21 by default) from clients. The clients can access a remote directory with their user account, and then download or upload files there, depending on the permissions that have been granted to them. It is also possible to configure anonymous authorization, which means that users will not need their own account in order to connect to the FTP server.

On Ubuntu Linux, there are a multitude of different FTP server and client software packages available. We can even use default GUI and command line tools as an FTP client. A very popular and highly configurable FTP server package is vsftpd, available for many Linux systems, including Ubuntu. We can use this sever.

Install and configure vsftpd on Ubuntu

First, we need to update our system as earlier than we can install using the command:

$ sudo apt install vsftpd

where vsftpd is the package name.

```
saurabh@saurabh-VirtualBox:~$ sudo apt install vsftpd
[sudo] password for saurabh:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  vsftpd
```

Once the installation is completed, the service will be disabled initially. Therefore, we need to start it manually and also enable it to start automatically from the next system boot.

---------- On SysVInit ----------

 # service vsftpd start

 # chkconfig --level 35 vsftpd on

Next, if we have UFW firewall enabled on the server, we have to open ports 21 and 20 where the FTP daemons are listening, in order to allow access to FTP services from remote machines, then add the new

firewall rules as follows:

$ sudo ufw allow 20/tcp

$ sudo ufw allow 21/tcp

$ sudo ufw status

 # chkconfig --level 35 vsftpd on

Configure vsftpd server

The configuration file for vsftpd server exists in /etc directory. It is always a best practice to keep a copy of original configuration file.

$ sudo mv /etc/vsftpd.conf /etc/vsftpd.conf_origNe

Next open the config file using any editor. Here we have opened using gedit.

$ sudo gedit /etc/vsftpd.conf

Then we can add/modify the following options with these values in config file.

anonymous_enable=NO	# disable anonymous login
local_enable=YES	# permit local logins
write_enable=YES	# enable FTP commands which change the filesystem
local_umask=022	# value of unmask for file creation for local users
dirmessage_enable=YES	# enable showing of messages when users first enter a new directory
xferlog_enable=YES	# a log file will be maintained detailing uploads and downloads
connect_from_port_20=YES	# use port 20 on the server machine for PORT connect
xferlog_std_format=YES	# keep standard log file format
listen=NO	# prevent vsftpd from running in standalone mode
listen_ipv6=YES	# vsftpd will listen on an IPv6 socket instead of an IPv4 one
pam_service_name=vsftpd	# name of the PAM service vsftpd

will use

userlist_enable=YES # enable vsftpd to load a list of usernames

tcp_wrappers=YES # turn on tcp wrappers

Now, we have to configure VSFTPD to allow/deny FTP access to users based on the user list file /etc/vsftpd.userlist. Note that by default, users listed in userlist_file=/etc/vsftpd.userlist are denied login access with userlist_deny=YES option if userlist_enable=YES. But, the option userlist_deny=NO twists the meaning of the default setting, so only users whose username is explicitly listed in userlist_file=/etc/vsftpd.userlist will be allowed to login to the FTP server.

userlist_enable=YES # vsftpd will load a list of usernames, from userlist_file

userlist_file=/etc/vsftpd.userlist # stores usernames.

userlist_deny=NO

With the configuration file saved and the firewall rules updated, restart vsftpd to apply the new changes.

$ sudo systemctl restart vsftpd

Create an FTP user

Our FTP server is ready to receive incoming connections. So, now it's time to create a new user account that we'll use to connect to the FTP service. Use this first command to create a new account called ftpuser1, and the second command to set a password for the account:

$ sudo useradd -m ftpuser

$ sudo passwd ftpuser

New password:

Retype new password:

passwd: password updated successfully

In order to verify that everything is working properly, we should store one file in ftpuser home directory. This file should be visible when we login to FTP in the next steps.

$ sudo bash -c "echo FTP TESTING > /home/ftpuser/FTP-TEST"

Connect to FTP server via command line

We should now be able to connect to your FTP server either by IP address or hostname. First, ensure that the default FTP client utility is installed on the system by running the following command. It will either install the software or tell you that it is already available on the system.

$ sudo apt install ftp

To connect from command line and verify that everything is working properly, open a terminal and use Ubuntu's ftp command to connect to your loopback address (127.0.0.1).

$ ftp 127.0.0.1

Now, we can login to the FTP server by specifying the username and password that we configured earlier.

14.7 CONFIGURING SQID PROXY

Squid is a full-featured caching proxy supporting popular network protocols like HTTP, HTTPS, FTP, and more. It can be used to improve the web server's performance by caching repeated requests, filter web traffic, and access geo-restricted content.

Installing Squid on Ubuntu

The squid package is included in the standard Ubuntu 20.04 repositories. To install it, run the following commands as sudo user

sudo apt update

sudo apt install squid

Once the installation is completed, the Squid service will start automatically. To verify it, check the service status using command.

sudo systemctl status squid

```
● squid.service - Squid Web Proxy Server
     Loaded: loaded (/lib/systemd/system/squid.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2020-10-23 19:02:43 UTC; 14s ago
       Docs: man:squid(8)
```

The output will look similar to this.

Configuring Squid

The squid service can be configured by editing the /etc/squid/squid.conf file. The configuration file contains comments that describe what each configuration option does. Before making any changes, it is recommended to back up the original configuration file

sudo cp /etc/squid/squid.conf /etc/squid/squid.conf.original

To start configuring the squid instance, open the file in any text editor, here we use gedit

sudo gedit /etc/squid/squid.conf

By default, squid is set to listen on port 3128 on all network interfaces on the server. If we want to change the port and set a listening interface, locate the line starting with http_port and specify the interface IP address and the new port. If no interface is specified, Squid will listen on all interfaces. Running Squid on all interfaces and on the default port should be fine for most users.

```
# Squid normally listens to port 3128
http_port IP_ADDR:PORT
```

Squid allows us to control how the clients can access the web resources using Access Control Lists (ACLs). By default, access is permitted only from the localhost. If all clients who use the proxy have a static IP address. The easiest option to restrict access to the proxy server is to create an ACL that will include the allowed IPs. Otherwise, we can set squid to use authentication. Instead of adding the IP addresses in the main configuration file, create a new text that will store the allowed IPs:

```
etc/squid/allowed_ips.txt
192.168.33.1
192.168.33.2
# All other allowed IPs
```

Now open the main configuration file and create a new ACL named allowed_ips and allow access to that ACL using the http_access directive. The order of the http_access rules is important. Make sure we have added

the line before http_access deny all.

```
/etc/squid/squid.conf
# ...
acl allowed_ips src "/etc/squid/allowed_ips.txt"
# ...
#http_access allow localnet
http_access allow localhost
http_access allow allowed_ips
# And finally deny all other access to this proxy
http_access deny all
```

After changes to the configuration file, we need to restart the Squid service for the changes to take effect:

sudo systemctl restart squid

Squid Authentication

If restricting access based on IP doesn't work for our use case, we can configure squid to use a back-end to authenticate users. Squid supports Samba, LDAP, and HTTP basic auth. Here, we will use basic auth. It is a simple authentication method built into the HTTP protocol.

