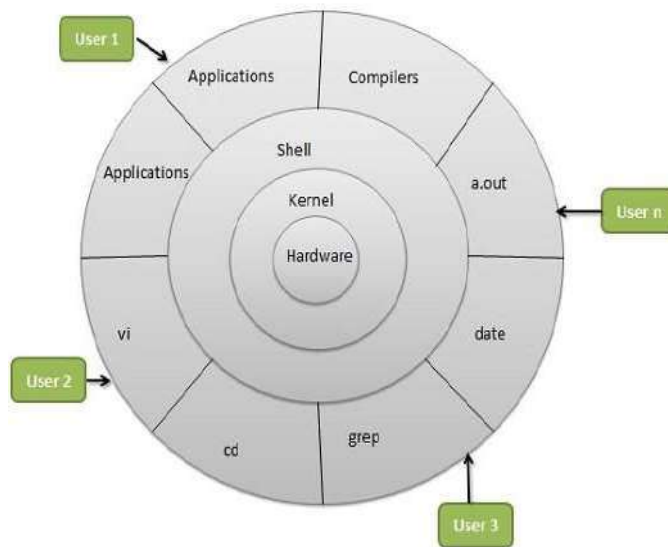


Linux System Administration



**VARDHMAN MAHAVEER OPEN UNIVERSITY
KOTA**

MCA-305



Vardhman Mahaveer Open University, Kota

Linux System Administration

Course Development Committee

Chair Person

Prof. Ashok Sharma

Vice-Chancellor

Vardhman Mahaveer Open University, Kota

Prof. L.R. Gurjar

Director Academic

Vardhman Mahaveer Open University, Kota

Convener and Members

Convener

Neeraj Arora

Assistant Professor, Computer Science

School of Science and Technology,

Vardhman Mahaveer Open University, Kota.

Members

1. Prof. (Dr.) Reena Dadich

Professor and Head (CS)

University of Kota, Kota

2. Prof. (Dr.) N.K. Joshi

Professor (CS) and Director, MIMT, Kota

3. Dr. Harish Sharma

Associate Professor, CSE Deptt.

Rajasthan Technical University, Kota

4. Mr. Abhishek Nagar,

Programmer Officer, VMOU, Kota

5. Dr. Anuradha Dubey

Deputy Director

School of Science & Technology

Vardhman Mahaveer Open University, Kota

Editor**Prof. (Dr.) Reena Dadich**Professor and Head, Dept. of CS and Informatics,
University of Kota, Kota***Unit Writers*****Mr. Mudit Chaturvedi**

Assistant Professor, Computer Science and Engineering, Jaipur National University, Jaipur

Mr. Om Prakash Suthar

Asst. Prof. (SR. Scale), Department of CSE, J.I.E.T., Jodhpur.

Mr. Vinay Mathur

Associate Professor, CSE Department, J.I.E.T., Jodhpur.

Mr. Poonam Chand Shukla

Jr. Technical Superintendent, Dept. of CSE, IIT Jodhpur.

Mr. Amresh Kumar Singh

Assistant Prof, Dept of CS, MGSU Bikaner

Mr. Nitin Mathur

Assistant Professor, Dept. Of CSE, J.I.E.T., Jodhpur.

Mr. Vishnu Sharma

Asst. Prof. , School of Computer and System Sciences, Jaipur National University, Jaipur

Units**1, 2****3****4, 9****5****6, 11****7, 8, 10, 12, 13, 14****15, 16**

Academic and Administrative Management

Prof. Ashok Sharma

Vice-Chancellor,

Vardhman Mahaveer Open University, Kota

Dr. Shiv Kumar Mishra

Director (MP&D)

Vardhman Mahaveer Open University, Kota

Prof. L.R. Gurjar

Director (Academic)

Vardhman Mahaveer Open University, Kota



Vardhman Mahaveer Open University, Kota

Linux System Administration

Contents

<i>Unit No.</i>	<i>Unit</i>	<i>Pages</i>
<i>Unit-1</i>	<i>Introduction:</i> Introduction to Linux, Basic idea on proprietary, Open Source, Free Software etc..	1-20
<i>Unit-2</i>	<i>Variants of Linux/GNU:</i> Introduction of Various Linux Distribution (Red Hat Enterprise Linux, Cent OS, Fedora Projects, Debian Linux, Ubuntu, etc.), GNU-Linux connection.	21-43
<i>Unit-3</i>	<i>Architecture:</i> Basic Architecture of Unix/Linux system, kernel, shell.	44-61
<i>Unit-4</i>	<i>File System:</i> Linux File System, Boot block, Super block, Inode table, Data blocks, How Linux access files, storage files, Linux standard directories, LILO, GRUB boot Loader.	62-76
<i>Unit-5</i>	<i>Installation:</i> Installation of Linux system, Using Live CD, Virtual Machine, Direct Installation Partitioning the Hard drive for Linux, init and run levels.	77-93
<i>Unit-6</i>	<i>Getting started with Linux:</i> Login process, Creating Users Account and Group, Getting Help. Services and Process.	94-108
<i>Unit-7</i>	<i>File in Linux:</i> Files and File System, File Types and Permissions, Links, Size and Space, Date and Time.	109-125
<i>Unit-8</i>	<i>Working with file</i> Reading Files, Searching for files, Copying, Moving, Renaming, Deleting, Linking, and Editing Files	126-142
<i>Unit-9</i>	<i>File Commands in Linux:</i> ls, rm, rmdir, pwd, more, less, grep, awk, sort, cat, head, tail, wc, tee, ps, top, tar, unzip, nice, kill, netstat, Disk related commands, checking disk free spaces..	143-162

Unit-10	<i>Programming with shell script:</i>	163-188
	Various types of Shell available in Linux, Comparisons between various Shells, 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-11	<i>System Administration:</i>	189-210
	Common administrative tasks, identifying administrative files – configuration and log files, Managing user accounts-adding & deleting users, changing permissions and ownerships, Creating and managing groups, modifying group attributes.	
Unit-12	<i>User Accounts :</i>	211-230
	Temporary disable user's accounts, creating and mounting file system, checking and monitoring system performance file security & Permissions, becoming super user using su.	
Unit-13	<i>Getting system information:</i>	231-253
	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.	
Unit-14	<i>Introduction to X-windows system:</i>	254-280
	Configure X-windows starting & using X desktop, KDE & Gnome graphical interfaces, changing X windows settings.	
Unit-15	<i>Linux Networking:</i>	281-304
	Installation and configuration of a simple LAN, Installation and configuration of Proxy server(Squid).	
Unit-16	<i>Introduction to Server:</i>	305-333
	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.	

Preface

The present book entitled “Linux System Administration” has been designed so as to cover the unit-wise syllabus of MCA-305 course for MCA 3rd year students of Vardhman Mahaveer Open University, Kota.

This book introduces the participants to the Linux computing environment or Linux Based Operating System, its architecture, Linux File System and the commands associated with them, programming with shell script, some administrative task in Linux and fundamentals of Linux networking and servers.

Each unit begins with objectives , introduction and principles together with illustrative and other descriptive material .The illustrative examples serve to illustrate and amplify the theory, bring into focus on important concepts .Numerous proofs of the theorems and derivations of the fundamental results are included . The units have been written by various experts in the field. We believe that this book is well suited to self learning. The text is written in a logical sequence and is beneficial for students. The concise and sequential nature of the book makes it easier to learn. Although we have made all efforts to make the text error free, yet errors may remain in the text. We shall be thankful to the students and teachers alike if they point these out to us. Any further comments and suggestions for future improvement are welcome and will be most gratefully acknowledged.

UNIT-1

Introduction to Linux

Structure of the Unit

- 1.0 Objective
- 1.1 Introduction
 - 1.1.1 History About Linux
 - 1.1.2 Linus and Linux
- 1.2 Why to use Linux
- 1.3 Linux Pros
- 1.4 Current application of Linux systems
- 1.5 The user interface
 - 1.5.1 Is Linux difficult?
- 1.6 Linux Cons
- 1.7 Components of Linux System
- 1.8 Properties of Linux
- 1.9 Difference between GUI and Command Line
 - 1.9.1 What is a GUI?
 - 1.9.2 What is a Command Line?
 - 1.9.3 What is the difference between a GUI & a Command Line
- 1.10 Open Source
 - 1.10.1 What is open Source Software?
- 1.11 Free Software
- 1.12 Free Software versus Open Source Software
- 1.13 Self Exercise Question
- 1.14 Summary
- 1.15 Glossary
- 1.16 Answer to Self Exercise Question

1.0 Objective

The objective of this chapter is to introduce participants to the Linux computing environment or Linux Based Operating System. This chapter will cover an introduction about elements of Linux system architecture and computing

philosophy and provide brief about command user interface & graphical user interface, open source and free software followed by their differences.

1.1 Introduction

1.1.1 History about Linux

In order to understand the popularity of Linux, we need to travel back in time, about 30 years ago...

Imagine computers as big as houses, even stadiums. While the sizes of those computers posed substantial problems, there was one thing that made this even worse: every computer had a different operating system. Software was always customized to serve a specific purpose, and software for one given system didn't run on another system. Being able to work with one system didn't automatically mean that you could work with another. It was difficult, both for the users and the system administrators.

Computers were extremely expensive then, and sacrifices had to be made even after the original purchase just to get the users to understand how they worked. The total cost per unit of computing power was enormous.

Technologically the world was not quite that advanced, so they had to live with the size for another decade. In 1969, a team of developers in the Bell laboratories started working on a solution for the software problem, to address these compatibility issues.

They developed a new operating system, which was

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

The Bell Labs developers named their project "UNIX." The code recycling features were very important. Until then, all commercially available computer systems were written in a code specifically developed for one system. UNIX on the other hand needed only a small piece of that special code, which is now commonly named the Kernel. This Kernel is the only piece of code that needs to be adapted for every specific system and forms the base of the UNIX system. The operating system and all other functions were built around this Kernel and written in a higher programming language, C.

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

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

1.1.2 Linus and Linux

Linux is an operating system that was first created at the University of Helsinki in Finland by a young student named Linus Torvalds. At this time the student was working on a UNIX system that was running on an expensive platform. Because of his low budget, and his need to work at home, he decided to create a copy of the UNIX system in order to run it on a less expensive platform, such as an IBM PC. He began his work in 1991 when he released version 0.02 and worked steadily until 1994 when version 1.0 of the Linux Kernel was released. The current full-featured version at this time is 2.2.X (released January 25, 1999), and development continues.

The Linux operating system is developed under the GNU General Public License (also known as GNU GPL) and its source code is freely available to everyone who downloads it via the Internet. The CD-ROM version of Linux is also available in many stores, and companies that provide it will charge you for the cost of the media and support. Linux may be used for a wide variety of purposes including networking, software development, and as an end-user platform. Linux is often considered as an excellent, low-cost alternative to other more expensive operating systems because you can install it on multiple computers without paying more.

1.2 Why to use Linux

There are no royalty or licensing fees for using Linux, and the source code can be modified to fit your needs. The results can be sold for profit, but original authors retain copyright and you must provide the source to your modifications.

Because it comes with source code to the Kernel, it is quite portable. Linux runs on more CPUs and platforms than any other operating system.

The recent direction of the software and hardware industry is to push consumers to purchase faster computers with more system memory and hard drive storage. Linux systems are not affected by those industries' orientation because of its capacity to run on any kind of computers, even aging x486-based computers with limited amounts of RAM.

Linux is a true multi-tasking operating system similar to his brother UNIX. It uses sophisticated, state-of-the-art memory management to control all system processes. That means that if a program crashes you can kill it and continue working with confidence.

Another benefit is that Linux is practically immunized against all kinds of viruses that we find in other operating systems.

viruses that were effective on Linux systems.

1.3 Linux Pros

Linux has several silent features, some of the important ones are

- **Linux is free:**

Linux can be downloaded in its entirety from the Internet completely for free. No registration fees, no costs per user, free updates, and freely available source code in case we want to change the behaviour of your system. Most of all, Linux is free as in free speech.

The license commonly used is the GNU Public License (GPL). The license says that anybody who may want to do so, has the right to change Linux and eventually to redistribute a changed version, on the one condition that the code is still available after redistribution. In practice, you are free to grab a Kernel image, for instance to add support

for tele-transportation machines or time travel and sell your new code, as long as your customers can still have a copy of that code.

- **Linux is portable to any hardware platform:**

A vendor who wants to sell a new type of computer and who doesn't know what kind of OS his new machine will run (say the CPU in your car or washing machine), can take a Linux Kernel and make it work on his hardware, because documentation related to this activity is freely available. Linux was made to keep on running.

As with UNIX, a Linux system expects to run without rebooting all the time. That is why a lot of tasks are being executed at night or scheduled automatically for other calm moments, resulting in higher availability during busier periods and a more balanced use of the hardware. This property allows for Linux to be applicable also in environments where people don't have the time or the possibility to control their systems night and day.

- **Linux is secure and versatile:**

The security model used in Linux is based on the idea of UNIX security, which is known to be robust and of proven quality. But Linux is not only fit for use as a fort against enemy attacks from the Internet: it will adapt equally to other situations, utilizing the same high standards for security. Your development machine or control station will be as secure as your firewall.

- **Linux is scalable:**

From a Palmtop with 2 MB of memory to a petabyte storage cluster with hundreds of nodes: add or remove the appropriate packages and Linux fits all. You don't need a supercomputer anymore, because you can use Linux to do big things using the building blocks provided with the system. If you want to do little things, such as making an operating system for an embedded processor or just recycling your old 486, Linux will do that as well.

- **The Linux OS and most Linux applications have very short debug-times:**

Because Linux has been developed and tested by thousands of people, both errors and people to fix them are usually found rather quickly. It sometimes happens that there are only a couple of hours between discovery and fixing of a bug.

1.4 Current applications of Linux systems

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

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

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

1.5 The user interface

1.5.1 Is Linux difficult?

Whether Linux is difficult to learn depends on the person you're asking. Experienced UNIX users will say no, because Linux is an ideal operating system for power-users and programmers, because it has been and is being developed by such people.

Everything a good programmer can wish for is available: compilers, libraries, development and debugging tools. These packages come with every standard Linux distribution. The C-compiler is included for free - as opposed to many UNIX distributions demanding licensing fees for this tool. All the documentation and manuals are there, and examples are often included to help you get started in no time. It feels like UNIX and switching between UNIX and Linux is a natural thing.

In the early days of Linux, being an expert was kind of required to start using the system. Those who mastered Linux felt better than the rest of the "lusers" who hadn't seen the light yet. It was common practice to tell a beginning user to "RTFM" (read the manuals). While the manuals were on every system, it was difficult to find the documentation, and even if someone did, explanations were in such technical terms that the new user became easily discouraged from learning the system.

The Linux-using community started to realize that if Linux was ever to be an important player on the operating system market, there had to be some serious changes in the accessibility of the system.

1.6 Linux Cons

There are far too many different distributions:

"Quot capites, tot rationes", as the Romans already said: the more people, the more opinions. At first glance, the amount of Linux distributions can be frightening, or ridiculous, depending on your point of view. But it also means that everyone will find what he or she needs. You don't need to be an expert to find a suitable release.

When asked, generally every Linux user will say that the best distribution is the specific version he is using. So which one should you choose? Don't worry too much about that: all releases contain more or less the same set of basic packages. On top of the basics, special third party software is added making, for

example, TurboLinux more suitable for the small and medium enterprise, RedHat for servers and SuSE for workstations. However, the differences are likely to be very superficial. The best strategy is to test a couple of distributions; unfortunately not everybody has the time for this. Luckily, there is plenty of advice on the subject of choosing your Linux. A quick search on Google, using the keywords "choosing your distribution" brings up tens of links to good advice. The Installation HOWTO also discusses choosing your distribution.

- **Linux is not very user friendly:**

It must be said that Linux, at least the core system, is less user friendly to use than MS Windows and certainly more difficult than MacOS, but... In light of its popularity, considerable effort has been made to make Linux even easier to use, especially for new users. More information is being released daily, such as this guide, to help fill the gap for documentation available to users at all levels. Is an Open Source product trustworthy?

How can something that is free can also be reliable? Linux users have the choice whether to use Linux or not, which gives them an enormous advantage compared to users of proprietary software, who don't have that kind of freedom. After long periods of testing, most Linux users come to the conclusion that Linux is not only as good, but in many cases better and faster than the traditional solutions. If Linux were not trustworthy, it would have been long gone, never knowing the popularity it has now, with millions of users. Now users can influence their systems and share their remarks with the community, so the system gets better and better every day. It is a project that is never finished, that is true, but in an ever changing environment, Linux is also a project that continues to strive for perfection.

1.7 Components of Linux System

Linux Operating System has primarily three components-

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

System Library - System libraries are special functions or programs using which application programs or system utilities access Kernel's features. These libraries implement most of the functionalities of the operating system and do not require Kernel module's code access rights.

System Utility - System Utility programs are responsible to do specialized, individual level tasks.

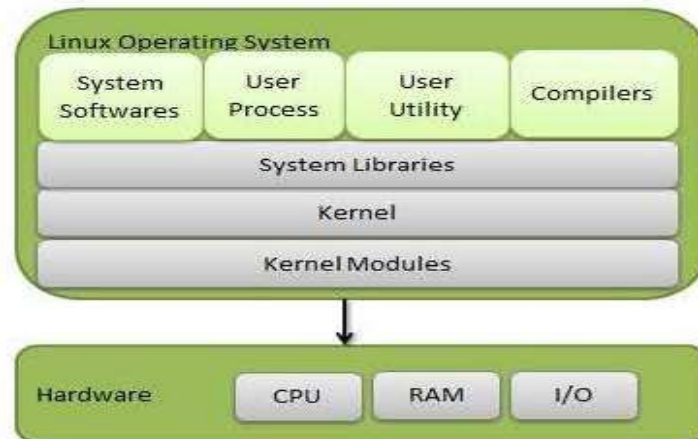


Figure 1.1

Kernel Mode vs User Mode

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

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

1.8 Properties of Linux

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

Portable - Portability means software can work on different types of hardware in same way. Linux Kernel and application programs support their installation on any kind of hardware platform.

Open Source - Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/RAM/application programs at the same time.

Multiprogramming - Linux is a multiprogramming system means multiple applications can run at the same time.

Hierarchical File System - Linux provides a standard file structure in which system files/ user files are arranged.

Shell - Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.

Security - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

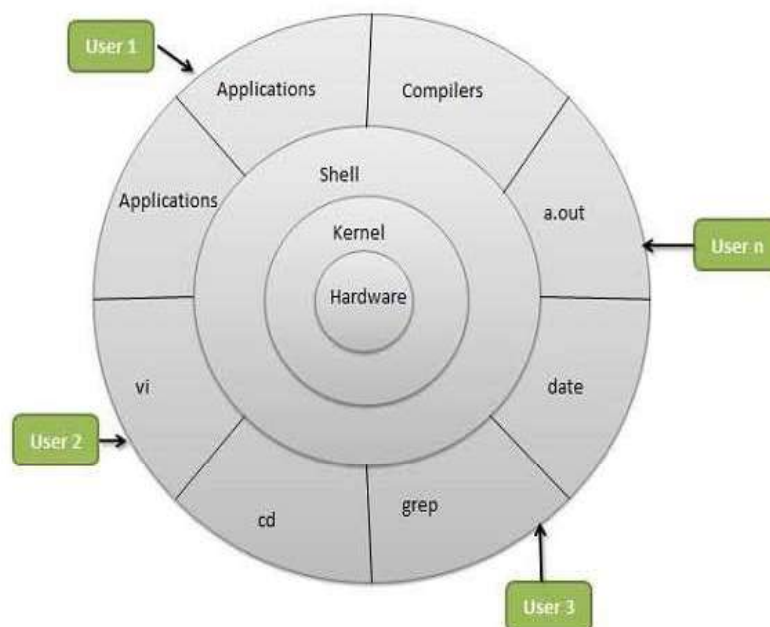


Figure 1.2

Linux System Architecture is consists of following layers

Hardware layer - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).