To generate a crypted password, use the openssl tool. The following command appends the

USERNAME:PASSWORD pair to the /etc/squid/htpasswd file.

```
printf "USERNAME:$(openssl passwd -crypt PASSWORD)\n"
| sudo tee -a /etc/squid/htpasswd
```

```
printf "user1:$(openssl passwd -crypt 'Password1')\n" | sudo
tee -a /etc/squid/htpasswd
```

For example, to create a user "user1" with password "Password1", we would run:

Output of above command will look like this:

user1:QMxVjdyPchJl6

The next step is to enable the HTTP basic authentication and include the file containing the user credentials to the squid configuration file. Open the main configuration using any editor

sudo gedit /etc/squid/squid.conf

And add the following:

```
/etc/squid/squid.conf
# ...
auth_param basic program /usr/lib/squid3/basic_ncsa_auth
/etc/squid/htpasswd
auth_param basic realm proxy
acl authenticated proxy_auth REQUIRED
# ...
#http_access allow localnet
http_access allow localhost
http_access allow authenticated
# And finally deny all other access to this proxy
http_access deny all
```

The first three highlighted lines are creating a new ACL named authenticated, and the last highlighted line is allowing access to authenticated users.

Restart the Squid service:

sudo systemctl restart squid

14.8 CONFIGURING FIREWALL

To open the Squid ports, enable the UFW 'Squid' profile:

sudo ufw allow 'Squid'

If Squid is running on another, non-default port, for example, 8888, we can allow traffic on that port with: sudo ufw allow 8888/tcp.

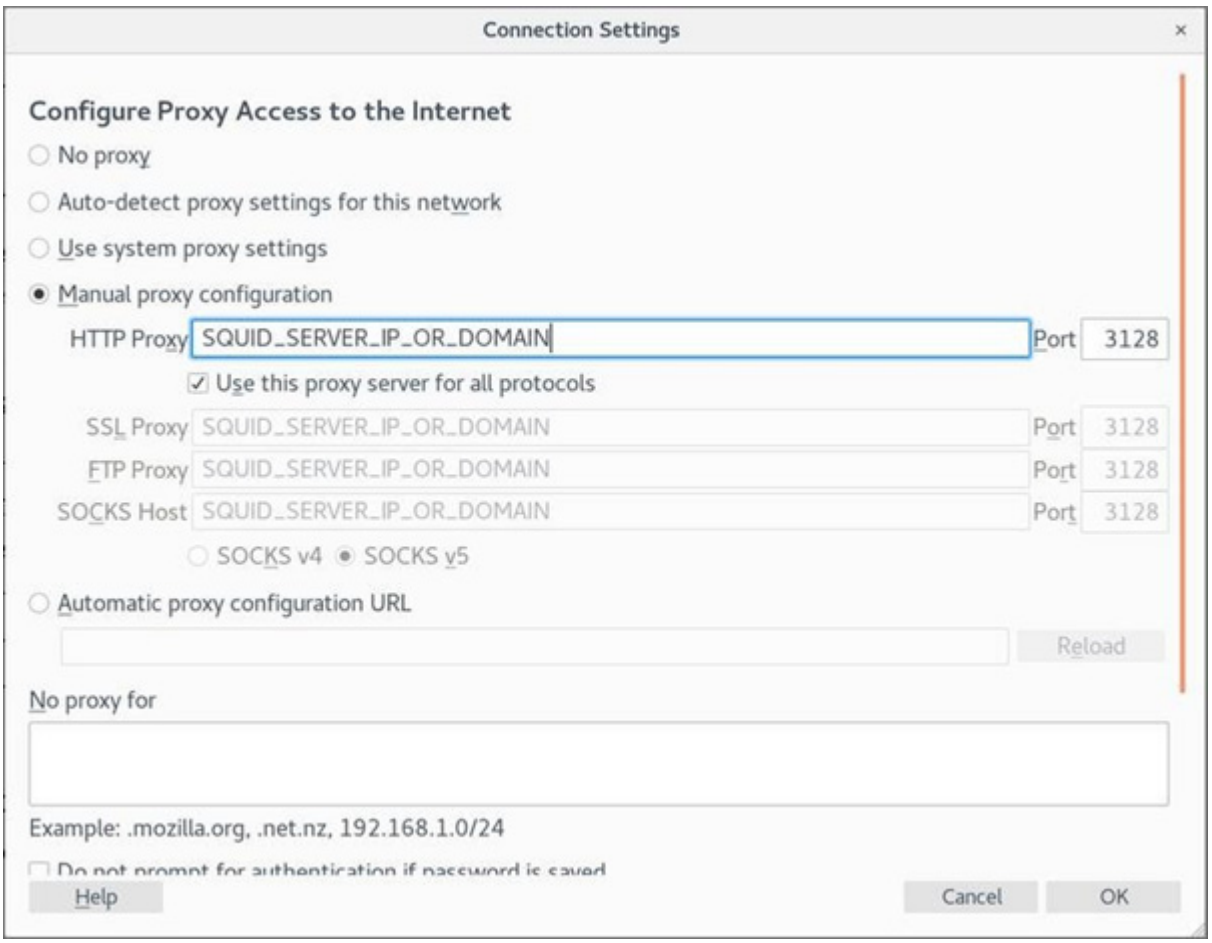
Configuring Your Browser to Use Proxy

Now that we have Squid set up, the last step is to configure a browser to use it. As ubunttu uses Firefox as default browser we will try to configure that.

Firefox

- In the upper right-hand corner, click on the hamburger icon ≡ to open Firefox's menu:

- Click on the ? Preferences link.
- Scroll down to the Network Settings section and click on the Settings... button.
- A new window will open.
- Select the Manual proxy configuration radio button.
- Enter the Squid server IP address in the HTTP Host field and 3128 in the Port field.
- Select the Use this proxy server for all protocols checkboxes.
- Click on the OK button to save the settings.



At this point, our Firefox has been configured, and we can browse the Internet through the Squid proxy. To verify it, open google.com, type "what is my ip" and you should see your Squid server IP address. To revert back to the default settings, go to Network Settings, select the Use system proxy settings radio button and save the settings.

14.9 CONFIGURING DHCP SERVER

When the computer boots up, it doesn't have an IP Address. It sends a broadcast message called a DHCP Discover. DHCP Servers are designed to respond to such broadcasts. They then send unicast traffic known as the DHCP Offer back to the requesting client. This DHCP Offer typically contains the Assigned IP Address, the Default Gateway's IP Address, and the DNS Server's IP Address. These are dynamic IP allocation.

The client on receiving the Offer sends a DHCP Request to the DHCP Server acknowledging that it has accepted the information given to it by the server. DHCP Servers keep a record of the assigned IP Addresses to prevent duplicate IP Address Collisions. DHCP Servers present on the local network and there shouldn't be more than 1 DHCP Server on a local network.

Allocation Methods for DHCP

Following are the two allocation methods for a DHCP Server:

Manual: In this method, the IP Address is given based on the MAC Address of the client system. It ensures that a particular machine always gets a fixed IP Address as its IP Address has been tied to its MAC Address. The DHCP Server sends a constant configuration to the client depending on its MAC Address in this type of allocation.

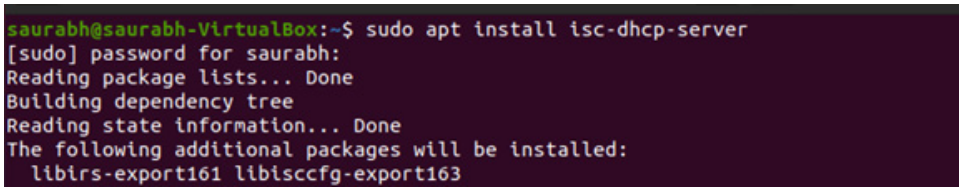
Automatic: In this method, the IP Addresses are assigned automatically by the DHCP Server on FCFS basis from a pool of addresses. It can be further divided into two categories based on the Lease Time - The time for which an IP Address is assigned to a client.

- **Fixed Lease Time:** When a DHCP client is no longer on the network for a specified period, the configuration is expired and released back to the address pool for use by other DHCP Clients. The client has to renegotiate to keep the previous IP Address.
- **Infinite Lease Time:** This has the effect of permanently assigning an IP Address to a client.

Installing DHCP Server

We can install the DHCP Server using the apt command as follows:

sudo apt install isc-dhcp-server



The main configuration file of ISC DHCP server is /etc/dhcp/dhcpd.conf. It is always a good practice to backup original configuration files. In case if something goes wrong, the original configuration can easily be restored. We can use the cp command or mv command to create a backup.

$sudo mv /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.backup

Now we can use the editor of our choice to edit dhcpd.conf file.

$sudo gedit /etc/dhcp/dhcpd.conf

Assigning Random IP Addresses from a pool

The most common thing done by a DHCP server is to assign addresses randomly. The settings for the same would be as follows:

```
# a simple /etc/dhcp/dhcpd.conf
default-lease-time 600;
max-lease-time 7200;
authoritative;
subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.100 192.168.1.200;
 option routers 192.168.1.254;
 option domain-name-servers 192.168.1.1, 192.168.1.2;
#option domain-name "mydomain.example";
}
```

According to this configuration:

- The default lease time for a client is 10 mins (600 seconds) and the maximum lease time is 2 hrs (7200 seconds).
- This DHCP Server is the official server for the local network. (indicated by authoritative).
- The Server will hand over the IP Address from the range 192.168.1.100

to 192.168.1.200.

- The server will also "advise" the client to use 192.168.1.254 as the default-gateway and 192.168.1.1 and 192.168.1.2 as its DNS servers.

Assigning Static IP Address to a client

The DHCP Server will need the MAC Address of that client to guarantee that a particular client will always get the same IP Address, this would be the Manual Allocation mode. To get the MAC Address of a client, we can use the ip command on the client machine:

$ ip a

```
saurabh@saurabh-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:a8:d5:fc brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 84310sec preferred_lft 84310sec
    inet6 fe80::ff62:6fe1:63bf:606e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Underline address is the MAC address of the ethernet interface. We need to select the MAC Address of the interface from which we are planning to connect to the network. Here we have only one interface and we will choose that one. Now, we have to put MAC Address in the configuration file:

```
host archmachine {
hardware ethernet 08:00:27:a8:d5:fc;
fixed-address 10.0.2.15;
}
```

This will reserve the IP Address 10.0.2.15 for the client with the MAC Address. 08:00:27:a8:d5:fc

Bind the DHCP Server to an interface

The DHCP Server can have many interfaces. We need to define the interface it should listen to. We can view the interfaces on the server using the command ip a just like the client. The interface to bind with is

defined in the /etc/default/isc-dhcp-server file. Open it using an editor. The interface for our server is eth0. It may be different for another server. Edit the file as:

INTERFACESv4="eth0"

Now that changes to the configuration are made, we need to restart the service to enable those changes. To do that, we will use the systemctl command:

sudo systemctl restart isc-dhcp-server.service

Check the status of DHCP server using the systemctl command. An active status indicates that the DHCP Server has successfully picked up the configuration and is ready to hand out IP Addresses.

sudo systemctl status isc-dhcp-server.service

14.10 IP TABLES AND FIREWALL

Simply put, iptables is a firewall program for Linux. It will monitor traffic from and to our server using tables. These tables contain sets of rules, called chains that will filter incoming and outgoing data packets.

Iptables filters packets based on:

- **Tables:** Tables are files that join similar actions. A table consists of several chains.
- **Chains:** A chain is a string of rules. When a packet is received, iptables finds the appropriate table, then runs it through the chain of rules until it finds a match.
- **Rules:** A rule is a statement that tells the system what to do with a packet. Rules can block one type of packet, or forward another type of packet. The outcome, where a packet is sent, is called a target.
- **Targets:** A target is a decision of what to do with a packet. Typically, this is to accept it, drop it, or reject it (which sends an error back to the sender).

Tables and Chains

Linux firewall iptables has four default tables. We will list all four along with the chains each table contains.

- **Filter**

The Filter table is the most frequently used one. It acts as a bouncer, deciding who gets in and out of your network. It has the following default chains:

- Input - the rules in this chain control the packets received by the server.
- Output - this chain controls the packets for outbound traffic.
- Forward - this set of rules controls the packets that are routed through the server.

- **Network Address Translation (NAT)**

This table contains NAT (Network Address Translation) rules for routing packets to networks that cannot be accessed directly. When the destination or source of the packet has to be altered, the NAT table is used. It includes the following chains:

- Prerouting - this chain assigns packets as soon as the server receives them.
- Output - works the same as the output chain we described in the filter table.
- Postrouting - the rules in this chain allow making changes to packets after they leave the output chain.

- **Mangle**

The Mangle table adjusts the IP header properties of packets. The table has all the following chains we described above: Prerouting , Postrouting ,Output ,Input ,Forward.

- **Raw**

The Raw table is used to exempt packets from connection tracking. The raw table has two of the chains we previously mentioned: Prerouting and Output.

- **Security (Optional)**

Some versions of Linux also use a Security table to manage special access rules. This table includes input, output, and forward chains, much like the filter table.

Targets

A target is what happens after a packet matches a rule criterion.

Non-terminating targets keep matching the packets against rules in a chain even when the packet matches a rule. With terminating targets, a packet is evaluated immediately and is not matched against another chain. The terminating targets in Linux iptables are:

- Accept - this rule accepts the packets to come through the iptables firewall.
- Drop - the dropped package is not matched against any further chain. When Linux iptables drop an incoming connection to the server, the person trying to connect does not receive an error. It appears as if they are trying to connect to a non-existing machine.
- Return - this rule sends the packet back to the originating chain so you can match it against other rules.
- Reject - the iptables firewall rejects a packet and sends an error to the connecting device.

Installing Iptables

Iptables comes pre-installed in most Linux distributions. However, if we don't have it in Ubuntu/Debian system by default, use the following commands :

sudo apt-get update

sudo apt-get install iptables

Check the status of your current iptables configuration by running:

sudo iptables -L -v

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
```

Here, the -L option is used to list all the rules, and -v is for showing the info in a more detailed format. Below is the example output:

Now we have the Linux firewall installed. At this point, we can notice that all chains are set to ACCEPT and have no rules. This is not secure since any packet can come through without filtering.

Defining Chain Rules

Defining a rule means appending it to the chain. To do this, we need to insert the -A

option (Append) right after the iptables command, like:

sudo iptables -A

It will alert iptables that we are adding new rules to a chain. Then, we can combine the command with other options, such as:

- **-i (interface) -** the network interface whose traffic we want to filter, such as eth0, lo, ppp0, etc.
- **-p (protocol) -** the network protocol where our filtering process takes place. It can be either tcp, udp, udplite, icmp, sctp, icmpv6, and so on. Alternatively, we can type all to choose every protocol.
- **-s (source) -** the address from which traffic comes from. We can add a hostname or IP address.
- -**dport (destination port) -** the destination port number of a protocol, such as 22 (SSH), 443 (https), etc.
- **-j (target) -** the target name (ACCEPT, DROP, RETURN). We need to insert this every time we make a new rule.

To use all of them, we must write the command in this order:

sudo iptables -A <chain> -i <interface> -p <protocol (tcp/udp) > -s <source> --dport <port no.> -j <target>

.

Enabling Traffic on Localhost

To allow traffic on localhost, type this command:

sudo iptables -A INPUT -i lo -j ACCEPT

For this iptables, we use lo or loopback interface. It is utilized for all communications on the localhost. The command above will make sure that the connections between a database and a web application on the same machine are working properly.

Enabling Connections on HTTP, SSH, and SSL Port

Next, we want http (port 80), https (port 443), and ssh (port 22) connections to work as usual. To do this, we need to specify the protocol (-p) and the corresponding port

(-dport). We can execute these commands one by one:

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

Filtering Packets Based on Source

Iptables allows to filter packets based on an IP address or a range of IP addresses. We need to specify it after the -s option. For example, to accept packets from 192.168.1.3, the command would be:

sudo iptables -A INPUT -s 192.168.1.3 -j ACCEPT

We can also reject packets from a specific IP address by replacing the ACCEPT target with DROP.

sudo iptables -A INPUT -s 192.168.1.3 -j DROP

If we want to drop packets from a range of IP addresses, we have to use the -m option and iprange module. Then, specify the IP address range with -src-range, a hyphen should separate the range of ip addresses without space, like this:

sudo iptables -A INPUT -m iprange --src-range 192.168.1.100-192.168.1.200 -j DROP

Dropping all Other Traffic

It is crucial to use the DROP target for all other traffic after defining -dport rules. This will prevent an unauthorized connection from accessing the server via other open ports.:

sudo iptables -A INPUT -j DROP

Now, the connection outside the specified port will be dropped.

Deleting Rules

If we want to remove all rules and start with a clean slate, we can use the -F option (flush):

sudo iptables -F

This command removes all current rules. However, to delete a specific rule, we must use the -D option.

Persisting Changes

The iptables rules that we have created are saved in memory. That means we have to redefine them on reboot. To make these changes

persistent after restarting the server, we can use this command:

sudo /sbin/iptables-save

It will save the current rules on the system configuration file, which will be used to reconfigure the tables every time the server reboots.



CHECK YOUR PROGRESS

Q 1: Which protocol is used by Network File System(NFS) to provide connection between client and server?

Q 2: What is NIS? Mention its uses.

Q 3: Fill in the blanks:

i) Computers that run DNS are called _____servers.

ii) The most common program used for maintaining a name server on Linux is _____

iii) _____ is a protocol for downloading files from a remote or local server, or uploading files onto the server.

iv) To improve the web server's performance by caching repeated requests, filter web traffic, and access geo-restricted content, _____ server can be used.

v) The squid service can be configured by editing the _____file.