Kernel - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.

Shell - An interface to Kernel, hiding complexity of Kernel's functions from users. Takes commands from user and executes Kernel's functions.

Utilities - Utility programs giving user most of the functionalities of an operating systems

1.9 Difference Between GUI and Command Line

Two most popular ways to interact with a computer are the Command Line and the GUI (Graphical User Interface). Command line is a text only interface, while GUI is an Interface, composed of graphical symbols. Most of the times, all the general tasks performed using a GUI can be performed by a Command line and vice versa (although when it comes to advanced tasks Command line may be the only option).

1.9.1 What is a GUI?

GUI (pronounced gooey) is a type of interface that allows users to communicate with operating systems in the form of images/animations/audio as opposed to text. This interface presents the user with the information/actions available through graphical objects (like icons). Both mouse and keyboard can be used for the interaction. The user performs actions by directly manipulating graphical objects on the screen.

1.9.2 What is a Command Line?

Command Line (typically known as Command-line interface/interpreter or CLI) is a mechanism that allows interacting with the computer operating system by inputting (typing) commands. This is a text only interface, which only requires input from the keyboard (usually referred to as “entering a command”). Typically, Enter key is pressed at the end of a command, after which the computer will receive, parse and execute that command. The output of the command will be returned back to the terminal as text lines. Output may

include a summary of the task and the actual result as well. To insert command in a batch mode, user can use a script file. A script is a file containing an ordered sequence of commands that will complete an entire job.

1.9.3 What is the difference between a GUI and a Command Line?

Most command line interface tasks only require the keyboard, while GUI systems require both the mouse and the keyboard. Therefore, command line users usually do not have to switch their hands between two places. And command line interface usually requires just few lines of code to perform a complicated task. Command line definitely uses fewer resources than a GUI system as a GUI system will load icons, fonts, I/O drivers and other resources. Because of these three reasons, command line users may be able to complete most tasks relatively faster than a GUI user. Command line users can create scripts and save time, while GUI users can do the same with facilities such as creating shortcuts.

Although new users might have to learn how to operate the mouse, GUI is easier to pickup than using the Command line. Unlike GUI, Command Line users need a fair amount of familiarity and need to memorize a number of commands in order to get their jobs done smoothly. But, a command line user has much more control of the file and operating system. And for performing some advanced tasks, command line may be the only option (sometimes). GUI systems inherently make it easy to multitask, by providing graphical means of monitoring several things (process) at once (many command line environments offer multitasking, but it is harder to view several things at once).

Linux is an open-source Kernel and usually comes bundled with free and open source software; however, proprietary software for Linux does exist and is available to end-users. In information technology, proprietary describes a technology or product that is owned exclusively by a single company that carefully guards knowledge about the technology or the product's inner workings. Some proprietary products can only function properly if at all when used with other products owned by the same company. An example of a proprietary product is Adobe Acrobat , whose Portable Document Format (PDF) files can only be read with the Acrobat Reader. Microsoft is often held up as the best example of a company that takes the proprietary approach. It

should be observed that the proprietary approach is a traditional approach. Throughout history, the knowledge of how an enterprise makes its products has usually been guarded as a valuable secret and such legal devices as the patent, trademark, and copyright were invented to protect a company's intellectual property.

The following is a list of proprietary software for Linux:

1Networking			
1.1Web browsers	1.2FTP clients	1.3Firewall (packet filtering)	1.4Visual traceroute
1.5P2P File Sharing	1.6"Hofline" P2P protocol clients/servers	1.7Multifunction sound modem program/s	1.8Fax software
1.9Network maintenance	1.10Routing	1.11Distributed Computing	1.12Instant messaging
1.13P2P file synchronisation	1.14VoIP Software		

Table: 1.1

2Desktop/System software			
Work with files	Asset Management And Real Property	PDF viewer	PDF Authoring
Cryptography	Scanner utilities	Antivirus	Bootloading
Hard disk partitioning	Backup software	PIM / DB / Hierarchical notebook with tree view	File hosting service clients

Table: 1.2

1.10 Open source

The term "open source" refers to something people can modify and share because its design is publicly accessible.

The term originated in the context of software development to designate a specific approach to creating computer programs. Today, however, "open source" designates a broader set of values—what we call "the open source way." Open source projects, products, or initiatives embrace and celebrate

principles of open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community-oriented development.

1.10.1 What is open source software?

Open source software is software with source code that anyone can inspect, modify, and enhance.

"Source code" is the part of software that most computer users don't ever see; it's the code that computer programmers can manipulate to change how a piece of software—a "program" or "application"—works. Programmers who have access to a computer program's source code can improve that program by adding features to it or fixing parts that don't always work correctly.

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Source code which creates any confusion is not allowed. Intermediate forms such as the output of a pre-processor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination against Fields of Endeavour

The license must not restrict anyone from making use of the program in a specific field of endeavour. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

The following is a list of currently recognized open source licenses:-

- i. Academic Free License 3.0 (AFL-3.0)
- ii. Affero GNU Public License
- iii. Adaptive Public License (APL-1.0)
- iv. Apache License 2.0 (Apache-2.0)
- v. Apple Public Source License (APSL-2.0)
- vi. Artistic license 2.0 (Artistic-2.0)

Many more are there.

1.11 Free Software

“Free software” means software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, “free software” is a matter of liberty, not price.

The four essential freedoms

A program is free software if the program's users have the four essential freedoms:

- a. The freedom to run the program as you wishes, for any purpose.
- b. The freedom to study how the program works and change it so it does your computing as you wishes.
- c. Access to the source code is a precondition for this. The freedom to redistribute copies so you can help your neighbour.
- d. The freedom to distribute copies of your modified versions to others.

By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

A program is free software if it gives users adequately all of these freedoms. Otherwise, it is non-free. While we can distinguish various non-free distribution schemes in terms of how far they fall short of being free, we consider them all equally unethical.

In any given scenario, these freedoms must apply to whatever code we plan to make use of, or lead others to make use of. For instance, consider a program A

which automatically launches a program B to handle some cases. If we plan to distribute A as it stands, that implies users will need B, so we need to judge whether both A and B are free. However, if we plan to modify A so that it doesn't use B, only A needs to be free; B is not pertinent to that plan.

“Free software” does not mean “non-commercial”. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important. You may have paid money to get copies of free software, or you may have obtained copies at no charge. But regardless of how you got your copies, you always have the freedom to copy and change the software, even to sell copies.

1.12 Free Software versus Open Source Software

Although the terms free software and open source software are usually used more or less interchangeably, there are some subtle differences. They arise from differences in their histories, in the philosophies of the groups promoting them and in their secondary meanings.

For example, the term open source is of much more recent origin. It was coined in 1997 with the intention of replacing the term free software in order to avoid the negative connotations that are sometimes associated with the word free and thereby make it more attractive to corporations. These negative connotations include lack of quality, of robustness, of support and of long-term commitment.

In general, there is a tendency for advocates of the term free software to emphasize the ideological aspects of software, including the ethical or moral aspects, and they view technical excellence as both a desirable and an unavoidable by-product of their ethical standards. Advocates of the term open source, in contrast, tend to place more emphasis on the business advantages of the software. They regard technical excellence as the primary goal, and sharing of the source code is seen as a means of achieving that goal. They prefer the term open source as a way of avoiding both the negative connotations and the ambiguity of the English word free (i.e., free price versus freedom of use).

However, although the use of the term open source clearly avoids the problem of the ambiguity of the word free, it introduces another ambiguity. It is the distinction between programs that provide the source code and give users the

freedom to use it for any desired purpose and programs that provide the source code but place restrictions on its use (e.g., do not allow it to be redistributed). Software with such restrictions is not free software as the term is most commonly used.

The terms liberated software and free open source software (FOSS) have been proposed as a means of overcoming the problems with the terms free software and open source software. However, although used occasionally, they have problems of their own and it thus appears unlikely that they will become replacements.

1.13 Self Exercise Question

1. What is Linux?
2. What is the difference between UNIX and LINUX?
3. What is BASH?
4. What is Linux Kernel?

1.14 Summary

In this chapter, we learned that:

Linux is an implementation of UNIX.

- The Linux operating system is written in the C programming language.
- Linux uses GNU tools, a set of freely available standard tools for handling the operating system.
- The Linux operating system is developed under the GNU General Public License (also known as GNU GPL) and its source code is freely available to everyone who downloads it via the Internet.
- Linux is an open-source Kernel and usually comes bundled with free and open source software; however, proprietary software for Linux does exist and is available to end-users.
- Linux is not very user friendly and is confusing for beginners: It must be said that Linux, at least the core system, is less user friendly to use than MS Windows and certainly more difficult than MacOS, but... In light of

its popularity, considerable effort has been made to make Linux even easier to use, especially for new users.

- Multi-User - Linux is a multiuser system means multiple users can access system resources like memory/ RAM/application programs at the same time.
- No registration fees, no costs per user, free updates, and freely available source code in case you want to change the behaviour of your system.
- "Source code" is the part of software that most computer users don't ever see; it's the code computer programmers can manipulate & change the code how a piece of software—a "program" or "application"—works.

1.15 Glossary

GNU:-

The GNU operating system is a complete free software system, upward-compatible with Unix.

GPL:-

General Public Licence

The GPL is a copyleft license, which means that derivative work can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses and the MIT License are widely used examples. GPL was the first copyleft license for general use.

Freeware

Freeware is software that is available for use at no monetary cost. In other words, while freeware may be used without payment it is most often proprietary software, as usually modification, re-distribution or reverse-engineering without the author's permission is prohibited.

Operating System

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs.

All computer programs, excluding firmware, require an operating system to function.

1.16 Answer to Self Exercise Question

Solution of Self Exercise Question

1. Linux is an operating system based on UNIX, and was first introduced by Linus Torvalds. It is based on the Linux Kernel, and can run on different hardware platforms manufactured by Intel, MIPS, HP, IBM, SPARC and Motorola.
2. Unix originally began as a propriety operating system from Bell Laboratories, which later on spawned into different commercial versions. On the other hand, Linux is free, open source and intended as a non-propriety operating system for the masses.
3. BASH is short for Bourne Again SHell. It was written by Steve Bourne as a replacement to the original Bourne Shell (represented by /bin/sh). It combines all the features from the original version of Bourne Shell, plus additional functions to make it easier and more convenient to use. It has since been adapted as the default shell for most systems running Linux.
4. The Linux Kernel is a low-level systems software whose main role is to manage hardware resources for the user. It is also used to provide an interface for user-level interaction.

UNIT-2

Variants of Linux

Structure of the Unit

- 2.0 Objective
- 2.1 Introduction
 - 2.1.1 Operating System
 - 2.1.2 Other Important Activities
- 2.2 Distribution of Linux
- 2.3 RedHat
 - 2.3.1 Introduction
 - 2.3.2 Advantage of RedHat
- 2.4 Centos Linux
 - 2.4.1 Introduction
 - 2.4.2 Advantage of Centos
- 2.5 Fedora
 - 2.5.1 Introduction
 - 2.5.2 Advantage of Fedora
- 2.6 Debian Linux
 - 2.6.1 Introduction
 - 2.6.2 Advantage of Debian Linux
- 2.7 Ubuntu Linux
 - 2.7.1 Introduction
 - 2.7.2 Advantage of Ubuntu
- 2.8 GNU-Linux Connection
- 2.9 Summary
- 2.10 Glossary
- 2.11 Exercise

2.0 Objective

LINUX now firmly taking its rightful place in the market as a viable and robust. There are a growing number of third parties selling their own LINUX variants. This chapter has been written from the outset with different linux distribution in

mind. And try to explain the pros and cons of each LINUX operating system with their features.

2.1 Introduction

Linux is a free and open Source Operating System developed by **Linus Torvalds** and came into existence in the year 1991, at AT&T's Bell Laboratories, released under the GPL (General Public License). As any other OS main function of Linux OS is to manage the system resources (Hardware). We can refer it as an interface between the user and the computer system. GNU Linux has received lots of success and popularity in the last 2 decades with most of the commercial servers now using GNU Linux. Recently home users also started using Linux as their day to day OS, and popular Laptop and PC manufactures also giving GNU Linux as a pre-installed OS on their systems as its free and helps in cutting overall cost.

2.1.1 Operating System

An Operating System (OS) is an interface between the computer user and computer hardware. An operating system is the software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

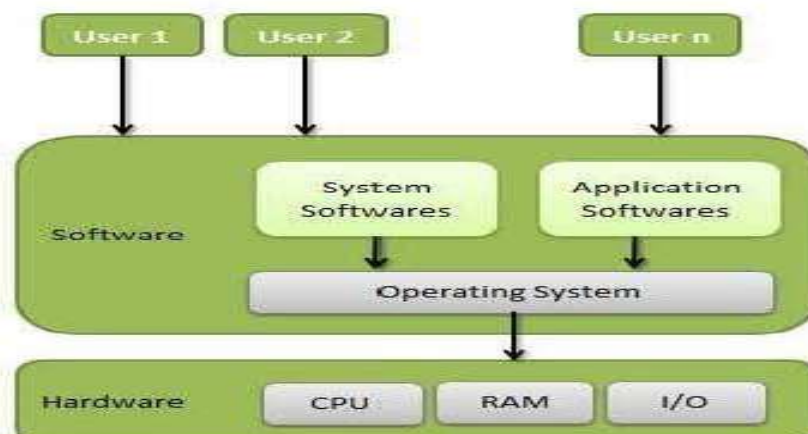


Figure: 2.1

Following are some of important functions of an operating System:-

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management:-

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address. Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must be in the main memory. An Operating System does the following activities for memory management:

- Keeps track of primary memory, i.e., what part of it are in use by whom, what part is not in use?
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management:-

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process

scheduling. An Operating System does the following activities for processor management:

- Keeps track of processor and status of process. The program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management:-

An Operating System manages device communication via their respective drivers. It does the following activities for device management:

- Keeps tracks of all devices. The program responsible for this task is known as the I/O controller
- Decides which process gets the device when and for how much time.
- Allocates the device in the most efficient way.
- De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directories. An Operating System does the following activities for file management:

- Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

2.1.2 Other Important Activities

Following are some of the important activities that an Operating System performs:

- Security -- By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- Control over system performance -- Recording delays between request for a service and response from the system.
- Job accounting -- Keeping track of time and resources used by various jobs and users.
- Error detecting aids -- Production of dumps, traces, error messages, and other debugging and error detecting aids.
- Coordination between other software and users -- Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

2.2 Distribution of Linux

Although there are a large number of Linux implementations, you will find a lot of similarities in the different distributions, if only because every Linux machine is a box with building blocks that you may put together following your own needs and views. Installing the system is only the beginning of a long term relationship. Just when you think you have a nice running system, Linux will stimulate your imagination and creativeness, and the more you realize what power the system can give you, the more you will try to redefine its limits.

Linux may appear different depending on the distribution, your hardware and personal taste, but the fundamentals on which all graphical and other interfaces are built, remain the same. The Linux system is based on GNU tools (Gnu's Not UNIX), which provide a set of standard ways to handle and use the system. All GNU tools are open source, so they can be installed on any system. Most distributions offer pre-compiled packages of most common tools, such as RPM packages on RedHat and Debian packages (also called deb or dpkg) on Debian, so you needn't be a programmer to install a package on your system. However, if you like doing things yourself, you will enjoy Linux all the better, since most distributions come with a complete set of development tools, allowing installation of new software purely from source code. This setup also allows

you to install software even if it does not exist in a pre-packaged form suitable for your system.

2.3 RedHat

2.3.1 Introduction

Red Hat Linux, assembled by the company Red Hat, was a popular Linux based operating system until its discontinuation in 2004.

Red Hat Linux 1.0 was released on November 3, 1994. It was originally called “Red Hat Commercial Linux”. It was the first Linux distribution to use the RPM Package Manager as its packaging format, and over time has served as the starting point for several other distributions, such as Mandriva Linux and Yellow Dog Linux.

Red Hat Linux introduced a graphical installer called Anaconda, intended to be easy to use for novices, and which has since been adopted by some other Linux distributions. It also introduced a built-in tool called Lokkit for configuring the firewall capabilities.

Red Hat Enterprise Linux WS is the desktop/client partner for Enterprise Linux AS and Enterprise Linux ES. It is ideal for all desktop deployments, including office productivity applications, software development environments, and targeted ISV client applications. When configured as a headless workstation, Enterprise Linux WS is also ideally suited for use as a compute node in a High Performance Computing (HPC) environment.

Red Hat Enterprise Linux WS is fully compatible with other members of the Red Hat Enterprise Linux product family and provides complementary technology and services.

Red Hat Enterprise Linux WS provides support for desktop/workstation systems with up to two CPUs. Designed with the desktop environment in mind, Red Hat Enterprise Linux WS does not include a number of server applications found in Red Hat Enterprise Linux AS and Red Hat Enterprise Linux ES.

At this time the family comprises five members:

Red Hat Enterprise Linux AS (formerly Red Hat Linux Advanced Server)
Designed for high end X86-based servers running corporate-level applications, such as databases.

Red Hat Enterprise Linux ES

Designed for small and medium X86-based servers running edge-of-network and departmental-level applications.

Red Hat Enterprise Linux WS

Designed for X86-based technical and commercial application workstations.

Red Hat Linux Advanced Server for the Itanium Processor

This product is equivalent to Red Hat Enterprise Linux AS, providing support for Intel's Itanium 2 processors.

Red Hat Linux Advanced Workstation for the Itanium Processor

This product is equivalent to Red Hat Enterprise Linux WS, providing support for Intel's Itanium 2 processors.

What's Included

Red Hat Professional Workstation is a complete suite of tools for the power desktop users. Supports up to 2 processor x86 workstations

- Includes applications and services power desktop users demand Bluecurve, Ximian Evolution, OpenOffice.org, Mozilla Web Server powered by Apache Samba, NFS, CUPS GCC 3.2
- Based on Red Hat Enterprise Linux WS with 30 days of phone and web installation support during standard business hours One year of Red Hat Network updates and upgrades
- Annually renewable – get the latest updates and even upgrades when they are available for future Red Hat Enterprise Linux WS versions.
- Hardware and Software certification identical to Red Hat Enterprise Linux WS

What's not Included

Red Hat Professional Workstation is designed for the advanced users requiring a single Linux desktop deployment with limited support and management capabilities. Customers looking for advanced support options and server features or planning on deploying Linux on a large scale should look to Red Hat Enterprise Linux for additional options.

Features not available with Red Hat Professional Workstation (but standard in Red Hat Enterprise Linux) include:

- **Additional Support options** -- Customers needing support beyond basic installation should look to Red Hat Enterprise Linux.
- **Open Source Architecture Integration** -- Red Hat Enterprise Linux provides the necessary components and support for integration into a managed corporate desktop infrastructure. Red Hat Enterprise Linux will be the core platform of the Open Source Architecture.

Red Hat Enterprise Linux creates a reliable, secure, high performance platform designed for today's commercial environments--with capabilities that match or surpass those of proprietary operating systems. Sold in three products that span client systems to the largest servers, Red Hat Enterprise Linux delivers a consistent application, management, and user environment.