14.11 LET US SUM UP

- Network File System (NFS) is a way of mounting Linux directories over a network. It was basically developed to share files and folders between two Linux systems faster and better.
- FTP stands for "File Transfer Protocol" and is a great protocol for downloading files from a remote or local server, or uploading files onto the server.
- iptables is a firewall program for Linux.
- NAT table contains NAT (Network Address Translation) rules for routing packets to networks that cannot be accessed directly.

14.12 FURTHER READINGS

1) Linux Administration Cookbook By Adam K. Dean
2) Linux Administration By Jason Cannon

14.13 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: NFS uses Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) to provide a stateless connection between client and server.

Ans to Q No 2: Network Information Service (NIS) is a distributed naming service based on Remote Procedure Call (RPC). It enables easy sharing of various information across the cluster including username, password, hosts, and service ports.

Ans to Q No 3:

i. Name
ii. Berkley Internet Naming Daemon (BIND)
iii. File Transfer Protocol (FTP)
iv. Squid Proxy
v. /etc/squid/squid.conf

14.14 MODEL QUESTIONS

Q 1: What is NFS service? Why We use them?

Q 2: Explain the basic configurations for FTP server?

Q 3: Explain how MAC address are bind to a specific IP address.

Q 4: Construct a rule that disallow any traffic from 192.168.53.10 ip address at port 198.

Q 5: Give a brief overview of NIS architecture.

Q 6: How to change default FTP Ports and directory ?

*** ***** ***

UNIT 15: NETWORK SECURITY

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 Computer Security

15.4 Network Security

15.5 Network Security Threats

15.6 Threat Prevention Techniques/ Solutions

- 15.6.1 Working of Network Security
- 15.6.2 Types of Network Security

15.7 Packet Sniffer

15.8 DoS Attack

15.9 Firewall

- 15.9.1 Types of Firewall
- 15.9.2 IPTABLES

15.10 Virtual Private Network

15.11 Let Us Sum Up

15.12 Further Readings

15.13 Answers To Check Your Progress

15.14 Model Questions

15.1 LEARNING OBJECTIVES

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

- understand the Issues related to network security
- learn different threats in network security
- learn how to prevent threats to network
- understand packet sniffer
- describe the concept of firewall and virtual private network

15.2 INTRODUCTION

In the previous unit, we have learnt how to configure the NFS, NIS,

DNS, FTP, squid proxy, and DHCP server. In this unit, we will discuss about network security issues and threat prevention techniques and solutions.

15.3 COMPUTER SECURITY

Computer Security refers to the process of detecting and preventing unauthorized access of computer. Security involves the process of protecting the data and information of the devices from intruders.

Types of Computer Security

- Information security: Information security refers to securing information from unauthorized access, modification & deletion.
- Application Security: Application security refers to securing different application in the system by taking appropriate measures at the time of designing the software.
- Network Security: Network security refers to securing the devices from different network attacks and keeps safe both software and hardware in the devices from intruders.
- Cyber Security: Cyber security refers to protecting of computer systems, which communicate over the computer networks.

15.4 NETWORK SECURITY

Network Security refers to the measures taken by an organization to secure their computer network and data by using both hardware and software systems. It aims to secure the confidentiality and accessibility of data and network. Every organization that handles large amount of data, has adopted different mechanism to prevent their network from many cyber threats.

The most basic example of Network Security is password protection which is set by the user itself. The network security solutions protect various vulnerabilities of the computer systems such as:

- data loss (deletion of data or encryption of data)
- security breaches
- malicious attacks, such as hacking and viruses

15.5 NETWORK SECURITY THREATS

Network security threats are the possible ways by which it can possibly hamper the normal functioning of your computer. Currently, network security threats are constantly increasing as the whole world is moving for digitization. New threats are arising day by day and attackers use new methods every day to break the securities. The most harmful types of computer security are:

Virus: A computer virus is a malicious program which is inserted into the computer without the knowledge of the user. These are pieces of software designed to spread from one computer device to another. It also replicates itself and infects the files and programs on the computer. Typically, they are automatically downloaded when users browse on the internet downloaded as an attachment with some other downloads or sent as e-mail attachments with the intent of infecting the host computer as well as other computers via network or by data transfer devices such as pen drive, CD etc. Virus can deactivate the security settings, send spam, steal and corrupt data in the computer. Further, it can delete every single thing on the hard drive. The ultimate goal of a virus is to ensure that the victim computer will never be able to operate properly.

Worm: A computer worm is a software program just like virus which can copy itself from one computer to another, without user interaction. The potential risk with the worm is that it will occupy the computer hard disk space because it can replicate in great volume and great speed.

Phishing: This is one type of online fraud, which is designed to steal sensitive information of users, such as bank account detail like credit card numbers, debit card numbers, ATM pins and passwords, OTPs, personal contacts etc. from users. In phishing, the phishers conceal as a trustworthy person or business or reputable banking institutions, websites etc. and attempt to steal sensitive financial or personal information through fraudulent email or instant messages. Phishing is unfortunately very easy to execute. Users may think it's the legitimate mail and enter the personal information, financial details and phishers use those information for fraud.

Sometimes phishers send fraud links on phone messages or WhatsApp or as popup messages in some websites by showing some greedy approach to the users. By clicking those URL or reply to the messages, users are prompted to enter financial details or user credentials, which then sends the data to the malicious source. Phishing attacks are the practice of sending fraudulent communications that appear to come from a reputable source.

Botnet: A botnet is a group of computers connected via internet, that have been compromised to the hacker. Botnet attacks typically involve sending spam, data theft, exploiting sensitive information, or launching vicious DDoS attacks. As different types of devices are in the botnet and grow on increasing, it is very difficult to monitor, detect, and stop botnet attacks.

Botnet attacks typically start when intruders gain access to devices by injecting Trojan viruses or utilizing basic social engineering tactics. Once hackers successfully gain access to these devices, they are brought under control with software that allows them to carry out attacks.

Malware/Ransomware/ Spyware: Malware is malicious software mostly used by hackers to hold control over the system, steal confidential data, or install malicious programs in the device without user knowledge. It spreads spyware, Trojans, and worms through pop-up ads, infected files, bogus websites, or e-mail messages.

On the other hand, ransomware is a type of malware where the attacker locks the victim's device through a virus or phishing e-mails, then asks for ransom to unlock the device. It can hinder the users from running applications, encrypting files, and even completely block the device.