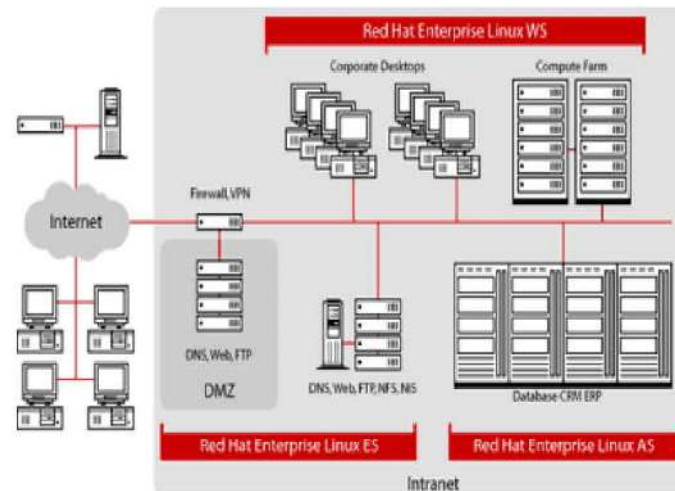
Red Hat Enterprise Linux is the corporate Linux standard. It's already at work running some of the world's largest commercial, government, and academic institutions. For any deployment--from the desktop to the data center -Red Hat Enterprise Linux delivers unmatched performance and cost savings, and the freedom of open source technology.

2.3.2 Advantages of RedhHat

User support is matched to the traditional open source ideology can make changes to any part of the system.

- Flexibility and security

- Software Compatibility Issues solved and find to check new operating system, new web browser, a new email client and a new photo editor for their Linux.
- Enterprise level support will make Red Hat easy to use



Red Hat Enterprise Linux Architecture

Figure 2.2

2.4 Centos Linux

2.4.1 Introduction

The CentOS Linux distribution is a stable, predictable, manageable and reproducible platform derived from the sources of Red Hat Enterprise Linux (RHEL). Looking to expand on that by creating the resources needed by other communities to come together and be able to build on the CentOS Linux platform. And today we start the process by delivering a clear governance model, increased transparency and access. In the coming weeks we aim to publish our own roadmap that includes variants of the core CentOS Linux.

Since March 2004, CentOS Linux has been a community-supported distribution derived from sources freely provided to the public by Red Hat. As such, CentOS Linux aims to be functionally compatible with RHEL. We mainly change packages to remove upstream vendor branding and artwork. CentOS Linux is no-cost and free to redistribute.

CentOS Linux is developed by a small but growing team of core developers. In turn the core developers are supported by an active user community including system administrators, network administrators, managers, core Linux contributors, and Linux enthusiasts from around the world.

CentOS is a community-supported, free and open source operating system based on Red Hat Enterprise Linux. It exists to provide a free enterprise class computing platform and strives to maintain 100% binary compatibility with its upstream distribution. CentOS stands for Community ENTerprise Operating System. CentOS developers use Red Hat's source code to create a final product very similar to Red Hat Enterprise Linux. CentOS is available free of charge. Technical support is primarily provided by the community via official mailing lists, web forums, and chat rooms.

CentOS version numbers have two parts, a major version and a minor version. The major and minor version numbers respectively correspond to the major version and update set of Red Hat Enterprise Linux from which the source packages used to build CentOS are taken. For example, CentOS 4.4 is built from the source packages from Red Hat Enterprise Linux 4 update 4.

2.4.2 Advantages of Centos :

- Virtualization allows multiple operating systems to run on a single server at the same time.
- SELinux Trouble Shooting Tool is a user-friendly tool for notification and diagnosis of access denials.
- Clustering and security.

There are three primary CentOS repositories (also known as channels), containing software packages that make up the main CentOS distribution

- **base** – contains packages that form CentOS point releases, and gets updated when the actual point release is formally made available in form of ISO images.
- **updates** – contains packages that serve as security, bugfix or enhancement updates, issued between the regular update sets for point

releases. Bugfix and enhancement updates released this way are only those unsuitable to be released through the CentOS-Fasttrack repository described below.

- **addons** – provides packages required for building the packages that make up the main CentOS distribution, but are not provided by the upstream.

CentOS Pros and Cons

- **Pros :**

- Freely available (no licenses)
- Everything in one “channel”
- Strong and big community support
- Stable/tested
- No need to upgrade to the next major release
- Not limited by Upstream restrictions

- **Cons :**

- No official support from upstream
- No RHN™ (SpaceWalk alternative available now)
- Not bleeding edge
- Newer softwares will require maybe newer underlying software in the future (all enterprise distributions)
- 'small' delay for the security releases (2h – 24h)

Supported architectures

- CentOS 2 only supports x86.
- CentOS 3 currently supports x86, x86_64 (AMD64 and Intel EM64T), s390, s390x, ia64 (Intel Itanium2).
- CentOS 4 currently supports x86, x86_64, s390, s390x and ia64. ppc (PowerPC), alpha (DEC Alpha) and sparc are released in beta for CentOS 4.
- CentOS 5 currently supports x86 and x86_64. ia64, ppc (PowerPC) and sparc are being developed.

2.5 Fedora

2.5.1 Introduction

Fedora is another example of a Linux distribution. In appearance, Fedora's desktop and standard operations are similar to those of Ubuntu and Kubuntu. Fedora is used primarily for older PCs due to its limited system requirements. The term fedora was in use as early as 1891. The fedora came into use in about 1919. Fedoras can be found in nearly any color imaginable, but black, grey, tan, brown, and red are the most popular.

Fedora is a popular open source Linux-based operating system. Fedora is designed as a secure, general purpose operating system. The operating system is developed on a six-month release cycle, under the auspices of the Fedora Project. Fedora is sponsored by Red Hat. According to the Fedora Project, Fedora is "always free for anyone to use, modify, and distribute." Fedora is said to be the second-most commonly used Linux distribution.

Fedora Minimum Requirements

The minimum requirements for the current (20) release of Fedora are:

- 1GHz Processor
- 1 GB
- 10 GB free disc space
- DVD drive or USB port (-for installation media)

2.5.2 Advantages of Fedora :

- Fedora is pretty similar to RHEL
- Fedora fairly convenient and relatively easy to install
- Fedora has the latest releases of most of the available software
- The choice of software is enormous
- Red Hat uses it as a vehicle for testing and proving technologies for release in its market-leading commercial RHEL release

- Reliability: in our experience Fedora 15 is extremely robust: it seems to cope well with flakey hardware and any application failures are confined to that application, rather than affecting the whole operating system

Fedora guarantees the Four Freedoms:

- The freedom to run the program, for any purpose
- The freedom to study how the program works, and adapt it to your needs
- The freedom to redistribute copies so you can help others
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits

New Features in Fedora 11

- 20 second boot time
- Ext4 as default filesystem
- Nouveau as the default driver for Nvidia cards
- Presto Yum Plugin for Delta RPMs
- GDM and GNOME finger print integration
- Security enhancements

Cons of Fedora

The main drawbacks of using Fedora are:

- Fedora is a great choice for the more experienced Linux hand but it is not, perhaps, the most user-friendly for the novice, with the user often having to resort to the command line to complete common tasks. Particular issues we've had are:
- If the Software Updater has two packages from different sources, it can be difficult to track down the offending package (-if you have a lot of updates pending) and to fix it without resorting to yum on the command line, which is confusing to most newcomers

- There are nowhere near as many applications available for Fedora as for Ubuntu - and those that are often need to be installed via the command line or by adding custom repositories (-such as RPM Fusion)
- Fedora is quite large and requires a reasonably capable machine to run effectively (-see the minimum requirements). If your machine is particularly old (-and you can't upgrade it) then you may be better off with something like Puppy instead
- Fedora 18 in particular is, by default, quite locked-down security-wise. This means the user may need to spend time tweaking the security in order to run certain applications
- Fedora lacks support for any proprietary formats (-such as Flash, MP3, MP4, etc.) meaning that the user will have to enable a secondary repository (-such as RPMFusion and install and configure all these things manually)

Fedora Applications

Fedora ships with a minimum of installed applications: please see our Useful Linux Applications to see which applications it ships with - and where to go to install others.

2.6 Debian Linux

2.6.1 Introduction

The Debian Project is an association of individuals who have made common cause to create a free operating system. This operating system that we have created is called Debian.

An operating system is the set of basic programs and utilities that make your computer run. At the core of an operating system is the kernel. The kernel is the most fundamental program on the computer and does all the basic housekeeping and lets you start other programs.

Debian systems currently use the Linux kernel or the FreeBSD kernel. Linux is a piece of software started by Linus Torvalds and supported by thousands of

programmers worldwide. FreeBSD is an operating system including a kernel and other software.

However, work is in progress to provide Debian for other kernels, primarily for the Hurd. The Hurd is a collection of servers that run on top of a microkernel (such as Mach) to implement different features. The Hurd is free software produced by the GNU project. A large part of the basic tools that fill out the operating system come from the GNU project; hence the names: GNU/Linux, GNU/kFreeBSD, and GNU/Hurd. These tools are also free. Of course, the thing that people want is application software: programs to help them get what they want to do, from editing documents to running a business to playing games to writing more software. Debian comes with over 43000 packages (precompiled software that is bundled up in a nice format for easy installation on your machine), a package manager (APT), and other utilities that make it possible to manage thousands of packages on thousands of computers as easily as installing a single application. All of it free.

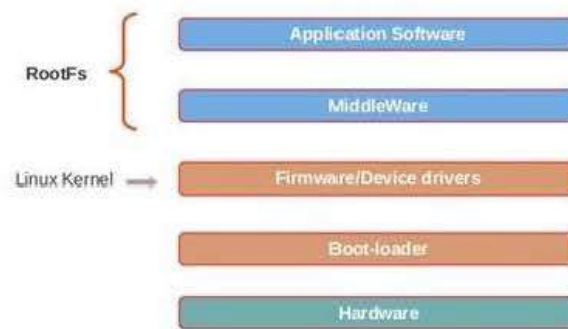
It's a bit like a tower. At the base is the kernel. On top of that are all the basic tools. Next is all the software that you run on the computer. At the top of the tower is Debian carefully organizing and fitting everything so it all works together.

Hardware supported

Debian will run on almost all personal computers, including older models. Each new release of Debian generally supports a larger number of computer architectures. For a complete list of currently supported ones, see the documentation for the stable release.

Almost all common hardware is supported. If you would like to be sure that all the devices connected to your machine are supported, check out the Linux Hardware Compatibility HOWTO.

There are a few companies that make support difficult by not releasing specifications for their hardware. This means you might not be able to use their hardware with GNU/Linux. Some companies provide non-free drivers, but that is a problem because the company could later go out of business or stop support for the hardware you have. We recommend that you only purchase hardware from manufacturers that provide free drivers for their products.



Debian Architecture

Figure 2.3

2.6.2 Advantages of Debian Linux

1. Upgradability

A Debian GNU system is upgraded to a new minor subrelease or to a new major release by using one out of many methods. Single desktop machines can use the CD-ROM-method or the network-method while the machine runs without the need to reboot it.

2. Availability

Debian GNU has been released for several architectures using the Linux kernel. Ports using a different kernel (Hurd, NetBSD and FreeBSD) are in progress as well.

3. Debian is Free

Debian reads and interprets licenses of software very careful. With the Debian Free Software Guidelines a rule-set exists that helps separate Free Software from non-free software. Using only Debian the user can be sure to be granted all required freedoms.

4. Integration

Debian packages integrate very well into the entire system. Several tools and gadgets are used to help connecting packages to each other, presenting the user a well maintained and round system.

5. Filesystem Standard

Debian follows the Filesystem Hierarchy Standard very close. Hence, all Debian packages implement the same file and directory structure. `/etc` contains all configuration files; they don't clutter the entire filesystem instead. `/usr/local` and `/opt` are not touched by Debian but reserved for the local admin. Documentation is placed in `/usr/share` etc.

6. **Configuration Files**

Configuration files are treated specially. Every package that provides configuration files has them marked so the package system knows about them. When a package is upgraded, the configuration file is not automatically overwritten when it was modified locally. Instead the admin is informed and offered to show the difference between the local and the new configuration file.

7. **Configuration Management**

Much of the configuration is handled by `debconf`, which itself offers a variety of frontends to use. People who like not to be annoyed with questions can configure it to skip many questions and use default values instead. Those who prefer a graphical interface can configure it to use a GTK frontend. This technique will also be used in the upcoming `debian-installer`.

8. **Source**

For all packages of the distribution the whole source is available. The entire distribution is free according to the DFSG which also means that everybody can improve packages and still distribute them.

9. **High Quality**

The maintainers generally have a strong personal interest in each package they maintain, since they normally volunteered to maintain it because they wanted to use it themselves. This results in high quality work, by highly motivated and generally technically skilful people, which in turn gives us high quality throughout the whole distribution.

10. **Pre-Configuration**

Each Debian package comes preconfigured. As a result, all programs work out-of-the-box once they're installed, there is no need to

reconfigure them afterwards. Of course, you are free to reconfigure or fine-tune the packages and Debian packages will not throw away your changes.

11. Bug Tracking System

The Debian Project maintains an open bug tracking system where everybody is able to report bugs to. Under normal circumstances bugs will be fixed within a few days.

12. Testbed

The Debian-Project pays more attention to quality and testing than to often-releasing. As a result of this, each released version of Debian GNU has been well tested over a long period and all major bugs have been removed.

13. Remote Maintenance

A Debian GNU system can fully administered remotely. This includes configuration and package maintenance as well as installation or removal of new packages. This is a unique feature of Debian GNU. You can do a full operating system upgrade remotely, almost always without the need to reboot, and generally without the services that the machine offers being off the air for more than a few minutes.

2.7 Ubuntu Linux

2.7.1 Introduction

The most popular and widely used Linux distribution is called “Ubuntu.” Ubuntu is aimed at bringing Linux to casual computer users and is comparable in features to Microsoft’s Windows operating system. Ubuntu is a computer operating system based on the Debian Linux distribution. Ubuntu focuses on usability and security. The Ubiquity installer allows Ubuntu to be installed to the hard disk from within the Live CD environment, without the need for restarting the computer prior to installation.

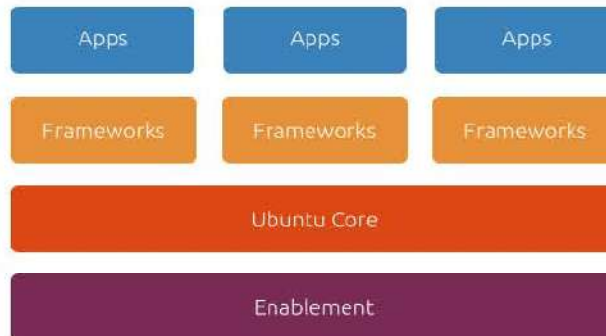
Ubuntu comes installed with a wide range of software that includes OpenOffice, Firefox, Pidgin, Transmission, GIMP, and several lightweight games (such as Sudoku and chess). Ubuntu allows networking ports to be

closed using its firewall, with customized port selection available. It offers support for more than 46 languages. Ubuntu can also run many programs designed for Microsoft Windows (such as Microsoft Office), through Wine or using a Virtual Machine.

Ubuntu Minimum Requirements

The minimum requirements for the current (14.x) releases of Ubuntu are:

- 700 MHz processor (Intel Celeron or better)
- 512 Mb RAM
- 5 GB free disc space
- Graphics card capable of at least 1024x768 resolution
- DVD drive or USB port (-for installation media)



Ubuntu Architecture

Figure 2.4

2.7.2 Advantages of Ubuntu

Pros and Cons of Ubuntu:-

Pros:-

- It is arguably the most user-friendly version of Linux out there. Ubuntu's whole philosophy is based around making it easier for the user - hiding any unnecessary complexity behind the scenes
- It has a huge repository of (free) software available - by far the most of any Linux distribution
- It has a huge installed base: it's the most popular Linux distribution, so there are plenty of people and websites out there supporting it

- It's backed by Canonical, which means that they have the resources to put out six-monthly releases plus bug-fixes

Cons:-

The main drawbacks of using Ubuntu are:

- Ubuntu has a slightly conservative approach to new technologies so, if you like to keep up with leading edge technologies then you may be better suited to a distribution such as Fedora instead
- In our experience, Ubuntu (-particularly 11.x) can be very sensitive to hardware (-or software) faults, which can make it much less stable than, for example, Fedora. If resilience is top of your list, then look to Fedora instead
- Ubuntu is quite large and requires a reasonably capable machine to run effectively: if your machine is particularly old (-and you can't upgrade it) then you may be better off with something like Puppy instead
- Ubuntu is not the most popular Linux distribution without good reason! However, question marks over its ability to cope with some dodgy
- hardware and software faults may lead you to look elsewhere.

2.8 GNU-Linux Connection

GNU is an operating system that is free software—that is, it respects users' freedom. The development of GNU made it possible to use a computer without software that would trample your freedom. GNU was launched by Richard Stallman (rms) in 1983, as an operating system which would be put together by people working together for the freedom of all software users to control their computing. rms remains the Chief GNUisance today.

We recommend installable versions of GNU (more precisely, GNU/Linux distributions) which are entirely free software. GNU is a Unix-like operating system. That means it is a collection of many programs: applications, libraries, developer tools, even games. The development of GNU, started in January 1984, is known as the GNU Project. Many of the programs in GNU are released under the auspices of the GNU Project; those we call GNU packages.

The name “GNU” is a recursive acronym for “GNU's Not Unix.” “GNU” is pronounced g'noo, as one syllable, like saying “grew” but replacing the r with n. The program in a Unix-like system that allocates machine resources and talks to the hardware is called the “kernel”. GNU is typically used with a kernel called Linux. This combination is the GNU/Linux operating system. GNU/Linux is used by millions, though many call it “Linux” by mistake.

GNU's own kernel, The Hurd, was started in 1990 (before Linux was started). Volunteers continue developing the Hurd because it is an interesting technical project.

2.9 Summary

- An operating system is the set of basic programs and utilities that make your computer run. At the core of an operating system is the kernel. The kernel is the most fundamental program on the computer and does all the basic housekeeping and lets you start other programs.
- Linux is a free and open Source Operating System developed by Linus Torvalds and came into existence in the year 1991, at AT&T's Bell Laboratories, released under the GPL. As any other OS main function of Linux OS is to manage the system resources.
- Distribution of Linux Although there are a large number of Linux implementations, you will find a lot of similarities in the different distributions, if only because every Linux machine is a box with building blocks that you may put together following your own needs and views.
- RedHat Introduction RedHat Linux Red Hat Linux, assembled by the company Red Hat, was a popular Linux based operating system until its discontinuation in 2004.
- The CentOS Linux distribution is a stable, predictable, manageable and reproducible platform derived from the sources of Red Hat Enterprise Linux (RHEL).

- The most popular and widely used Linux distribution is called “Ubuntu.” Ubuntu is aimed at bringing Linux to casual computer users and is comparable in features to Microsoft’s Windows operating system. Ubuntu is a computer operating system based on the Debian Linux distribution. Ubuntu focuses on usability and security.
- Fedora is a popular open source Linux-based operating system. Fedora is designed as a secure, general purpose operating system. The operating system is developed on a six-month release cycle, under the auspices of the Fedora Project. Fedora is sponsored by Red Hat.

2.10 Glossary

1. GIMP:- The GNU Image Manipulation Program, for X Windows system
2. GNU:-The GNU operating system is a complete free software system, upward-compatible with Unix.
3. CPU:- Central Processing Unit
4. I/O:- Input Output Unit
5. ARM is a leader in microprocessor Intellectual Property. ARM designs and licenses fast, low-cost, power-efficient RISC processors, peripherals

2.11 Exercise

1. What Linux Torvalds Created?
2. Torvalds, Wrote most of the Linux Kernel in C++ programming Language, do you agree?
3. Is it legal to edit Linux Kernel?
4. Is Linux Operating system Virus free?

Solution:

1. Linux Torvalds created Linux, which is the kernel (heart) of all of the above Operating System and all other Linux Operating System.
2. No! Linux Kernel contains 12,020,528 Lines of codes out of which 2,151,595 Lines are comments. So remaining 9,868,933 lines are codes

and out of 9,868,933 Lines of codes 7,896,318 are written in C Programming Language.