Spyware describes software with malicious behaviour that aims to gather information about a person or organization and sends such information to cyber criminals. Cyber criminals use those information's and hack user's computer or use those information's for financial fraud.

Denial-Of-Service Attack: A denial-of-service attempts to hinder legitimate users from accessing services or information from website. It happens when botnet or malicious software attacks a website with heavy

traffic. It is carried out by one computer and its internet connection, which may enable the intruder to access user's credentials. A distributed denial-of-service is similar to the denial-of-service but is harder to overcome. This is because it is launched from different computers that are distributed all over the globe. The Distributed DoS attack will send multiple requests to the attacked web resource, with the aim of exceeding the website's capacity to handle multiple requests and prevent the website from functioning correctly.

Rogue Security Software: This type of malicious software cheats users by making them to believe that their security measures are not up-to-the-minute or their computer has a virus. They do help by installing software or by updating user's security settings by asking users to pay for the tool or download their program to keep away with the alleged viruses. This can lead to the installation of actual malware in the device.

15.6 THREAT PREVENTION TECHNIQUES/ SOLUTIONS

Typical preventive measures that can be adopted by any enterprise or individual to avoid network security threats include:

- Secure the computer physically by installing reliable, reputable security and anti-virus software and activating your firewall.
- Stay up-to-date on the latest software and news surrounding your devices and perform software updates as soon as they become available.
- Avoid clicking on email attachments unless the source is genuine and known.
- Change passwords on regular basis, using a unique combination of numbers, letters and upper case letters.
- Use the internet with caution and ignore pop-ups, drive-by downloads while surfing.
- Taking the time to research the basic aspects of computer security and educate yourself on evolving cyber-threats.
- Perform full system scans daily and create a periodic system backup

schedule to ensure that your data is retrievable.

- Set the security settings as per latest norms in the router or the system.
- Regular updating of firmware and software.
- Use data encryption technique to store sensitive data.
- Regular backup of data in secondary storage devices not attached to the computer.
- Restricting access of network infrastructure to authorised personnel only.
- Train staff to safe and secure use of the equipment.

As well as training staff, we should also implement policies and rules for computer use in the workplace.

15.6.1 Working of Network Security

The basic principle of network security is to protect data in the computer as well as in the cloud and networks communicating information or data as per the rules and regulations that have to be acknowledged before performing any activity on the data. There are basically three types of network security-

Physical Network Security: Physical network security is the most basic level security that includes protecting the data and network from any unauthorized access to break the confidentiality of the network communication. Physical network security incudes securing of external peripherals, routers and other devices connected in the network communication.

Technical Network Security: This network security is basically to protect the data stored in the network or data involved in transitions through the network. Technical network security serves two purposes: protects from unauthorized access and protects from malicious activities.

Administrative Network Security: Administrative network security deals with the permission and authorization given to the users and guests. This also ensures the level of sophistication the

network might need for protecting it from the attacks. This level also suggests necessary amendments that have to be done to the infrastructure.

15.6.2 Types of Network Security

Few network securities are discussed in the following:

Access Control: All the users in the system or in the network should not have the complete accessibility to the network or its data. This is done through Network Access Control which ensures that only a handful of authorized personnel must be able to work with the allowed number of resources. Not all the users have same authorization.

Antivirus and Anti-malware Software: Antivirus software also known as anti-malware is a computer program used to prevent, detect, and remove malware. Antivirus software was originally developed to detect and remove computer viruses. This type of network security ensures that no malicious software enters to the network and expose the security of the data. The malicious software like Viruses, Trojans, and Worms is handled by the Anti-virus software's.

Cloud Security: Now a days, a lot many organizations are joining hands with cloud technology where a large amount of important data is stored over the internet. Cloud is very vulnerable to the hackers. These data must be protected and it should be ensured that this protection is not compromised at any cost.

Computer Firewall: A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted network and an untrusted network over the Internet.

15.7 PACKET SNIFFER

For monitoring traffic in the network, packet sniffer is used. Packet sniffer can be a software application or hardware tool. The data packets transferred from one computer to another computer in a network is examined by sniffer. Packet sniffer is also known as a packet analyser, protocol analyser or network analyser. Sniffers work by examining streams of data packets that flow between computers on a network as well as between networked computers and the larger Internet. Each of the network packets are intended to a target machine. It contains the address of the target machine. A packet sniffer is in "promiscuous mode" and allows IT professionals, end users or malicious intruders to examine any packet in the network, regardless of destination. Packet sniffers are configured in two ways, "unfiltered," and "filtered". In unfiltered mode the sniffer captures all the possible packets and write them to the local hard drive for examination at a later state. In "filtered" mode sniffers will capture packets that contain some specific data elements as set in the filter.

Packet sniffing is the practice of gathering, collecting, and logging some or all packets that pass in a computer network, regardless of how the packet is addressed. In this way, every packet, or a defined subset of packets, may be gathered for further analysis. A network administrator can use the collected data for a wide variety of purposes like monitoring bandwidth and traffic. Packet sniffers work on both wired and wireless networks, the efficacy of packet sniffers depends on how much they are able to "see" as a result of network security protocols. On a wired network, sniffers might have access to the packets of every connected machine or may be limited by the switches or routers. On a wireless network, most sniffers can only scan one channel at a time. If we use multiple wireless interfaces, it can expand the capability.

POPULAR PACKET SNIFFING TOOLS IN LINUX

- **tcpdump:** It is a popular free and open-source packet analyser. tcpdump uses libpcap library to capture packets on OSI layer 2 and above. It supports regular expressions for packet filtering, and allows one to export

and import packet dumps to and from a file. Using command line interface, tcpdump provides us the packet contents in desired format.

- **Wireshark:** Wireshark is an open-source packet analyser with graphical front end. It captures network traffic from local area networks like Ethernet, Bluetooth, Token Ring, IEEE 802.11 and can store and analyse the data in offline. Wireshark is quite similar to tcpdump though it is more flexible than tcpdump. Wireshark is available on multiple platforms like Linux, BSD, and Mac OS and MS Windows.
- **dhcpdump:** Snoops on DHCP request/response traffic is captured by libpcap library, and displays the DHCP packets in human readable formats. dhcpdump is useful for DHCP debugging and troubleshooting.
- **mysql-sniffer:** An open source MySQL sniffer tool captures live packets sent to a MySQL server, and dumps MySQL queries in a human readable format.