3. Yes, Kernel is released under General Public Licence (GPL), and anyone can edit Linux Kernel to the extent permitted under GPL. Linux Kernel comes under the category of Free and Open Source Software (FOSS).
4. No! There is no Operating System on this earth that is virus free. However Linux is known to have least number of Viruses, till date, yes even less than UNIX OS. Linux has had about 60-100 viruses listed till date. None of them actively spreads nowadays. A rough estimate of UNIX viruses is between 85 -120 viruses reported till date.

UNIT-3

Architecture

Structure of the Unit

- 3.0 Objective
- 3.1 Basic Architecture of Unix/Linux System
- 3.2 Kernel
- 3.3 Shell
- 3.4 Self Learning Exercise
- 3.5 Summary
- 3.6 Glossary
- 3.7 Answers to Self-Learning Exercise
- 3.8 Exercise
- 3.9 Answers to Exercise

3.0 Objective

In this chapter we shall focus upon the following topics

- Basic Architecture of Unix/Linux System
- Kernel
- Shell

3.1 Basic Architecture of Unix/Linux System

Introduction

Linux is one of the most popular OS version of UNIX operating system. Linux is open source so, its source code is freely available. It is free to use for all. Linux was designed considering UNIX compatibility. Linux functions set is quite similar to UNIX operating system. Basically UNIX is no longer a small system, But it certainly a powerful one. We need to require the understanding of its software architecture.

Some key feature of UNIX architecture concept are as follows:

- UNIX systems use a centralized OS kernel which process tasks and manages system.
- UNIX systems are preemptively multitasking i.e. multiple processes can run at the same time or with in small time slices and nearly at the same time, any process can be interrupted and moved out of execution by the kernel. This concept is known as thread management.
- All non – kernel software is organized into separate kernel managed processes.
- With certain exceptions, devices and some kind of communication between processes are managed and visible as files or pseudo files within the file system hierarchy. Linus Torvalds who invent Linux OS, states that this is in-accurate and may be better phrased as “everything is a stream of bytes”.
- Files are stored on disk in hierarchical manner i.e. called file system, with a single top location throughout the system, with both files and directories, subdirectories and sub- subdirectories and so on.

BASIC FEATURES OF LINUX

Following are some important features of Linux operating system:

1. Open Source

Linux operating system source code is freely available and I community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

2. Portability

The term portability means software can work on different type of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform. It can be run on any hardware configuration system.

3. Multiuser System

According to fundamental point of view, UNIX is multiprogramming system, it allows multiple program to run and compete for the attention of the CPU. That is done in two ways:-

- Multiple users can run separate jobs.
- A single user can also run multiple jobs.

Several processes constantly running on a UNIX system. Multiple users feature working on a single system often baffles window users. Basically, window is a single user system where the CPU, memory and hard disk are all dedicated to a single user. In the UNIX system, resources are shared between all users. So we can say UNIX is a multiuser system. For creating illusory effect, computer breaks up a unit of time into several segments, and each user is allotted a segment. So at any point of time, machine will be doing the job of single user. The moment allocated time expires, the previous job is kept in stored in buffer and the next user's job is taken up. This process goes on until the clock has turned full circle and the first user's job is taken up once again. This the kernel does several time in one second.

4. **Multitasking System**

A single user can run multiple jobs concurrently, So UNIX is a multitasking system. It is usual for a user to edit a file, send email to his friend, print another one on the printer and browse the World Wide Web i.e. all without leaving any of the applications. The kernel is designed in such a manner that can handle user's multiple needs.

In the multitasking environment, one job running in the foreground, the rest run in the background in the system. You can switch jobs between background and foreground, terminate them and even though suspend. Programmers can use this feature in a very productive way. You can edit a C or C++ program and then suspend it to run the compiler, you don't have to quit the editor to do that. This feature is provided by most of shells except Bourne shell.

5. **Building Block Approach**

The programmer or designer never attempted to pack too many features into a few tools. Instead they think "small is beautiful". And developed a few hundred commands each of which performed one simple job only. Using two commands ls and wc with the | (pipe) to count the number of files in your current directory. No separate command was designed to

perform a job. The command that can be connected in this way are called filters because they filter/manipulate data in different ways.

It is through pipes and filters that UNIX implements the small is beautiful philosophy. Many UNIX tools are designed with the requirement that the output of the one tool be used as input to another tool. So that UNIX architecture had to make sure that commands did not throw out and clutter the output. One reason behind this is UNIX programs are not interactive. And output of the `ls` command contained column header, or if it prompted the user for specific information, output of `ls` command could not have been used as useful input to `wc` command.

By interconnection of number of tools, you can have a large number of combinations of their usage. So it's better for a command to handle a specialized function rather than solve multiple problems.

6. **Pattern matching**

Pattern matching is very sophisticated feature. By using `ls` command with `*` (`chap*`), it denote all files names that are started with name Chapter. The `*` is a special character used by the system to indicate that it can match a number of filenames. If you use filename carefully, you can use a simple expression to access the whole information of them.

The `*`, known as metacharacter, is not only a special character used by UNIX operating system, there are many others. UNIX features elaborate pattern matching phenomena that generally use several characters from this metacharacter set. The matching is not defined to file name only. Most advanced and useful tools also use a special expression that is called as a regular expression which is framed with characters from this set. We also emphasis on how complex pattern matching task is evaluated using `*` metacharacter.

7. **Programming Facility**

Shell script of UNIX is also a programming language. It was designed for a programmer, not a casual end user. It contains all necessary ingredients, like control structure, loops and variables, that build it as a powerful programming language. These features are used to design shell

script for any real time problem. Programs that can also invoke the UNIX commands.

Many system can be controlled and automated by using these shell scripts. If you intended taking up as a system administrator, you will have to know about complete features detail of shell programming.

8. **Documentation**

This feature also supported by UNIX operating system. The principle online help facility available is the 'man' command, which remains the most important reference for commands with their configuration files. Apart from UNIX documentation there is a vast resources available on the internet. There are several newsgroups on UNIX where you can fire your queries. Problem can be related to shell programming or a network configuration issue. A document that addresses common problems is also widely available on the internet.

9. **Shell**

Linux provides a special type interpreter program which can be used to execute different commands of the UNIX/Linux operating system. It can be used to do various types of operations, call application programs etc.

10. **Security** – Linux provides user security using authentication features like password protection/controlled access to specific files/encryption of data.

Components of Linux System

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

System Library – These are special functions or programs using which application program or system utilities access kernel features. System libraries implements most of the functionalities of the operating system and don't requires kernel module's code access rights.

System Utility – System Utility programs are responsible to do specialized, individual level tasks.

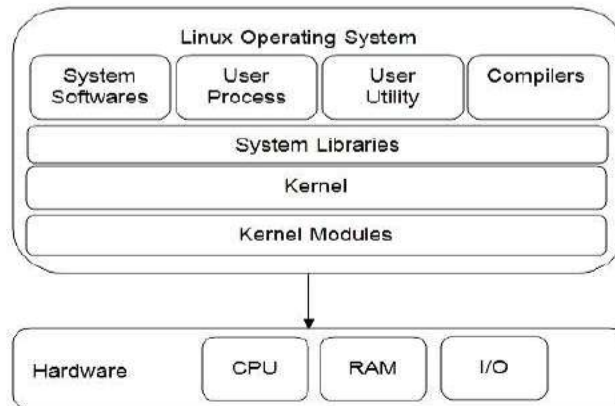


Figure 3.1 Basic Components of Linux operating System

Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and that's why it is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes. Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in User Mode which has no access to system hardware and kernel code. User programs design in any language or script /utilities use system libraries to access kernel functions to get system's low level tasks.

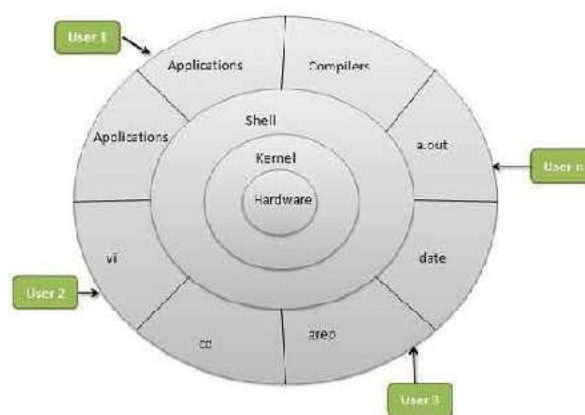


Figure-3.2 (a) Architecture of UNIX/ Linux operating system (Functional Model)

The architecture of Linux consists of five parts:

Kernel

1. Shell
2. System Utilities
3. User applications
4. Hardware Platform

The basic architecture of UNIX/Linux operating system is as follows:

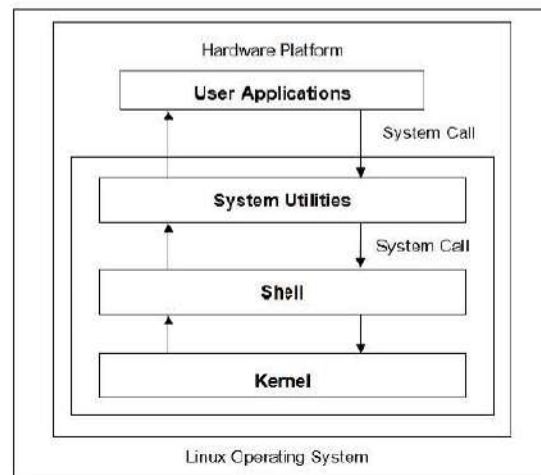


Figure- 3.2(b) Linux System Architecture

Linux system architecture is consists of following layers:

Hardware Layer – This layer consists of all peripheral devices to the system such as RAM/ HDD/ CPU etc. It may also consists of additional peripheral devices such as secondary storage devices, cache memory registers etc.

Kernel – This layer is the core component of the operating system. Kernel directly interacts with hardware, provide low level services to the upper layer components. Kernel is the interface between hardware and shell.

Shell – An interface to kernel, with hiding complexity of kernel's functions from the users. Take commands from users and executes kernel's functions. It is mainly known as command interpreter.

System Utilities – Utility of the program giving user most of the functionalities of an operating system. These program are responsible to do specialized, individual level tasks.

3.2 Kernel

1. Kernel

On a purely technical level, the kernel is an intermediary layer between the hardware and the software. Its purpose is to pass application requests to the hardware and to act as a low-level driver to address the devices and other components of the system.

Nevertheless, there are other interesting ways of viewing the kernel. The kernel is the core of the Linux operating system, it manages communication between devices and software, manages the system resources (like CPU time, memory, network ...) and shields off the complexity of device programming from the developer as it provides an interface for the programmer to manipulate hardware.

- The kernel can be regarded as an enhanced machine that, in the view of the application, abstracts the computer on high level. For example, when the kernel addresses a hard disk, it must decide which path to use to copy data from disk to memory, where the data reside, which commands must be sent to the disk via which path, and so on. Applications, on the other hand, need only issue the command that data are to be transferred. How this is done is irrelevant to the applications the details are abstracted by the kernel. Application programs have no linkage with the hardware itself, only with the kernel, which, for them, represents the lowest level in the hierarchy they know and is therefore an enhanced machine.
- Viewing the kernel as a resource manager is justified when several programs run concurrently on a system. In this case, the kernel is an instance that shares available resources like CPU time, disk space, network connections, and so on between the various system processes while at the same time ensuring system integrity.
- Another view of the kernel is as a library providing a range of system-oriented commands. As is generally known, system calls are used to send requests to the computer, with the help of the C standard library, these appear to the application programs as normal functions that are invoked in the same way as any other function.

Kernel Architecture (UNIX) :

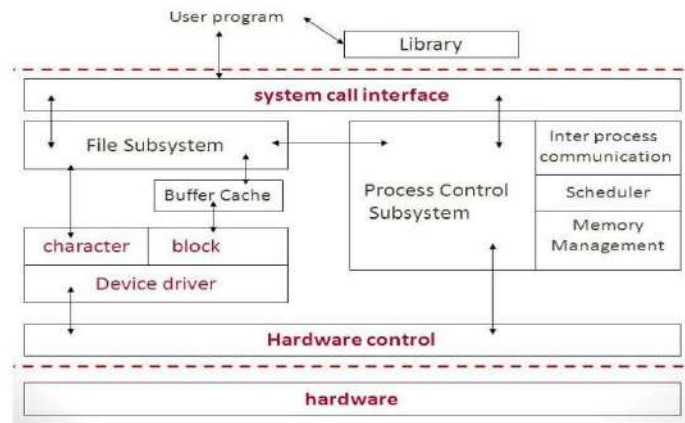


Figure- 3.3 Kernel Architecture of UNIX operating system

The Linux kernel is composed of five main subsystems:

a) The Process Scheduler (SCHED)

It is responsible for controlling process access to the CPU (Central Processing Unit). The scheduler enforces a policy that ensures that processes will have fair access to the CPU, while ensuring that necessary hardware actions are performed by the kernel on exact time.

b) The Memory Manager (MM)

Memory manager permits multiple processes to securely share the machine's main memory system. In addition, the memory manager supports virtual memory that allows Linux to support processes that use more memory than is available in the system. Unused memory is swapped out to persistent storage using the file system then swapped back in when it is needed.

c) The Virtual File System (VFS)

The virtual file system abstracts the details of the different types of hardware devices by presenting a common file interface to all devices. In addition, the VFS allows several file system formats that are compatible with other operating systems.

d) The Network Interface

Network Interface provides access to several networking standards and various type of network hardware.

e) **The Inter-Process Communication (IPC)**

Inter process communication subsystem supports several mechanisms for process-to-process communication (peer to peer communication) on a single Linux system.

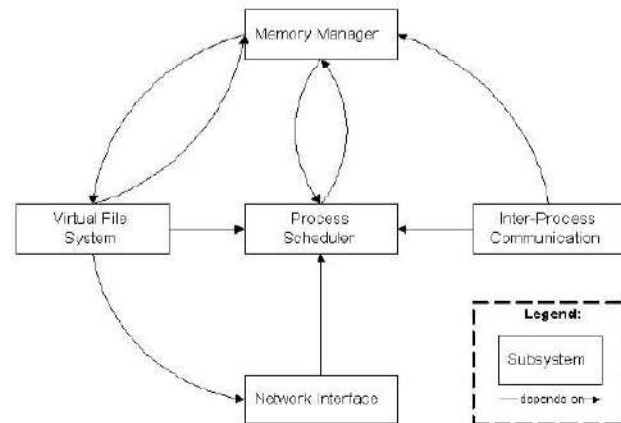


Fig. 3.4 Kernel Subsystem Overview

The fig. 3.4 emphasizes that the most central subsystem is the process scheduler, and remaining all other subsystems depend on the process scheduler since all subsystems need to suspend and resume processes. Generally a subsystem will suspend a process that is waiting for a hardware operation to complete, and resume the process when the operation is finished. For example, when a process attempts to send a message across the network, the network interface may need to suspend the process until the hardware has completed sending the message successfully. After the message has been sent (or the hardware returns a failure), the network interface then resumes the process with a return code indicating the success or failure of the operation. The other subsystems like memory manager, virtual file system, and inter-process communication etc. all depend on the process scheduler for similar reasons.

The other dependencies are somewhat less obvious, but equally important as these are.

- The process-scheduler subsystem uses the memory manager for adjustment of the hardware memory map for a specific process when that process is resumed.

- The inter-process communication subsystem depends on the memory manager to support a shared-memory communication mechanism. This mechanism allows two processes to access an area of common memory in addition to their usual private memory.
- The virtual file system uses the network interface to support a NFS, and also uses the memory manager to provide a RAM disk device.
- The memory manager uses the virtual file system to support swapping; this is the only reason that the memory manager depends on the process scheduler. When a process accesses memory that is currently swapped out, the memory manager makes a request to the file system to fetch the memory from persistent storage, and suspends the process.

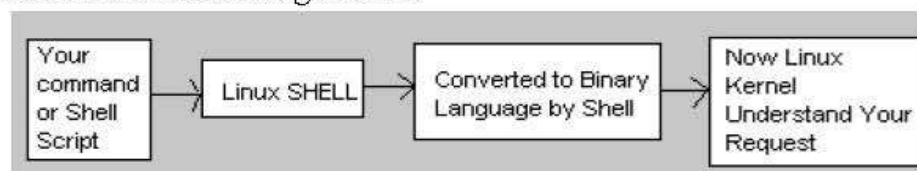
In addition to the dependencies that are shown explicitly, all subsystems in the kernel rely on some common resources which may be shared in the nature that are not shown in any subsystem. These include procedures that all kernel subsystems use to allocate and free memory for the kernel's use, procedures to print warning or error messages, and system debugging routines. Each of the depicted subsystems contains state information that is accessed using a procedural interface, and the subsystems are each responsible for maintaining the integrity of their managed resources.

3.3 Shell

1. Shell

Computer understands the language of 0's and 1's called binary language, In early days of computing, instructions and commands were provided using binary language, which was difficult for all of us, to read, write and understand. So in Operating system there is special program called Shell, Which accepts your instruction or commands in English and translate it into computers native binary language.

This is denoted in following manner:-



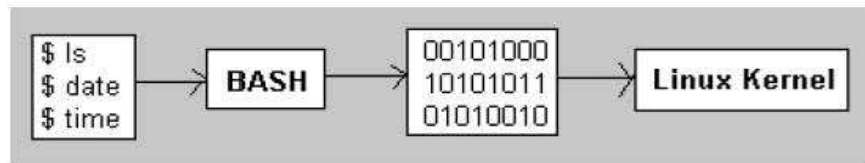


Fig. 3.5 Working of shell

Its environment provided for user interaction. Shell is a command interpreter that executes commands read from the standard input device or from a file. Linux may use one of the following most popular shells (In MS-DOS, Shell name is COMMAND.COM which is also used for same purpose, but it's not as powerful as Linux Shells.).

Shell is an interface between user and the operating system. It is the software that provides an interface for the user of an operation system which needs services of a kernel. An operating system shell is divided into two parts:-

- Command line
- GUI

Command line Shell

Command line shell is the part of the operating system which receives and executes the operating system command by the user. The commands are then sent to the kernel for execution. If the command is valid the kernel starts the execution else an error results. This process is also called as shell interactive cycle.

GUI (Graphical User Interface)

GUI provides a user-friendly environment. Users cannot remember the syntax of all the command and thus it helps to simply point toward the object by the mouse or some other pointing device which a user required for its execution.

Interconnection between Kernel and Shell

When user enters any command as an input for performing any operation the request goes to the SHELL. The Shell then translates these human-readable programs to machine language and then forwards the request to the kernel. The kernel receives the request from the shell, processes the request and then displays the result on the screen (on terminal or command prompt). All these functions are performed by the kernel in a transparent manner.

Type of Shell

Basically, in Linux operating system there are four types of shells which are as follows:

- sh
- bash
- csh and tcsh
- ksh

1. sh

The Bourne shell, which is called "sh," is one of the original shells, developed for UNIX computers by Stephen Bourne at AT&T's Bell Labs in 1977. It offers features such as input and output redirection, shell scripting with string and integer variables, and condition testing and looping etc.

2. bash

It is the most popular shell which is of sh motivated, that is more compatible with user, but with several enhancements. Linux systems still offer the sh shell, but "bash" is the "Bourne-again Shell," based on sh has become the new default standard. One attractive feature of bash is its ability to run sh shell scripts unchanged. Shell scripts are complex sets of commands that automate programming and able to reuse these scripts saves programmers time. Conveniences not present with the original Bourne shell include command completion and a command history.

3. csh and tcsh

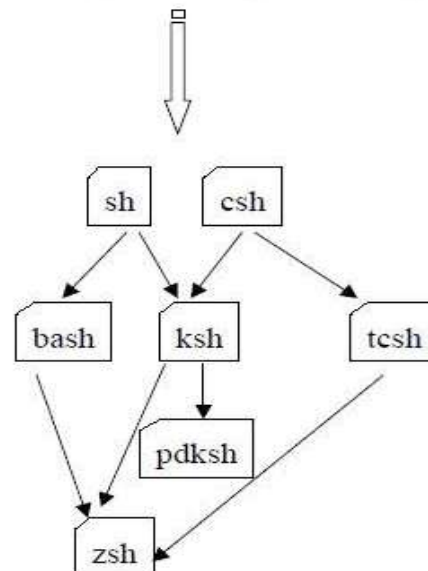
OS Developers have written large parts of the Linux operating system in the C and C++ languages. Using C syntax as a model, Bill Joy at Berkeley University developed the "C-shell" csh, in 1978. Ken Greer, working at Carnegie-Mellon University, took csh concepts a step forward with a new shell, tcsh, which Linux systems now offer. Tcsh fixed problems in csh and added command completion, in which the shell makes educated "guesses" as you type, based on your system's directory structure and files. Tcsh does not run bash scripts.

4. ksh

David Korn developed the Korn shell or ksh, about the time tcsh was introduced. Ksh is compatible with sh and bash. Ksh improves on the Bourne shell by adding floating-point arithmetic, job control, and command aliasing and completion. AT&T held proprietary rights to ksh until 2000, when it became open source.

Any of the above shell reads command from user (via Keyboard or Mouse) and tells Linux OS what user wants. If we are giving commands from keyboard it is called command line interface (Usually in-front of \$ prompt, This prompt is depend upon your shell and Environment that you set or by your System Administrator, therefore you may get different prompt).

NOTE: To find your shell type following command



\$ Echo \$SHELL

Fig. 3.6 Shell Hierarchy System

3. System Utilities –

The System Utilities consist of various system interrupts and system calls which is to transfer the control from the user mode to the kernel mode containing the kernel and the shell for further execution of the commands. The control can be transferred using system calls.

Shell	Roots	Default shell	Facts
sh		<ul style="list-style-type: none"> • on all UNIX • standard 	<ul style="list-style-type: none"> • the most popular • simple and robust • <u>best for shell prog.</u> • system utilities in sh
csh		<ul style="list-style-type: none"> • almost on all 	<ul style="list-style-type: none"> • a syntax similar to C • good for interactive • buggy for shell prog.
tsh	csh		<ul style="list-style-type: none"> • <u>best for interactive</u> • login shell in SLAC • less in shell prog.
bash	sh	<ul style="list-style-type: none"> • on Linux 	<ul style="list-style-type: none"> • <u>best for both</u> • less popular
ksh	sh and csh		<ul style="list-style-type: none"> • good for shell prog. • less popular
zsh	tsh,ksh, bash		<ul style="list-style-type: none"> • great for UNIX gurus

Comparison between different shells

System Call

System call is an interface between a process and the operating system. In simple words a system call is the request for running any program and for performing any operation on the system that the user has requested. Example a user has requested MS paint and this request will be sent to the operating system, which will then generate some activity to support MS paint and runs it on the system.

System calls are of different types –

- File Management System calls** – For performing open, close, read, write Operations.
- Process Control System calls** – For performing Load, execute, and create operations.
- Device Management System calls** – For performing request device, write device, and release device etc operations.

d) **Communication System calls** – For performing Send message, transfer status etc operations etc.

4. **User Applications** –

These are the applications which a user requires to perform some basic tasks. Linux and other operating systems come up with various different applications in them like g++, gcc, office suits etc. Kernel is used to generate processes to support these applications.

5. **Hardware Platform** –

The resources of the system such as keyboard, monitor, printer etc with which the user can input/output the request.

3.4 Self Learning Exercise

- Q.1 What are the basic features of Linux operating system? Explain in detail.
- Q.2 What is shell in Linux operating system? Explain its types.
- Q.3 Explain the architecture of Linux OS. Explain its different layers.
- Q.4 Explain the kernel architecture in details.

3.5 Summary

Operating system is the part of the system which is responsible for use and administration. In this chapter we have discussed about UNIX/Linux operating system Linux is CUI OS. Basic features of Linux OS are: open source, portability, multiuser system, multitasking system, building block approach, pattern matching, programming facility, documentation, shell and security etc. In the architecture of Linux OS there are mainly two layers one is kernel and another is shell. The kernel is the core of the Linux operating system, it manages communication between devices and software, manages the system resources. A Shell is an interface between user and the operating system. It is the software that provides an interface for the user of an operation system which needs services of a kernel. Shell are basically four types in Linux OS i.e. sh, bash, csh and ksh. The System Utilities consist of various system interrupts and system calls which is to transfer the control for the user mode to the kernel

mode containing the kernel and the shell for further execution of the commands.

3.6 Glossary

CUI: Command User Interface.

IPC: Inter Process Communication.

NFS: Network File System.

3.7 Answers to Self-Learning Exercise

Ans.1, Ans.2, Ans.3 and **Ans.4** is discussed in chapter in details. See respective contents.

3.8 Exercise

- Q.1** Kernel minimizes the frequency of disk access by keeping a pool of internal data buffer which help to increase the response time, this is known as?
- (A) Buffer cache (B) Spooling
(C) Virtual Memory (D) Pooling
- Q.2** The system calls in UNIX is written using which language?
- (A) Java (B) C
(C) C++ (D) Php
- Q.3** What is the property of UNIX?
- (A) Multi User (B) Multi Processes
(C) Multi-Tasking (D) All of these
- Q.4** Which is most important for Multi-tasking?
- (A) Modularity (B) Time Sharing
(C) Multi Programming (D) Multi User
- Q.5** What is kernel? Explain its architecture and working.
- Q.6** What is system call in Linux OS? Explain its different types.

Q.7 Differentiate all four types of shell in Linux OS on the basis of different parameters.

Q.8 Draw the functional model of UNIX/Linux operating system.

3.9 Answers to Exercise

Ans.1:A

Ans.2: B

Ans.3:D

Ans.4: B

References and Suggested Readings

1. A practical Guide to Linux, Sobell, Pearson.
2. A Guide to Fedora and Red Hat Enterprise Linux, Sobell, 5e, Pearson.
3. The Official Fedora Documentation available at “<http://docs.fedoraproject.org>”.
4. The Official Ubuntu Documentation available at “<https://help.ubuntu.com/>”.
5. UNIX concept and applications, 4e, Tata McGraw Hill.

UNIT-4

File System

Structure of the Unit

- 4.0 Objective
- 4.1 Introduction
- 4.2 File Systems
- 4.3 Linux File System
- 4.4 Standard Directories of Linux
- 4.5 Windows File System
- 4.6 Boot block, Super block, Inode table, Data blocks
- 4.7 Storage of files
- 4.8 elf-Learning Exercise
- 4.9 LILO, GRUB boot loader
- 4.10 Summary
- 4.11 Glossary
- 4.12 Answers to Self-Learning Exercise
- 4.13 Exercise
- 4.14 Answers to Exercise

4.0 Objective

In this chapter we shall focus upon the following topics

- File System
- Linux File System
- Storage of files

4.1 Introduction

A file is potentially large amount of information or data that lives a (potentially) very long time.

- Often much larger than the memory of the computer
- Often *much* longer than any computation
- Sometimes longer than life of machine itself

4.2 File Systems

File system is a method of organizing and retrieving files from a storage medium, such as a hard drive. File systems usually consist of files separated into groups (usually contained in directories), and relevant information. The hard drive may have a single file system or it may have several file systems.

The disk space allocated to Linux file system is made up of blocks. The block size depends on how the file system is implemented on a particular installation, and it may be of 512/1024/2048 bytes. It may vary from version to version.

To know the size of block on a file system, there is a command named `cmchk`.

\$cmchk

BSIZE=1024

When a file is created, one block is allocated to store the contents of file. If size of file created, is smaller than block size still it will be allocated one complete block and the remaining storage is unutilized.

Eg: - If block size is 1024 bytes, and we create a file of say 490 bytes. This will result in 534 (1024 - 490) bytes unutilized.

What we can think from this is that, we should have block size as small as possible. But this approach is also not so fruitful, because larger the file size more blocks the file would require for storage. Thus, more disk accesses will be required for complete access of the file. Disk accesses are time consuming tasks, so more time would be required to read/write to such file.

4.3 Linux File System

Linux file system is similar to that of Unix. Everything is treated as a file, whether it is application, utility, data, directory or even devices. The file system is similar to tree data structure i.e hierarchical.

The file system begins with a root directory (denoted by /). All files and directories are contained within this root directory. Root directory contains various other directories such as bin, usr, etc, tmp, dev, home etc.

The home directory contains directories of different users. Since Linux is a multiuser operating system, it creates a separate directory for each user. As we can see in Fig 1, directories named jono, mako, cory are home directories of respective users by the same name. Home directory of user is denoted by ~ (tilde). If we login with credentials of username jono, ~ is denoting home directory of jono. Similarly if we login with credentials of cory, ~ is denoting home directory of cory.

The absolute path of any file/directory is denoted starting from root directory. For eg: - absolute path of directory photos in Fig 1 is given by: -

/home/jono/photos

The first /(before home) denotes root directory, other / in the absolute path are used as separator.

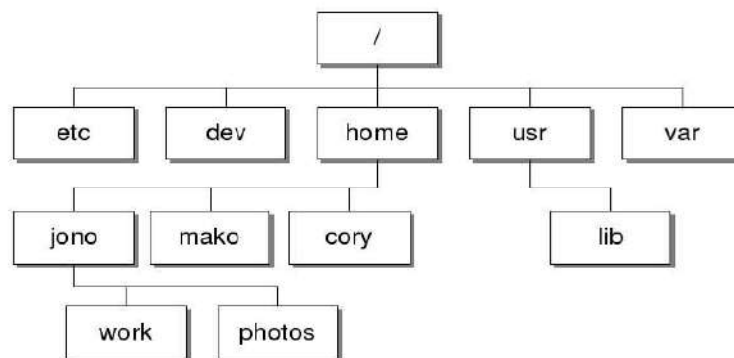


Fig 4.1:- File System of Linux

4.4 Standard Directories of Linux

As we can see in Fig 1, there are different directories in Linux file system. Directories are used to keep related file together. Different directories in root directory have their own significance. Let's discuss some of them

Usr :- keep all user related files.
Dev :- contains all device related files.
Tmp :- all temporary files.
Bin :- executable files for most of the commands.
Etc :- configuration files for the system.

4.5 Windows File System

Windows file system is different from Linux file system. Windows file system has different root for each logical drive, and each logical drive in itself is a tree like structure. So, we can say that windows file system resembles forest (collection of trees) data structure.

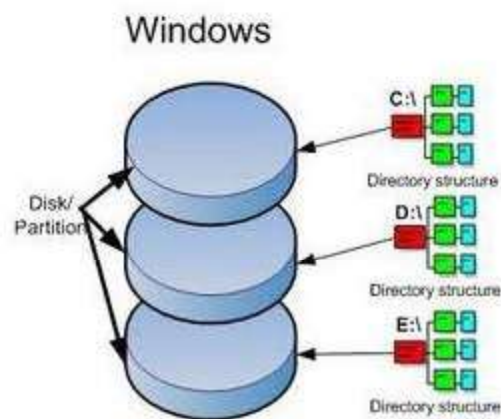


Fig 4.2: - Windows file system

So major difference between Windows and Linux file system is that, Linux file system resembles tree data structure, whereas Windows file system resembles forest data structure.

4.6 Boot block, Super block, Inode table, Data blocks

The structure of a simple Linux file system is usually divided into four parts: -

- Boot block
- Super block
- Inode list
- Data block

Boot Block

The boot block represents beginning of the file system and can be accessed with the minimal code incorporated in the computer's ROM BIOS. It contains a program called 'bootstrap loader'. This program is executed when we 'boot' the host machine. The boot block of the bootable partition contains the code needed to further initialize the operating system.

Super Block

The super block describes the state of a file system - how large it is, how many maximum files it can store, how many more files can be created, what parts of the storage area are already in use and what parts are available, etc.

The super block contains the following information

- Size of the file system.
- This is the storage size of the drive or current partition on the drive.
- List of storage blocks
- The storage space is divided into a series of standard size blocks. When data is moved to or from the filesystem, it is moved in blocks.
- Number of free blocks.
- Location of all free blocks.
- Index of the next free block in the free block list.
- Size of the inode list.
- The inode list is initialized to keep a track of the maximum number of files (cannot be more than the maximum number of blocks)
- Number of free inodes in the file system.
- Index of the next free inode in the free inode list.

Inode List

As we have already discussed in section 4.3 that, all entities are treated as files in Linux. The information about the files is stored in Inode (acronym for Index Node) table on the drive. For each file, there is an inode entry in the table. The inode entry of file contains data about file, and not the actual file content. Each file has an associated inode number, which is unique for each file. The inode number for any file can be obtained by using following command on command prompt: -

\$ ls -i <filename>

Where<filename> need to be replaced by actual file name. For example, to find the inode number of file named xyz, we need to write the following command on command prompt: -

\$ ls -i xyz

An inode entry corresponding to a file contains following information: -

- **File owner id(uid)**
This is the numeric id used in the password file to uniquely identify a user on the system.
- **Group id (gid)**
Group to which owner belongs or in other words, this identifies a group that can be granted special access by the owner.
- **File type.**
By default Linux has only 3 types of files viz Regular files, Directory files and Special files. There are five sub types of special files, which are discussed in following table

File denoted by	Type of file	Description
-	Regular file	A file of data
d	Directory file	File that contains filenames and their associated inode numbers.
b	Special file	Block special file - a file that accesses a set number of bytes (block) at a time. The video screen is usually accessed as a blockspecial file device. These files are hardware files, most of them are present in /dev
c	Special file	Character special file - a file that is supposed to access one character at a time. The keyboard would be accessed as a character

		file. Provides a serial stream of input or output.
p	Special file	Named pipe file (fifo) - when connecting the output of a command to the input of another, the system may need to buffer the data. The other name of pipe is a “named” pipe, which is sometimes called a FIFO. FIFO stands for “First In, First Out” and refers to the property that the order of bytes going in is the same coming out. The “name” of a named pipe is actually a file name within the file system.
l	Special file	Symbolic link file - a file that contains the path information needed to access a file. These are linked files to other files. They are either Directory/Regular File. The inode number for this file and its parent files are same. There are two types of link files available in Linux/Unix i.e soft and hard link.
s	Special file	Socket file- A socket file is used to pass information between applications for communication purpose.



File access permissions

There are three types of permissions

read (r)- permission to read the data stored in the file.

write (w)- to be able to modify/overwrite the data stored in the file.

execute (x)- to be able to request that the system attempt to execute.

These three permissions are for three types of users

Owner – user who owns the file, usually the creator.

Group - access by the member of group.

Other—all other users, who are not the owner or recognized group, (run) the file as a command.

To check the current permissions of a file, following command can be used

ls -l <filename>

where <filename> is to be replaced with original filename.

Eg:- ls -l prg1.c

If we do not specify the file name, the output will be generated for all files in current working directory.

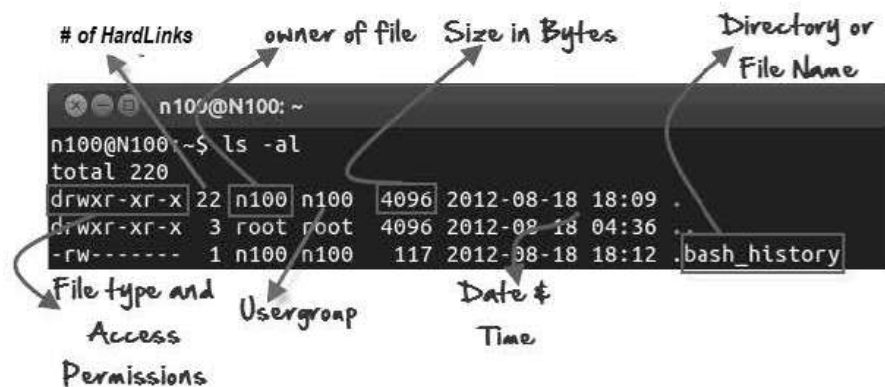


Fig 3: - output of ls -al

Fig 3 shows output of ls -al command, the -a option is used to display hidden files also. As we can see here that it first shows file type (d or -), thereafter it is showing three sets of rwx permissions, corresponding to owner, group and others respectively.

Whichever permission is not granted for that particular file, it is denoted by -, as we can see in Fig 3, file named .bash_history have permissions rw - - - - - which means owner has permission to read and write only, group and other users have no permissions.

To change the permissions of a file, chmod command is used.

Each permission has been assigned a numerical value

read (r) has permission 4

write (w) has permission 2

execute (x) has permission 1

So numerical value corresponding to the permissions of file named .bash_history will be 600. This is because owner has read and

write permission i.e 4+2=6, group and other users have no permissions so 0. To change the permission from 600 to 644(rw- r - - r - -), following command can be given

Schmod 644 .bash_history

- Date and time of last access (atime)
When the file data was last read.

Few actions that will change this access time are: -

- Displaying the contents of the file.
- Copying the file to a new location.
- Editing the file with any text editor (like vi or gedit), even if we don't save any changes.

Some actions that does not changes this value.

- Moving the file to another name or directory in the current file system partition.
- Using redirection to append data to an existing file.
- Date and time of file creation (ctime)
- Date and time of last modification (mtime)

It denotes when the file data is last changed. When a new file is created, this value is initialized. Editing a file and saving the file will update this value. Also, overwriting the file with new data or appending data to an existing file will change the mtime.

- Size of file (size)
Denotes, how much storage is required for the file.
- Number of blocks used by file (block count)
- Number of links
Number of entries, which refers to the same inode.
- Inode modification time

Denotes the time when information in the inode was last changed. The changes occur whenever new hard links to a file is created, or when there are changes in the size of the file.

If there are any changes in the file access timestamp, it does not change the inode modification time.

- Addresses of blocks, where file is physically present/Table of disk addresses

Addresses of disk drive, where data(actual file content) is stored.

It is not necessary that files are stored in contiguous storage, it may be saved in disk blocks scattered all over the storage device. The inode identifies the blocks storing the file's data and the order in which to retrieve them. There is room for 13 addresses or pointers, the first 10 point directly to the actual file content.

If additional storage is required for the file, the 11th file pointer will be used as an indirect pointer (single indirection). It will point to a disk storage block that can be used as a table of additional block pointers.

If file is large and it cannot be stored with single in direction the 12th file pointer will be used as a double indirect pointer (double indirection). It will point to a block that will be converted into a table of additional block pointers. The blocks these pointers point to will also be converted to tables of pointers.

If additional storage is required for the file, the 13th file pointer will be used as a triple indirect pointer (a pointer to a table of pointers that point to tables of pointers that point to other tables of pointers that point to the blocks containing the file).

Data Blocks

They contain the actual file contents. A block of storage can be a part of only one file in the file system. A block cannot be used for storing any other file's content, until the file which is originally stored in that block is deleted.

4.7 Storage of files

As discussed in section 4.6, each inode entry in the inode table has 13 addresses of blocks on storage device. Each address point to block of 1KB (block size) on disk. It seems from here that we can have files of size up to 13 KB only, but in actual it is not so. We can store files much larger than this.

Out of these 13 addresses, the first ten addresses point to 1 KB block. On these addresses, actual file content is stored. So if a file is of 5 KB, it will occupy 5 blocks of 1KB each. The addresses may be in different parts of disk drive and not necessarily contiguous. But with first ten addresses and each pointing to 1 KB block, we would only be able to store a file up to size 10 KB.