15.8 DoS ATTACK

A denial-of-service (DoS) attack is a cyber-attack on devices, information systems, or other network resources. It prevents authentic users from accessing expected services and resources. A DoS attack is accomplished by flooding the targeted host or network with traffic until the target stops responding or crashes. DoS attacks can launch anywhere from a few hours to many months and can cost institutions or businesses time and money by making their resources and services unavailable.

Working of Denial-of-Service Attacks

The primary focus of a DoS attack is to oversaturate the capacity of a targeted machine, resulting in denial-of-service to additional requests. DoS attack can be categorized mainly into two forms.

- **Buffer overflow attacks -** This is the most common DoS attack. The buffer overflow attack sends more and more packets to a network address so that system cannot able to handle those requests. As a result of this attack, memory buffer overflows and causes the machine to consume all available hard disk space, memory, or CPU time. This

form of exploit often results in sluggish behaviour, system crashes, or other deleterious server behaviours, resulting in denial-of-service.

- **Flood attacks -** By saturating a targeted server with an overwhelming number of packets, a malicious actor is able to oversaturate server capacity, resulting in denial-of-service. In order to a DoS flood attacks to be successful, the malicious machine must have more bandwidth than the target machine. Flood attack can be of one of two types-

ICMP flood - effects misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic.

SYN flood - sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

DoS attacks are on the rise as businesses and consumers use more digital platforms to communicate and transact with each other. In a DoS attack, the cyber-attacks typically use one Internet connection and one device to send rapid and continuous requests to a target server to overload the server's bandwidth. DoS attackers exploit a software vulnerability in the system and proceed to exhaust the RAM or CPU of the server.

Distributed Denial-of-Service (DDoS) Attack

A common type of DoS attack is the distributed denial-of-service (DDoS) attack. The attacker floods its target with numerous amounts of internet traffic so that normal traffic does not able to reach the intended destination. This attack is launched from many compromised connected devices (e.g., smartphones, PCs, network servers, and Internet of Things devices) from around the world go after a targeted website, network, web application, application programming interface, or data centre infrastructure simultaneously to block traffic.

In most of the time, sources of attack are operated form of a botnet. A botnet is a network of personal devices that have been compromised to cybercriminals without the knowledge of the owners. For creating botnet,

the hackers infect the devices with malicious software and gain control of the system to send spam and fake requests to other devices and servers. A target server that attacks by DDoS attack will experience an overload due to hundreds or thousands of network packets come to it. Since in DDoS attack, the victim is attacked from multiple sources, detecting it becomes very difficult. Separating legitimate traffic from fake traffic also becomes impossible, hence it is harder for the victim server to withstand to a DDoS attack.

15.9 FIREWALL

A firewall is a network security device that monitors incoming and outgoing network traffic and permits or blocks data packets based on a set of security rules. The basic purpose of firewall is to establish a barrier between host computer on the network and the network traffic from external sources in order to block malicious traffic like viruses, trojan, worm etc.

A Linux firewall is a solution or service that regulates, protects, and blocks network traffic as it passes to and from a Linux-based environment. Most of the Linux distributions came with In-built firewall services of their own. There are two types of Linux firewall:

- **A command line or GUI utility:** Linux firewall utilities sit on top of pre-built firewall services such as Netfilter, UFW, FirewallD, iptables, etc. We can configure these manually or install an additional utility that reveals the service's full functionalities, simplified configurations and enables point-and-click setup. The pre-built firewall imposes some default firewall zones, like a trusted zone, a demilitarized zone, or a block zone. The utility lets us configure these zones further, set up custom zones, and enforce more granular policies as per our needs.
- **A standalone Linux firewall solution:** It is a comprehensive firewall solution that exists independent of Netfilter, iptables, etc. This comes within a secure, hardened OS that we can install in a shell of our choice - a public cloud environment, or a private virtualized shell. These solutions usually include network management capabilities like traffic

routing or monitoring reports to enable a 360-degree network management landscape.

Both types of Linux firewall solutions can coexist in the same organization. A rule of thumb is to use the first one for solo deployments, while the latter is more suited to enterprise.

Firewalls are one of the core components of a network security implementation. Several vendors in market sells firewall solutions catering to all levels of the marketplace: from home users to data centre solutions. Firewalls can also be a stand-alone hardware solution, such firewall appliances are manufactured by Cisco, Nokia, and Sonicwall. Vendors such as Checkpoint, McAfee, and Symantec have developed proprietary software firewall solutions for home and business markets.

15.9.1 Types of Firewall

- **Packet Filtering Firewall:** One of the most fundamental types of firewalls is packet filtering firewall. The main function of this firewall is to monitor traffic and filter of packets based on security rules configured earlier. If any certain IP address or port number violates the security rules, it immediately blocks it. The main advantages of packet filtering firewall include its inexpensiveness, fastness and efficient.
- **Circuit Level Gateways:** This firewall performs at session layer of OSI model. It does not verify the actual content like packet filtering firewall. It only focuses on TCP connections and sessions. If certain packet follows the valid TCP connection, then it can pass through the gateway irrespective of its content.
- **Application Level Gateways (Proxy Firewall):** Application level gateways or proxy firewall performs at application layer of OSI model. It acts as an intermediary filter between two end systems. Proxy firewall protects the internal network by not letting it direct connect with the outside internet. It verifies validity of a packet based on particular standard protocol (e.g. FTP, HTTP) through

which the packet is sent. If it does not match, it drops the packet.

15.9.2 IPTABLES

Iptables is a firewall, installed by default on all official Ubuntu distributions. After fresh installation of ubuntu iptables allows all traffic by default. Ubuntu comes with ufw - a program for managing the iptables firewall easily. Following command lists the current rules in iptables. After fresh installation of server iptables -L shows nothing.

sudo iptables -L

Basic Iptables Options

Some of the Iptables options are:

- -A - Append this rule to a rule chain. Valid chains for what we're doing are INPUT, FORWARD and OUTPUT, but we mostly deal with INPUT in this tutorial, which affects only incoming traffic.
- -L - List the current filter rules.
- -m conntrack - Allow filter rules to match based on connection state. Permits the use of the --ctstate option.
- --ctstate - Define the list of states for the rule to match on. Valid states are:
- NEW - The connection has not yet been seen.
- RELATED - The connection is new, but is related to another connection already permitted.
- ESTABLISHED - The connection is already established.
- INVALID - The traffic couldn't be identified for some reason.
- -m limit - Require the rule to match only a limited number of times. Allows the use of the --limit option. Useful for limiting logging rules.
- --limit - The maximum matching rate, given as a number followed by "/second", "/minute", "/hour", or "/day" depending on how often you want the rule to match. If this option is not used and -m limit is used, the default is "3/hour".
- -p - The connection protocol used.