To store files of size larger than this, 11th address is used. This address is also of 1 KB block, but at this block actual file content is not stored. This 1KB block contains 256 (1024/4) addresses. Each of these 256 addresses contains address of 1KB block, and on those blocks actual file contents are stored (As shown in Fig 4). So we can store file of size 256 KB, with the help of this single indirection.

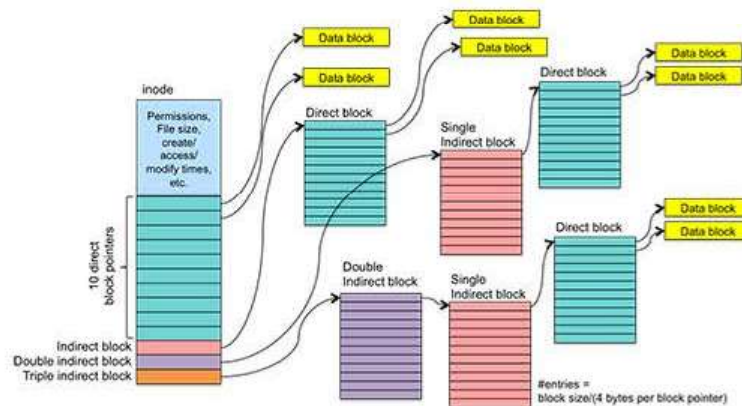


Fig 4: - Storage of files

If file size is still more than this, double indirection is used at 12th address, where 12th address points to block of 256 addresses, each of these addresses points to another 256 addresses. At these addresses are actual file content stored. Therefore by doubly indirection, total numbers of blocks that can be used to store file content are: -

$$256 \times 256 = 65536 \text{ KB} = 64 \text{ MB} (65536/1024)$$

With doubly indirection, file of size 64 MB can be stored.

But what if file size is more than 64MB, so answer is obviously triple indirection at 13th address. So we can have

$$256 \times 256 \times 256 = 16777216 \text{ KB} = 16384 \text{ MB} (16777216/1024) = 16 \text{ GB} (16384/1024).$$

It means that maximum file size, we can have is sum of sizes that can be stored by 13 addresses i.e.

$$10\text{KB} + 256\text{KB} + 64\text{MB} + 16 \text{ GB}$$

(Direct) + (Single Indirection) + (Double Indirection) + (Triple Indirection).

If we still need to store file size larger than this, we can have block size as 2048 bytes.

The question also arises, why we can't have indirection from 1st address. This is because with indirection, access time of file is increased and we would not like to have access time larger for smaller files.

4.8 Self Learning Exercise

Q.1 File is stored in contiguous blocks in storage device. (T/F)

Q.2 Home directory of user is denoted by

(A) % (B) !

(C) ~ (D) #

Q.3 What is doubly indirection?

4.9 LILO, GRUB Boot Loader

A boot loader is a program that places the operating system (OS) into memory. When a computer is powered-up or restarted, the basic input/output system (BIOS) performs some initial tests, and then transfers control to the master boot record (MBR) where the boot loader resides.

For Linux, the two most common boot loaders are LILO (Linux LOader) and GRUB (GRand Unified Bootloader).

LILO

LILO is an acronym for the Linux LOader and has been used to boot Linux on x86 systems for many years. In addition to booting Linux, LILO can boot other operating systems, such as MS-DOS, Windows 95/98, or OS/2. LILO can be installed on the master boot record (MBR) of hard drive or as a secondary boot loader on the Linux partition. LILO consists of several pieces, including the boot loader itself, a configuration file (`/etc/lilo.conf`), a map file (`/boot/map`) containing the location of the kernel, and the lilo command (`/sbin/lilo`), which reads the configuration file and uses the information to create or update the map file and to install the files LILO needs.

GRUB

GNU GRUB (short for GNU GRand Unified Bootloader) is a boot loader package from the GNU Project. GRUB is the reference implementation of the Free Software Foundation's Multi boot Specification, which provides a user the

choice to boot one of multiple operating systems installed on a computer or select a specific kernel configuration available on a particular operating system's partitions.

GRUB provides a simple command line interface as it loads, allowing boot-time changes such as selecting different kernels or initial RAM disks and letting users write new boot sequences.

GRUB is highly portable. It supports multiple executable formats and is geometry-translation independent, including support for logical block addressing (LBA). It also supports all commonly used Linux file systems, and the Windows file systems FAT and NTFS.

4.10 Summary

We have discussed the concept of file system and compared the file system of Linux and Windows family of operating systems. Thereafter, we have also discussed how files are stored in Linux and the underlying concept of single, double and triple indirection. The boot loaders LILO and GRUB were also discussed.

4.11 Glossary

Inode: -Index NODE

The inode is a data structure in a Linux file system that describes a file system object such as a file or a directory. Each inode stores the attributes and disk block location(s) of the file.

LILO: - Linux LOader

A boot loader commonly used to boot Linux for many years. Now a days GRUB boot loader is used, as it is more advanced.

GRUB: -GRand Unified Bootloader

GNU GRUB is a bootloader capable of loading various free and proprietary operating systems. GRUB works well with Linux, DOS, Windows, or BSD.

4.12 Answers to Self-Learning Exercise

Ans.1 F

Ans.2 C

4.13 Exercise

- Q.1 A file system with 300 GB uses a file descriptor with 8 direct block address, 1 indirect block address and 1 doubly indirect block address. The size of each disk block is 128 Bytes and the size of each disk block address is 8 Bytes. The maximum possible file size in this file system is
- (A) 3 KB (B) 35 KB
(C) 350KB (D) 280 Bytes
- Q.2 To find the block size on your file system the command is
- (A) blkisz (B) zblk
(C) chksz (D) cmchk
- Q.3 Each entry in inode table is of size
- (A) 64 KB (B) 32 KB
(C) 32 bytes (D) 64 bytes
- Q.4 The state of the file system is contained in
- (A) Boot block (B) A special block created by Book Block
(C) Super Block (D) None of the above
- Q.5 Which of the following information is not present in an inode
- (A) Contents of the file (B) Name of the file
(C) Size of the file (D) Both (A) and (B)
- Q.6 Which of the following are character special files
- (A) Terminal (B) Printer
(C) Modem (D) All of the above
- Q.7 Among the directory entries, i-node and the file contents, which will be changed when a file is updated?
- (A) Only directory entry and tile contents (B) Only i-node and file contents
(C) All the three (D) None of the above
- Q.8 Explain the term file system. Also, compare Windows and Linux file systems.
- Q.9 What is boot block, superblock and data block?

4.14 Answers to Exercise

Ans.1: B Ans.2:D Ans.3: D Ans.4: C Ans.5: D
Ans.6: D Ans.7: B

References and Suggested Readings

1. Unix Shell Programming by Yashwant Kanetkar; BPB Publication, First Edition 1996 REPRINTED 2009
2. <https://www.youtube.com/watch?v=hZpom8ouYD8>
3. Unix Concepts and Applications by Sumitabha Das; Tata McGraw-Hill, Eighth Reprint 2008
4. Practical Guide to Linux Commands,Editors, and Shell Programming by Mark G. Sobell; Prentice Hall, Second Edition

UNIT-5

Installation

Structure of the Unit

- 5.0 Objective
- 5.1 Installation of Linux System
- 5.2 Using Live CD/DVD/USB
- 5.3 Virtual Machine
- 5.4 Direct Installation
- 5.5 Partitioning of Hard Drive for Linux
- 5.6 Linux Boot Process
- 5.7 Self-Learning Exercise
- 5.8 Summary
- 5.9 Glossary
- 5.10 Answers to Self-Learning Exercise
- 5.11 Exercise
- 5.12 Answers to Exercise

5.0 Objective

This unit provides an overview of the Linux OS (Operating System) installation process, and is not meant to be a comprehensive source of information for installing a Linux OS. In this chapter we shall focus upon the following topics and show you the installation steps for Ubuntu 16.04 LTS.

Installation of Linux System

- Using Live CD/DVD/USB
- Virtual Machine
- Direct Installation
- Partitioning of Hard Drive for Linux
- Init and Run Levels

5.1 Installation of Linux System

The installation of a Linux system is quite different from the installation of Windows OS. This section will provide you installation methods and tasks for installing a supported version of Linux OS on your system.

Planning for Installation of Linux OS: Before starting the installation of Linux system, you have to identify the minimum requirements and verify the following tasks for the installation.

- Upgrade or Install?
- Is Your Hardware Compatible?
- Do You Have Enough Disk Space?
- Partitioning the Hard drive
- Selecting an Installation Boot Method

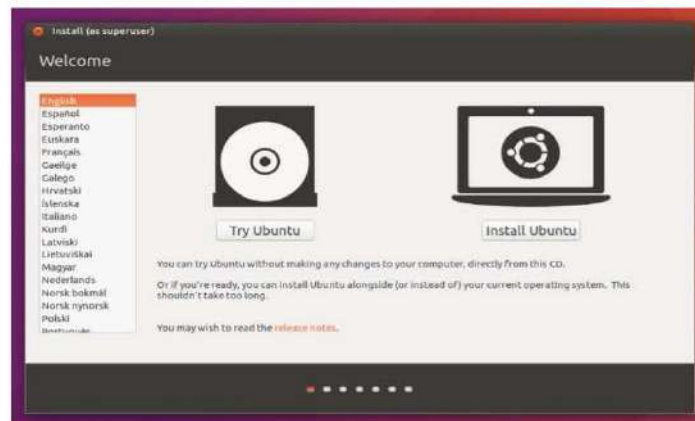
Methods for Installation of Linux OS: Choose from the following list of installation methods to determine the type of installation that you want to do and the information source for the installation.

- Using Live CD/DVD/USB
- Virtual Machine
- Direct Installation

5.2 Using Live CD/DVD/USB Flash Drive

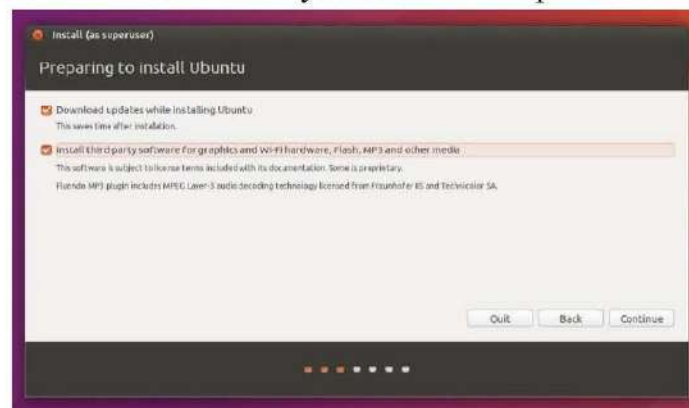
1. Ubuntu 16.04 LTS Installation

It's easy to install Ubuntu from a live installation media like CD/DVD/USB. Download the Ubuntu ISO file from the Ubuntu website and burn the ISO file on a CD/DVD/USB. Put the bootable Ubuntu CD/DVD/USB into the system and restart your computer. You should see a welcome screen prompting you to choose your language and giving you the option to install Ubuntu or try it from the CD/DVD/USB. If your computer doesn't automatically do so, you might need to press the required function key to bring up the boot menu and select the appropriate option.



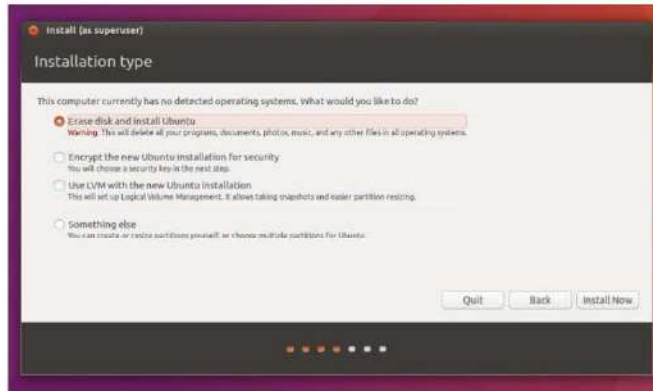
2. Prepare to install Ubuntu

- We recommend you plug your computer into a power source,
- You should also make sure that you have enough space on your computer to install Ubuntu,
- We advise you to select Download updates while installing and Install this third-party software now,
- You should also stay connected to the internet so you can get the latest updates while you install Ubuntu,
- If you are not connected to the internet, you will be asked to select a wireless network, if available. We advise you to connect during the installation so we can ensure your machine is up to date.



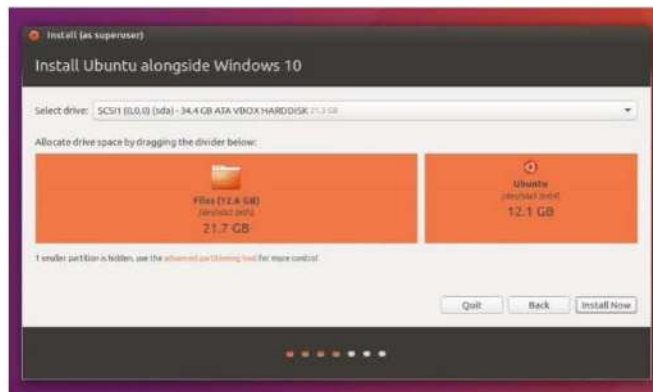
3. Allocate drive space

Use the checkboxes to choose whether you'd like to Install Ubuntu alongside another operating system, delete your existing operating system and replace it with Ubuntu, or - if you're an advanced user - choose the 'Something else' option.



4. **Begin the installation**

Depending on your previous selections, you can now verify that you have chosen the way in which you would like to install Ubuntu. The installation process will begin when you click the Install Now button. Ubuntu needs about 4.5 GB to install, so add a few extra GB to allow for your files.



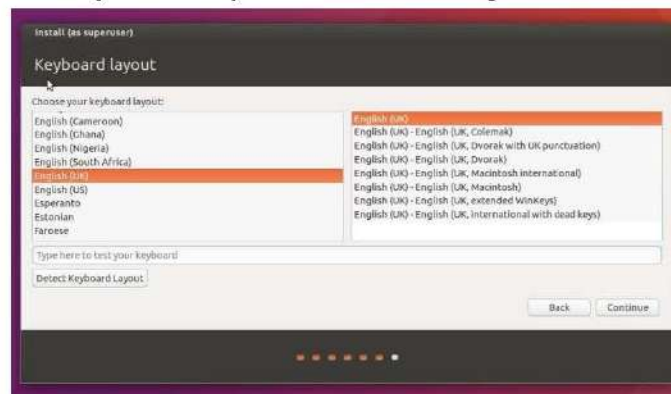
5. **Select your location**

If you are connected to the internet, this should be done automatically. Check your location is correct and click 'Continue' to proceed. If you're unsure of your time zone, type the name of the town you're in or click on the map to select your time zone.



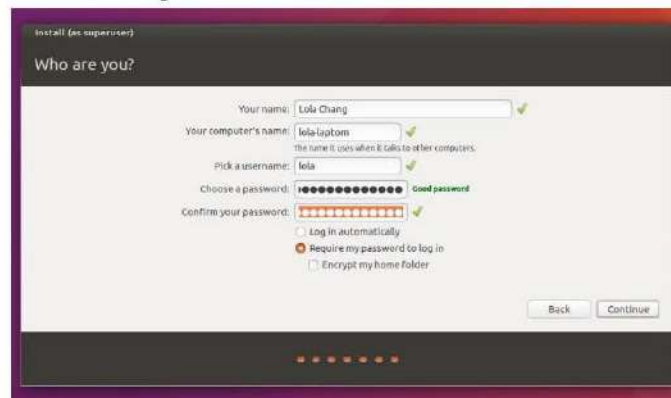
6. Select your preferred keyboard layout

Click on the language option you need. If you're not sure, click the 'Detect Keyboard Layout' button for help.



7. Enter your login and password details

Fill your name, system name, username, password and click 'Continue' to proceed.

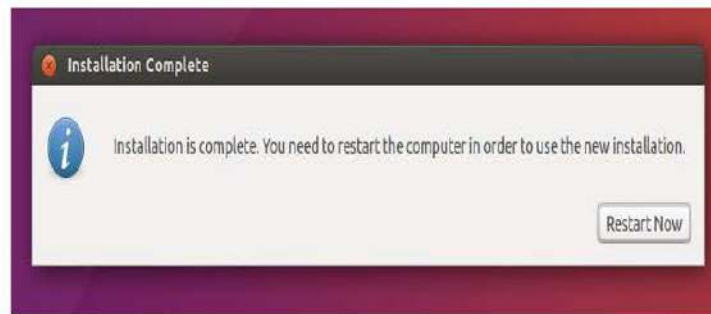


8. Learn more about Ubuntu while the system installs...



9. Restart the system

All that's left is to restart your computer and start enjoying Ubuntu.

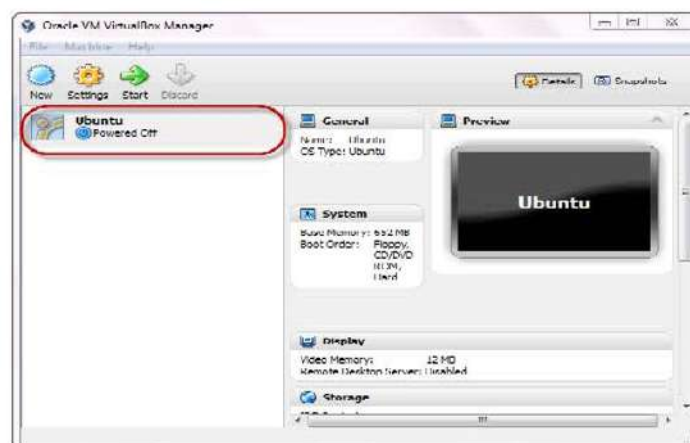


5.3 Virtual Machine

This is a popular method to install a Linux operating system. Virtual installation offers you the freedom of running Linux on an existing OS already installed on your computer. This means if you have Windows running, then you can just run Linux with a click of a button.

Virtual machine software like Oracle VM, VMWare, etc. can install Ubuntu in easy steps. Let us look at Oracle VM.

- Download and install Oracle VM on your computer.
- Click on New on VM tool bar.
- Enter the VM name and select OS type.
- Select the source of media and follow the onscreen instructions to install the Linux OS like Ubuntu.
- You will create Ubuntu configured in your Oracle VM.



Double click on it to run. Ubuntu will load as a separate machine while you are working on Windows.



5.4 Direct Installation

There are some more options to install any Linux OS i.e. NFS, FTP, HTTP, Hard Drive, etc.

NFS – Choosethis option to install or upgrade from an NFS shared directory. In the text field for the NFS server, enter a fully-qualified domain name or IP address. For the NFS directory, enter the name of the NFS directory that contains the installation directory.

For example, if you have boot **CD-ROM** (use the linux ask method or linux repo-nfs:server:options:/path boot option, or the NFSdirectory option on the **InstallationMethod** menu of installation instructions. Note that NFS installations may also be performed in GUI mode.

URL - Choosethis option to install or upgrade from an FTP/HTTP server. In the FTP/HTTP server text field, enter a fully-qualified domain name or IP address. For the FTP/HTTP directory, enter the name of the FTP/HTTP directory that contains the installation directory. If the FTP/HTTP server requires a username and password, specify them as well.

For example, if you have boot CD-ROM (use the linux ask method, linux repo=ftp://user:password@host/path, or linux repo=http://host/path boot option, or linux repo=https://host/path boot option, or the URL option on the InstallationMethod menu for **FTP**, **HTTP**, and **HTTPS** installation instructions.

Hard Drive - Choosethis option to install or upgrade from a hard drive. Hard drive installations require the use of ISO (or CD-ROM) images. Be sure to verify that the ISO images are intact before you start the installation. To verify them, use an md5sum program as well as the linux media check boot option.

Enter the hard drive partition that contains the ISO images (for example, `/dev/hda1`) in the Hard Drive Partition text box. Enter the directory that contains the ISO images in the Hard Drive Directory text box.

For example, if you have boot CD-ROM (use the `linuxaskmethod` or `linux repo=hd:device:/path boot` option), or by selecting `Harddrive` on the `InstallationMethod` menu of installation instructions.

5.5 Partitioning of Hard Drive for Linux

It is possible to get a perfectly functioning Linux system running on a single-partition system, and, in fact, is a bit easier to configure this way, there are a number of benefits from partitioning one or more of your storage devices into multiple partitions; for at least the four main file systems (`root`, `usr`, `home`, and `swap`).

First, it may reduce the time required to perform file system checks (both upon bootup and when doing a manual `fsck`), because these checks can be done in parallel. (By the way, NEVER run an `fsck` on a mounted file system!!! You will almost certainly regret what happens to it. The exception to this is if the file system is mounted read-only, in which case it is safe to do so.) Also, file system checks are a lot easier to do on a system with multiple partitions. For example, if I knew my `/home` partition had problems, I could simply unmount it, perform a file system check, and then remount the repaired file system (as opposed to booting my system with a rescue diskette into single-user mode and doing the repairs).

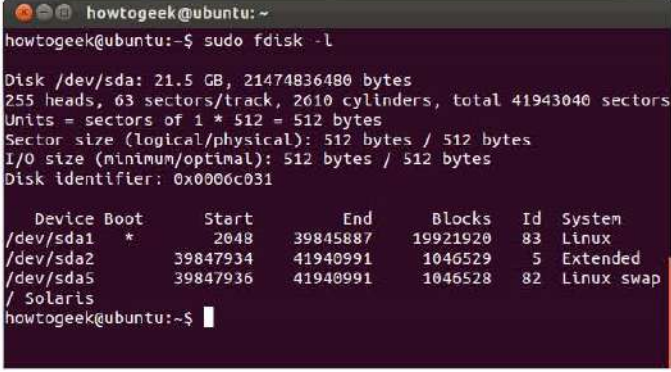
Second, with multiple partitions, you can, if you wish, mount one or more of your partitions as read-only. For example, if you decide that everything in `/usr` will not be touched even by `root`, you can mount the `/usr` partition as read-only.

Third & finally, the most important benefit that partitioning provides is protection of your file systems. If something should happen to a file system (either through user error or system failure), on a partitioned system you would probably only lose files on a single file system. On a non-partitioned system, you would probably lose them on all file systems.

The `fdisk` command is a text-based utility for viewing and managing hard disk partitions on Linux. It's one of the most powerful tools you can use to manage partitions, but it's confusing to new users. On Ubuntu, Linux Mint or other Ubuntu-derived distributions, the `fdisk` and `mkfs` commands must be prefixed

with `sudo`. On distributions that don't use `sudo`, use the `su -` command first to get a root shell, then type every command without `sudo`.

1. **List Partition:** The `sudo fdisk -l` commands lists the partitions on your system.

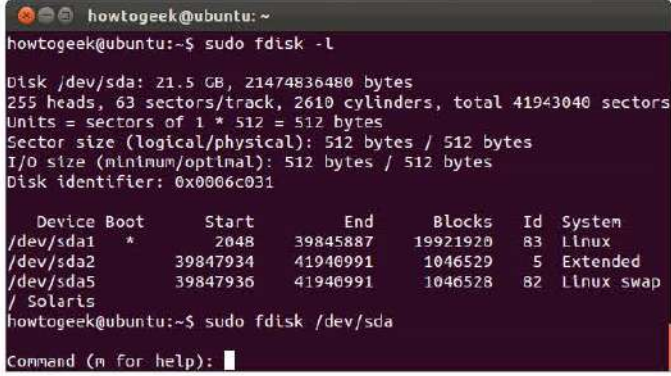


```
howtogeek@ubuntu:~$ sudo fdisk -l
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0006c031

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *         2048         39845887   19921920   83  Linux
/dev/sda2           39847934   41940991    1046529    5  Extended
/dev/sda5           39847936   41940991    1046528    82  Linux swap
/ Solaris
howtogeek@ubuntu:~$
```

You can add a disk's device name to list only partitions on it. For example, use the following command to only list partitions on the first disk device: `sudo fdisk -l /dev/sda`

2. **Enter into Command Mode:** To work on a disk's partitions, you have to enter command mode. You'll need the device name of a disk from the `fdisk -l` command. The following command enters command mode for the first disk device: `sudo fdisk /dev/sda`



```
howtogeek@ubuntu:~$ sudo fdisk -l
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 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 identifier: 0x0006c031

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *         2048         39845887   19921920   83  Linux
/dev/sda2           39847934   41940991    1046529    5  Extended
/dev/sda5           39847936   41940991    1046528    82  Linux swap
/ Solaris
howtogeek@ubuntu:~$ sudo fdisk /dev/sda
Command (m for help):
```

Don't edit partitions while they're in use. If you want to edit system partitions, boot from a live CD first.

3. **Using Command Mode:** In command mode, you use single-letter commands to specify actions you want to take. Type `m` and press Enter to see a list of the commands you can use.

```

howtogeek@ubuntu: ~
Command (m for help): m
Command action
a toggle a bootable flag
b edit bsd disklabel
c toggle the dos compatibility flag
d delete a partition
l list known partition types
n print this menu
n add a new partition
o create a new empty DOS partition table
p print the partition table
q quit without saving changes
s create a new empty sun disklabel
t change a partition's system id
u change display/entry units
v verify the partition table
w write table to disk and exit

```

4. **View the Partition Table:** Use **p** to print the current partition table to the terminal from within command mode.
5. **Delete a Partition:** Use the **d** command to delete a partition. You'll be asked for the number of the partition you want to delete, which you can get from the **p** command. For example, if you want to delete the partition at `/dev/sda5`, you'd type **5**. After deleting the partition, you can type **p** again to view the current partition table. The partition appears deleted, but **fdisk** doesn't write these changes to disk until you use the **w** command.
6. **Create a Partition:** Use the **n** command to create a new partition. You can create a logical or primary partition (**l** for logical or **p** for primary). A disk can only have four primary partitions.

```

howtogeek@ubuntu: ~
Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *          2048     39845887   19921920   83  Linux
/dev/sda2                39847934   41940991    1046529    5  Extended

Command (m for help): n
Command action
  l  logical (5 or over)
  p  primary partition (1-4)
l
First sector (39849982-41940991, default 39849982):
Using default value 39849982
Last sector, +sectors or +size[K,M,G] (39849982-41940991, default 41940991):
Using default value 41940991
Command (m for help):

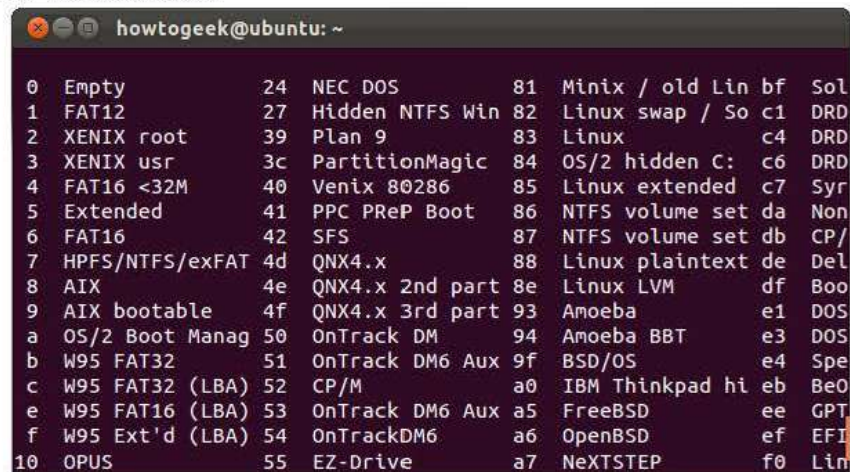
```

Next, specify the sector of the disk you want the partition to start at. Press **Enter** to accept the default sector, which is the first free sector on the disk.

Last, specify the last sector of the partition on the disk. If you want to use up all available space after the initial sector, just press **Enter**. You can also specify a specific size, such as **+5G** for a five gigabyte partition or **+512M** for a 512 megabyte partition. If you don't specify a unit after

the + sign, fdisk uses sectors as the unit. For example, +10000 results in the end of the partition being 10000 sectors after its beginning.

7. **System ID:** The `n` command I just ran recreated the swap partition you deleted earlier or did it? If you run the `p` command again, you'll see that the new `/dev/sda5` partition is a "Linux" partition instead of a "Linux swap" partition. If you want to change its type, you can use the `t` command and specify the partition's number. You'll be asked for the hex code of the type. You don't know it, so you can type `L` to view a list of hex codes.



```
howtogeek@ubuntu: ~  
0 Empty 24 NEC DOS 81 Minix / old Lin bf Sol  
1 FAT12 27 Hidden NTFS Win 82 Linux swap / So c1 DRD  
2 XENIX root 39 Plan 9 83 Linux c4 DRD  
3 XENIX usr 3c PartitionMagic 84 OS/2 hidden C: c6 DRD  
4 FAT16 <32M 40 Venix 80286 85 Linux extended c7 Syr  
5 Extended 41 PPC PReP Boot 86 NTFS volume set da Non  
6 FAT16 42 SFS 87 NTFS volume set db CP/  
7 HPFS/NTFS/exFAT 4d QNX4.x 88 Linux plaintext de Del  
8 AIX 4e QNX4.x 2nd part 8e Linux LVM df Boo  
9 AIX bootable 4f QNX4.x 3rd part 93 Amoeba e1 DOS  
a OS/2 Boot Manag 50 OnTrack DM 94 Amoeba BBT e3 DOS  
b W95 FAT32 51 OnTrack DM6 Aux 9f BSD/OS e4 Spe  
c W95 FAT32 (LBA) 52 CP/M a0 IBM Thinkpad hi eb Be0  
e W95 FAT16 (LBA) 53 OnTrack DM6 Aux a5 FreeBSD ee GPT  
f W95 Ext'd (LBA) 54 OnTrackDM6 a6 OpenBSD ef EFI  
10 OPUS 55 EZ-Drive a7 NeXTSTEP f0 Lin
```

It says `82` is the code for Linux swap partitions, so you can type that. This doesn't format the partition with the file system you select. You'll have to do that later with the appropriate `mkfs` command.

8. **Writing Change:** Use `w` to write the changes you've made to disk. Use `q` if you want to quit without saving changes.
9. **Format a Partition:** You must format new partitions with a file system before you can use them. You can do this with the appropriate `mkfs` command. For example, this command formats the fifth partition on the first disk with the `ext4` file system.

```
sudo mkfs.ext4 /dev/sda5
```

Use the `mkswap` command if you want to format a partition as a swap partition.

```
sudo mkswap /dev/sda5
```

disk contains a variety of other commands, including expert commands you can access by running the `x` command first. Explore `fdisk`'s manual page for more detailed information.

5.6 Linux Boot Process

After few moments you see the Linux login prompt, when you press the power button of your system. Have you ever wondered what happens behind the scenes from the time you press the power button until the Linux login prompt appears? The following are the 6 high level stages of a typical Linux boot process.

BIOS (Basic Input/Output System)	Execute MBR
MBR (Master Boot Record)	Execute GRUB
GRUB (Grand Unified Bootloader)	Execute Kernel
Kernel	Execute /sbin/init
Init	Execute Run Level Program
Run Level	Executed from /etc/rc.d/rc*.d/

- 1. BIOS:** The BIOS program is written into permanent read-only memory and is always available for use. The BIOS provides the lowest level interface to peripheral devices and controls the first step of the boot process. The BIOS tests the system, checks peripherals and then looks for a drive to use to boot the system. Usually it looks for boot loader in floppy, CD-ROM, or hard drive. You can press a key (typically F12 or F2, but it depends on your system) during the BIOS startup to change the boot sequence. Once Linux is installed on the hard drive of a system, the BIOS looks for a Master Boot Record (MBR) starting at the first sector on the first hard drive, loads its contents into memory, then passes control to it.
- 2. MBR:** It contains instructions on how to load the GRUB (or LILO) boot-loader, using a pre-selected operating system. MBR is less than 512 bytes and located in the 1st sector of the bootable disk, typically /dev/hda, or /dev/sda. If boot-loader is installed in the MBR then loads

the boot-loader which takes over the process. In the default Linux configuration, GRUB uses the settings in the MBR to display boot options in a menu. Once GRUB has received the correct instructions for the operating system to start, either from its command line or configuration file, it finds the necessary boot file and hands off control of the machine to that operating system.

3. **GRUB:** This boot method is called *direct loading* because instructions are used to directly load the operating system, with no intermediary code between the boot-loaders and the operating system's main files (such as the kernel). If you have multiple kernel images installed on your system, you can choose which one to be executed. The Microsoft operating systems, as well as various other proprietary operating systems, are loaded using a chain loading boot method. GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file. Grub configuration file is `/boot/grub/grub.conf` (`/etc/grub.conf` is a link to this).
4. **Kernel:** It Mounts the root file system as specified in the "root=" in `grub.conf` and executes the `/sbin/init` program. Since `init` was the 1st program to be executed by Linux Kernel, it has the process identification number of 1 (PID). Initial RAM Disk (`initrd`) is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.
5. **Init:** When `init` starts, it becomes the parent or grandparent of all of the processes that start up automatically on your Linux system. This looks at the `/etc/inittab` file to decide the Linux run level. This instructs `init` to read an initial configuration script for the environment, which sets the path, starts swapping, checks the file systems, and so on. Basically, this step takes care of everything that your system needs to have done at system initialization: setting the clock, initializing serial ports and so forth. Then `init` continues to read the `/etc/inittab` file, which describes how the system should be set up in each run level and sets the default *run level*.

After having determined the default run level for your system, `init` starts all of the background processes necessary for the system to run by looking in the appropriate `rc` directory for that run level. `init` runs each of the kill scripts (their file names start with a K) with a stop parameter. It then runs all of the start scripts (their file names start with an S) in the appropriate run level directory so that all services and applications are started correctly. In fact, you can execute these same scripts manually after the system is finished booting with a command like `/etc/init.d/httpdstop` or `service httpd stop` logged in as `root`, in this case stopping the web server.

After `init` has progressed through the run levels to get to the default run level, the `/etc/inittab` script forks a `getty` process for each virtual console (login prompt in text mode). `getty` opens tty lines, sets their modes, prints the login prompt, gets the user's name, and then initiates a login process for that user. This allows users to authenticate themselves to the system and use it. The `inittab` file can also tell `init` how it should handle a user pressing `Ctrl+Alt+Delete` at the console. As the system should be properly shut down and restarted rather than immediately power-cycled, `init` is told to execute the command `/sbin/shutdown -t3 -r now`, for instance, when a user hits those keys. In addition, `inittab` states what `init` should do in case of power failures, if your system has a UPS unit attached to it.

On most RPM-based systems the graphical login screen is started in run level 5, where `/etc/inittab` runs a script called `/etc/X11/prefdm`. The `prefdm` script runs the preferred X display manager, based on the contents of the `/etc/sysconfig/desktop` directory. This is typically `gdm` if you run GNOME or `kdm` if you run KDE, but they can be mixed, and there's also the `xdm` that comes with a standard X installation. On Debian, for instance, there is an `initscript` for each of the display managers, and the content of the `/etc/X11/default-display-manager` is used to determine which one to start. The `/etc/default` and/or `/etc/sysconfig` directories contain entries for a range of functions and services, these are all read at boot time.

6. **Run Level Program:** A run level program is a configuration of processes. Depending on your default `init` level setting, the system will

execute the programs from one of the following available run level directories. Execute 'grep initdefault /etc/inittab' or 'who -r' on your system to identify the default run level. If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that. Typically you would set the default run level to either 3 or 5.

- 0 – halt – /etc/rc.d/rc0.d/
- 1 – Single user mode – /etc/rc.d/rc1.d/
- 2 – Multiuser, without NFS – /etc/rc.d/rc2.d/
- 3 – Full multiuser mode – /etc/rc.d/rc3.d/
- 4 – unused – /etc/rc.d/rc4.d/
- 5 – X11 – /etc/rc.d/rc5.d/
- 6 – reboot (Do NOT set initdefault to this) – /etc/rc.d/rc6.d/

Please note that there are also symbolic links available for these directory under /etc directory. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d. Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K. Programs starts with S are used during startup. S for startup. Programs starts with K are used during shutdown. K for kill. There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed. For example, S12syslog is to start the syslog daemon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

5.7 Self-Learning Exercise

- Q.1 Architecture of Ubuntu is
- a) RPM
 - b) Debian
 - c) EXE
 - d) None of these
- Q.2 NFS stand for
- a) Not Found Suitable
 - b) No File System
 - c) Network File System
 - d) Network Found Suitable
- Q.3 Which command is required to run a command as root in Ubuntu?
- a) sudo
 - b) su

- c) Anyone of these d) None of these
- Q.4 Which command is use to format the partition?
- a) fdisk b) mkfs
- c) mkswap d) None of these

5.8 Summary

The installation of a Linux system is quite different from the installation of Windows OS. There are different installation methods like live CD, DVD, USB, Virtual and Direct installation of Linux system.

There are a number of benefits from partitioning one or more of your storage devices (Hard Drive) into multiple partitions; for at least the four main file systems (root, usr, home, and swap).

When init starts, it becomes the parent or grandparent of all of the processes that start up automatically on your Linux system. The /etc/inittab file instructs init to read an initial configuration script for the environment, which sets the path, starts swapping, checks the file systems, and so on.

A run level program is a configuration of processes. Depending on your default init level setting, the system will execute the programs from one of the available run levels.

5.9 Glossary

Virtual Machine – Tool to run a System on to another Operating System.

Partition – Divide the Hard Drive into multiple sub drive.

Boot Process - When you press the power button of your system after few moments you see the Linux login prompt; this whole process is known as Linux Boot Process.

5.10 Answers to Self-Learning Exercise

- Q.1 (b) Q.2 (c)
- Q.3 (a) Q.4 (b)

5.11 Exercise

- Q.1 Can we install any OS using Virtual Machine?
- a) Yes b) No
- c) Only Windows d) Only Linux

- Q.2 Selection of hard drive is done during installation before
 a) Time zone/Location selection b) Keyboard type selection
 c) Preparing to install Ubuntu d) None of these
- Q.3 The/etc/inittabscript forks a process
 a) tty b) getty
 c) GNOME Desktop d) shutdown
- Q.4 In partitioning, *m* is used to
 a) Enter into command mode b) See the list of commands
 c) Move the partition d) None of these
- Q.5 Total high level stages in Linux boot process.
 a) 4 b) 5
 c) 6 d) 3
- Q.6 How many run level programs are there?
 a) 7 b) 5
 c) 3 d) 6

5.12 Answers of Exercise

- | | |
|----------|----------|
| Q. 1 (a) | Q. 2 (a) |
| Q. 3 (b) | Q. 4 (b) |
| Q. 5 (c) | Q. 6 (d) |

References and Suggested Readings

1. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
2. UNIX Shell Programming by Yashwant Kanetkar, BPB Publications, 2009.
3. <http://www.thegeekstuff.com/2011/02/linux-boot-process>
4. <https://www.ubuntu.com/download/desktop/install-ubuntu-desktop>

UNIT-6

Getting started with Linux

Structure of the Unit

- 6.0 Objective
- 6.1 Introduction
- 6.2 Login Process
- 6.3 Creating user accounts and groups in Linux
- 6.4 Getting help
- 6.5 Services and processes
- 6.6 Self Learning Exercises
- 6.7 Summary
- 6.8 Glossary
- 6.9 Answers to Self Learning Exercises
- 6.10 Exercises
- 6.11 Answers to Exercises

6.0 Objective

In this chapter we shall focus upon the following topics

- Login process
- Creating user accounts and groups in Linux
- Getting help in Linux
- Services and processes

6.1 Introduction

Linux provides the environment to create users and groups using command line interface and graphical user mode as well. Valid users can log into the system and perform different tasks depending on the type of user-account. Account

may be standard that doesn't have much privilege to perform administrative task. User-account created as administrator may perform all the administrative actions. There is a concept of groups to share some common things among a group of users. Linux also provides the help about the usage and other details of commands and packages installed on the system. We can manage services and processes running in the system using commands in Linux.