- --dport - The destination port(s) required for this rule. A single port may be given, or a range may be given as start:end, which will match all ports from start to end, inclusive.
- -j - Jump to the specified target. By default, iptables allows four targets:
- ACCEPT - Accept the packet and stop processing rules in this chain.
- REJECT - Reject the packet and notify the sender that we did so, and stop processing rules in this chain.
- DROP - Silently ignore the packet, and stop processing rules in this chain.
- LOG - Log the packet, and continue processing more rules in this chain. Allows the use of the --log-prefix and --log-level options.
- --log-prefix - When logging, put this text before the log message. Use double quotes around the text to use.
- --log-level - Log using the specified syslog level. 7 is a good choice unless you specifically need something else.
- -i - Only match if the packet is coming in on the specified interface.
- -I - Inserts a rule. Takes two options, the chain to insert the rule into, and the rule number it should be.
- -I INPUT 5 would insert the rule into the INPUT chain and make it the 5th rule in the list.
- -v - Display more information in the output. Useful for if you have rules that look similar without using -v.
- -s --source - address[/mask] source specification
- -d --destination - address[/mask] destination specification
- -o --out-interface - output name[+] network interface name ([+] for wildcard)

Allowing Established Sessions: We can allow established sessions to receive traffic by following command

sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

The above rule has no spaces either side of the comma in ESTABLISHED,RELATED

Allowing Incoming Traffic on Specific Ports: We can start blocking traffic as we wish, but if we are working over SSH, we need to allow SSH before blocking everything else.

To allow incoming traffic on the default SSH port (22), we can tell iptables to allow all TCP traffic on that port to come in.

sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT

To block an IP address (say xxx.xxx.xxx.xxx) in Iptables firewall:

iptables -A INPUT -p tcp -s xxx.xxx.xxx.xxx -j DROP

To save changes in iptables:

iptables-save

To view all options available or for help:

iptables -h

15.10 VIRTUAL PRIVATE NETWORK

Virtual Private Network (VPN) allows users to connect computers to a private network, creating an encrypted connection that masks the computers connecting to the network securely share data and surf on the web, protecting the identity of the user.

A virtual private network, or VPN, is an encrypted connection over the Internet from a device to a network. The encrypted connection helps to ensure that sensitive data is safely transmitted. It prevents unauthorized people from eavesdropping on the traffic and allows the user to conduct work remotely. VPN technology is widely used in corporate environments. Virtual private network is mainly of two types:

Remote Access VPN: Through remote access VPN, users can connect to private network and able to access all resources and services remotely. It is used by both individual users and business organisation. For

example, during work from home service by an employee for an organisation, the employee can take the help of VPN to access the private network of his company securely to access all the resources of the company and perform his work.

Router-to-Router VPN: Router-to-router VPN is also known as site-to-site VPN which is commonly used in large organisations. An organisation or multiple organisations, which have many branches at different locations of a state or country is connected over their own private network using site-to-site VPN. It can be of two types:

- **Intranet based VPN:** A site-to-site Virtual Private Network through which many branches of the same organization are connected, is known as Intranet based VPN.
- **Extranet based VPN:** A site-to-site Virtual Private Network through which a branch of an organization is connected to branch or office of another organization, then it is known as extranet-based VPN.



CHECK YOUR PROGRESS

Q 1: In computing _____ is a network security system that monitors and controls network traffic. (spyware / Cookie / Spam / Firewall)

Q 2: A computer virus is a _____ . (Program/ Hardware / freeware)

Q 3: DDoS stands for _____.

Q 4: Iptables use _____ . (for IP address tracking / to find a IP Address / as firewall)

Q 5: _____ is a popular packet sniffing toll. (Wireshark/ IP capture / Iptables)



15.11 LET US SUM UP

- Computer Security refers to the process of detecting and preventing unauthorized access of computer. Security involves the process of safeguarding the devices data and information from intruders.

- Network Security refers to the measures taken by an organization to secure their computer network and data using both hardware and software systems. Network security aims to secure the confidentiality and accessibility of data and network.
- Basic network security threats are virus, worm, phishing, botnet, malware, DoS attack
- The few types of network securities that one can adopt are Access Control, Antivirus, Cloud Security, Computer Firewall.
- Packet sniffing is the practice of gathering, collecting, and logging some or all packets that pass in a computer network, regardless of how the packet is addressed.
- Popular packet sniffing tools are tcpdump, Wireshark, dhcpdump, mysql-sniffer
- A denial-of-service (DoS) attack is a cyberattack on devices, information systems, or other network resources that prevents authentic users from accessing expected services and resources.
- DoS attack can be categorized mainly two ways, Buffer overflow attacks and Flooding attacks.
- A firewall is a network security device that monitors incoming and outgoing network traffic and permits or blocks data packets based on a set of security rules.
- Iptables is a firewall, installed by default on all official Ubuntu distributions.
- Virtual Private Network (VPN) allows users to connect computers to a private network, creating an encrypted connection that masks the computers connecting to the network securely share data and surf on the web, protecting the identity of the user.



15.12 FURTHER READINGS

1) Linux System Administration by Tom Adelstein, Bill Lubanovic
2) Linux Network Administrators Guide by Olaf Kirch
3) Linux security cookbook by Daniel J. Barrett

15.13 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: Firewall

Ans to Q No 2: Program

Ans to Q No 3: Distributed Denial of Service

Ans to Q No 4: as firewall

Ans to Q No 5: Wireshark

15.14 MODEL QUESTIONS

Q 1: What is network security? What are the threats in network security?

Q 2: Explain the different types of network security.

Q 3: What is firewall? Explain its working.

Q 4: What is virtual private network? Explain briefly.

Q 5: What is DoS attack? How DoS attacks are performed? Explain in detail.

Q 6: What is packet sniffing? Explain the different packet sniffing tools.

*** ***** ***