6.2 Login process

Normally all Linux distributions provide the option for users to login into the system. In Figure 6.1, login page is shown for a very popular Linux distribution, Ubuntu, these days.

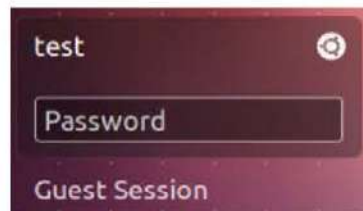


Figure 6.1

By default, there are two users, first user that is created while installing the Linux operating system and second one is guest. The name of user is 'test' in Figure 6.1 that was created during installation, has powers to perform administrative tasks. Guest user has very limited privileges and contents created by him are usually lost once logged out.

Now let us see how login process works. When the kernel (core of operating system) is first loaded into memory, it initializes itself and any hardware that are attached to the computer. The first program run by the kernel is called 'init' whose pid is 1; its job is to function as the ancestor of all processes.

One of the responsibilities of 'init' is to start a process called 'getty', that takes care of complete login process. The getty process initiates login command and gives users with login prompt display on the terminal screen and waits for the user to enter loginname. After entering the loginname, the user gets the prompt for login password. The password that user types are not echoed to the screen. Now getty checks user credentials by verifying it with `/etc/passwd` and `/etc/shadow` file, if password is found to be correct it will initiate gathering user

properties else getty will terminate login process and re-initiates once again with new login: prompt. This is done for three times in most Linux flavors. If user failed to enter correct password for three consecutive times, getty disables terminal for 10 seconds to control unauthorised logins. In case of successful login getty process reads user properties and shell related settings and then presents prompt for user to execute their commands. After that, user can start executing their commands at the terminal and perform any task in GUI mode.

6.3 Creating user accounts and groups in Linux

Creating user accounts and groups is simple in Linux. Let us see how we can create user accounts using commands. We can use 'adduser' or 'useradd' command on terminal to create user accounts. adduser is more user friendly and interactivethan its back-end useradd. It sets up the account's home directory and other settings, whereas useradd just creates the user. Let's use the command `ls /home` to see the existing human users on the system

```
$ ls/home
```

```
amresh test1
```

Here, there are two existing users shown above in the output. User with name 'amresh' is administrator (that has privileges to do some administrative tasks) and we are assuming that we are logged onto the system with this login. Now we use adduser command to create a new user-account with login name test2. Since creating user-accounts is an administrative task, so we shall prefix sudo to run this command. Actaually , in order to perform any administrative task we need to use sudo.

```
$ sudoadduser test2
```

```
[sudo] password for amresh:
```

```
Adding user `test2' ...
```

```
Adding new group `test2' (1005) ...
```

```
Adding new user `test2' (1002) with group `test2' ...
```

```
Creating home directory `/home/test2' ...
```

```
Copying files from `/etc/skel' ...
```

```
Enter new UNIX password:
```

```
Retype new UNIX password:
```

```
Sorry, passwords do not match
```

```
passwd: Authentication token manipulation error
```

```
passwd: password unchanged
Try again? [y/N] y
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for test2
Enter the new value, or press ENTER for the default
Full Name []: Test Kumar
Room Number []: 20
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] y
$
```

In the above snippet , after checking with the correct password of administrator, system creates a new user with name test2 (with userid =1002) and a new group with name test2 (with groupid-1005).Then system adds the newly created user test2 in the newly created group with the same name. Afterwards, home directory for the user is being created. The *'/etc/skel'* directory contains files and directories that are automatically copied over to a new user's *home directory* when such user is created by the *adduser* command. Now system shows the prompt to enter the password to be set for the new user. If the system doesn't get the matching password while administrator retypes the password, it will ask the administrator if he wants to retry. After setting the password successfully, administrator may enter details of the user or may skip it. Then after getting the confirmation , system updates these details and shows primary prompt that indicates the terminal is ready to accept commands.

Now let us run the command 'ls /home' to list out human users.

```
$ ls /home
amresh test1 test2
```

Here we can see a new user-account has been created with home directory as *'/home/test2'* .We can also create user-accounts in **GUI**(Graphical User Interface) mode. Here we shall see this process in case of Ubuntu. First we shall go to *'system settings'* that can be easily found on the right-top corner of

desktop. Now we choose 'User Accounts' option to enter into a GUI tab for managing user-accounts.



Figure 6.2

The screenshot, as shown in Figure 6.2, is taken with existing three user-accounts in the system. For authentication, we need to click on 'unlock' button and enter the password of the user with which we are logged in. Then we click on '+' button to add a new user. Account type of the new user may be **Standard** or **Administrator**. After entering full name and username of the user to be added, we click on Add button. This can be easily visualized by Figure 6.3.

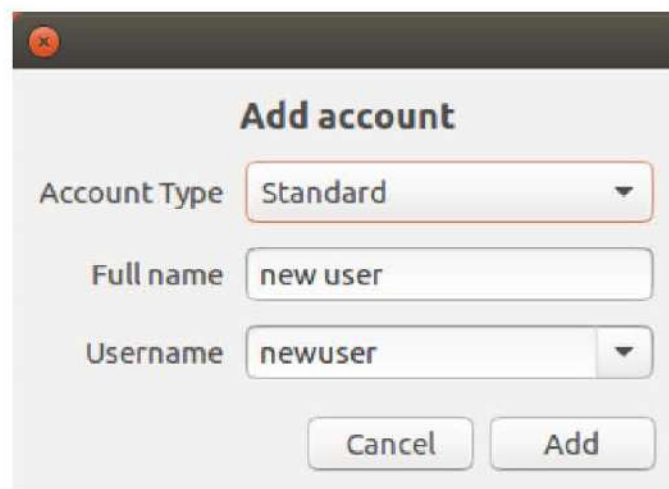


Figure 6.3

Still **newuser** is not ready to use, we are required to enable it by assigning password or allowing it to enable the account without the password. To do so we choose 'Account disabled' and we provide the password to be set. Figure 6.4 clearly shows this step.



Figure 6.4

In most of Linux distributions, there are three basic types of Linux user accounts: **administrator** (root), **regular(standard)**, and **service**. The Linux administrative root account is automatically created when we install Linux, and it has administrative privileges for all services on Linux Operating System. The root account is also known as super user. Regular users have the necessary privileges to perform standard tasks on a Linux computer such as running word processors, databases, and Web browsers. They can store files in their own home directories. Since regular users do not normally have administrative privileges, they cannot accidentally delete critical operating system configuration files. Services such as Apache, mail, games, and printing have their own individual service accounts. These accounts exist to allow each of these services to interact with our computer.

Each user on a Linux system is assigned a unique user identification number, also known as a UID.

A user group is a group of one or more users. A user can be a member of more than one group. Normally, when a user is added, a primary group is created—meaning that a user group of the same name is created and that the new user is the sole user in that group.

Ubuntu is one of the few Linux distributions in which the root account is disabled. If we want to do something with administrative permission on the console we have to type “**sudo**” before the command.“sudo” means superuser do. “sudo” will prompt for “Password:”. After entering correct password, the user can perform intended task.

Creating a new group - Now we shall see how to create a new group. This can be easily done by the use of user-friendly command 'addgroup'. We can also use 'groupadd' that is usually preferable for scripting.

```
$ sudoaddgroup g1
[sudo] password for amresh:
Adding group `g1' (GID 1006) ...
Done.
```

Here we have created a new group called g1. Each group has been assigned a unique group id (GID). GID of the newly created group is 1006. This is also possible in GUI mode. We can use the tool 'Users Settings' in Ubuntu and similar types of tools are available in other Linux distributions. It can be easily accessed using Unity Dash, located on the top-left, in Ubuntu. To add a new group and manage the group members of a group we can select 'Manage Groups' and then 'add' in next pop-up window as shown in Figure 6.5.



Figure 6.5

Here we have created a new group called group2 with group ID, 1008. We can also manage which users can be members of that group. There is an important built-in group 'sudo' in Ubuntu, which all administrators belong to. All users of this group can use sudo command to perform administrative tasks. We shall see some more details about user management and groups in next chapter.

6.4 Getting help

In this section, we shall study about how to get help about the commands and the programs installed in Linux. It is very difficult to keep all the details about commands in mind. We shall use some commands like type, which, man, info, whatis to identify and show the details about commands.

'type' command - It displays the kind/type of command the shell will execute, given a particular command name. Let us examine some commands.

```
$ type cd
cd is a shell builtin
$ type cat
cat is hashed (/bin/cat)
```

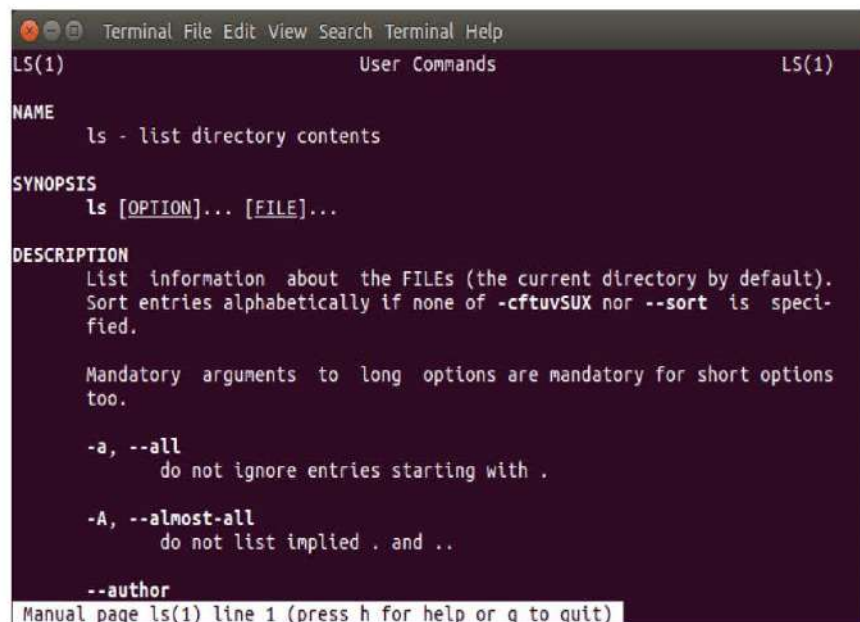
Here type of 'cd' command is shell built-in, type of 'cat' command is external binary file mapped to location /bin/cat.

'which' command- This command is used to locate executables in the system. It allows user to pass a command name as argument to get its path in the system. 'which' commands searches the path of executable in system paths set in \$PATH environment variable.

```
$ which java
/usr/bin/java
```

Here executable file of Java is located at /usr/bin/java.

'man' command - The **man** command shows detailed manuals for each command. These are referred to as "man pages." This is similar to the manual of some gadget or device. For example, if we want to view the manual page for the **ls** command, we'd type **man ls**. The output also shows the usage of all options that can be used with **ls** as shown in Figure 6.6.



```
Terminal File Edit View Search Terminal Help
LS(1) User Commands LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
DESCRIPTION
  List information about the FILES (the current directory by default).
  Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
  fied.

  Mandatory arguments to long options are mandatory for short options
  too.

  -a, --all
      do not ignore entries starting with .
  -A, --almost-all
      do not list implied . and ..
  --author
```

Figure 6.6

We can scroll down the pages by pressing the spacebar or by using the up and down arrow keys. To quit from the manual page, we can press 'q'.

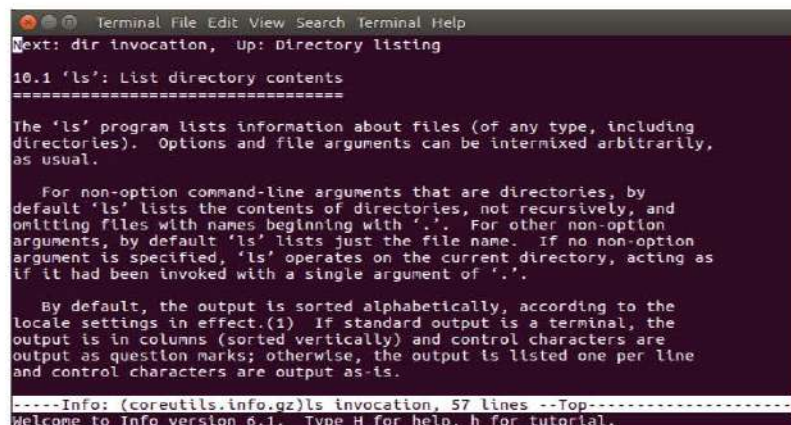
'**whatis**' command- **whatis** command shows a one-line summary of a command, taken from its manual page. It's a quick way of seeing what a command actually does. Here we use **whatis** command to see one-line description about **cat** command.

```
$ whatis cat
```

```
cat (1) - concatenate files and print on the standard output
```

info - reads documentation in the info format. It is similar to **man**, with a more robust structure for linking pages together. Info pages are made using the texinfo tools, and can link with other pages, create menus and ease navigation in general.

The default location of info documentation is `/usr/share/info`. Let us see the output of command '**info ls**' in Figure 6.7.



```
Terminal File Edit View Search Terminal Help
Next: dir invocation, Up: Directory listing
10.1 'ls': List directory contents
=====
The 'ls' program lists information about files (of any type, including
directories). Options and file arguments can be intermixed arbitrarily,
as usual.

For non-option command-line arguments that are directories, by
default 'ls' lists the contents of directories, not recursively, and
omitting files with names beginning with '.'. For other non-option
arguments, by default 'ls' lists just the file name. If no non-option
argument is specified, 'ls' operates on the current directory, acting as
if it had been invoked with a single argument of '.'.

By default, the output is sorted alphabetically, according to the
locale settings in effect.(1) If standard output is a terminal, the
output is in columns (sorted vertically) and control characters are
output as question marks; otherwise, the output is listed one per line
and control characters are output as-is.

-----Info: (coreutils.info.gz)ls invocation, 57 lines --Top-----
Welcome to Info version 6.1. Type H for help, h for tutorial.
```

Figure 6.7

6.5 Services and processes

Let's first understand the meaning of services and processes. In Linux a service is just another name for a daemon, which is a client / server application that runs in the background. A service is continuously listening for incoming requests and sends a response based on the request given. A process is simply an application or a script which can be running in the foreground or the background. '**ps**' command can be used with various options to check the status of running processes and services. We can also use '**top**' command to show the

running processes and resources utilized by those processes. Following snapshot (shown in Figure 6.8) shows the output of top command at some particular time.

```

top - 12:02:53 up 2:42, 1 user, load average: 0.76, 0.82, 0.75
Tasks: 234 total, 2 running, 232 sleeping, 0 stopped, 0 zombie
%Cpu(s): 20.0 us, 2.2 sy, 0.0 ni, 77.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4120376 total, 410188 free, 1816460 used, 1893728 buff/cache
KiB Swap: 4168700 total, 4168700 free, 0 used, 1791052 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
16249 amresh    20   0 1170212 561812 109768 R   80.7  13.6  42:45.63 Web Content
16089 amresh    20   0 1056332 359512 130204 S   10.0   8.7  16:46.71 Firefox
 895 message+ 20   0   7012   4700   3276 S    0.3   0.1   0:10.49 dbus-daemon
 981 root      20   0  101840  18888  12752 S    0.3   0.5   0:09.17 NetworkMan+
1111 root      20   0   241732 115328  55860 S    0.3   2.8  12:01.32 Xorg
17470 amresh    20   0  125012  32780  27884 S    0.3   0.8   0:04.51 gnone-term+
   1 root      20   0    24228   5216   3720 S    0.0   0.1   0:02.56 systemd
   2 root      20   0     0     0     0 S    0.0   0.0   0:00.00 kthreadd
   3 root      20   0     0     0     0 S    0.0   0.0   0:00.45 ksoftirqd/0
   5 root      0  -20     0     0     0 S    0.0   0.0   0:00.00 kworker/0:+
   7 root      20   0     0     0     0 S    0.0   0.0   0:07.25 rcu_sched
   8 root      20   0     0     0     0 S    0.0   0.0   0:00.00 rcu_bh
   9 root      rt   0     0     0     0 S    0.0   0.0   0:00.04 nigration/0
  10 root      rt   0     0     0     0 S    0.0   0.0   0:00.04 watchdog/0
  11 root      rt   0     0     0     0 S    0.0   0.0   0:00.04 watchdog/1
  12 root      rt   0     0     0     0 S    0.0   0.0   0:00.04 nigration/1
  13 root      20   0     0     0     0 S    0.0   0.0   0:00.06 ksoftirqd/1

```

Figure 6.8

‘service’ is a command which allows us to start, stop or restart services running in the background. We can start the apache service httpd as shown below

```
$ servicehttpd start
```

Stopping and changing priority of processes/services – To stop a process, the user can use kill command with its pid (unique identifier assigned to a process). For instance , to kill a process with pid 2673 we can use the command

```
$ kill 2673
```

kill command can be used with -9 option to kill a process forcefully ,but this should be used cautiously so as not to kill any important system process.

Every running process in Linux has a priority assigned to it. We can change the process priority using nice command. Linux Kernel schedules the processes and allocates CPU time accordingly for each of them. But, when a process requires higher priority to get more CPU time, nice command can be used to increase the priority.

The process scheduling priority range is from -20 to 19. We call this as nice value. A nice value of -20 represents highest priority, and a nice value of 19 represents least priority for a process. By default when a process starts, it gets the default priority of 0.

```
$ nice -10 p1
```

The above command increases the priority of process, p1 by 10 and command used in following manner will reduce the priority of process, p2 by 10 .

```
$ nice --10 p2
```

We cannot use nice command to change the priority of an already running process. In order to change the priority of an already running process we can use "renice" command. Following command will increase the priority of a running process having pid 2345

```
$ renice -5 -p 2354
```

6.6 Self Learning Exercise

- Q.1 Which one of the following options is correct-
- a) 'password' command is used to change the password of a user.
 - b) 'passwd' command is used to change the password of a user.
 - c) First process created by kernel is 'login'.
 - d) 'ps' command is used to kill a process.
- Q.2 'which' command is used
- a) to locate executables in the system.
 - b) to install a package in Linux.
 - c) to uninstall a package in Linux.
 - d) to find out what users can use that particular command.
- Q.3 To change the priority of a process, which one of following commands is used-
- a) ps.
 - b) type.
 - c) nice.
 - d) ls.

6.7 Summary

In this chapter we learnt how login process works in Linux. Kernel creates init process that further creates getty process. Process getty takes care of showing prompt for user to enter user-name and password. After authentication, the user is allowed to enter into the system and perform intended tasks as per the type of

user. New users are created using command 'adduser' and new groups are created using command 'addgroup'. This can also be done in graphical user mode in Linux. Linux gives the facility for getting help about commands and details about the packages installed on the system. Some of the commands used for getting help are man, type, which, whatis etc. Commands are available in Linux to manage the services and processes running in the system. We can start a process/service, stop and prioritize these utilities as per our requirements.

6.8 Glossary

Adduser	-	command to add a new user in Linux.
Addgroup	-	command to add a new group in Linux.
Man	-	shows manual page about a command.
Nice	-	command to change the priority of a process.
Getty	-	process that takes care of complete login process
Init	-	first process started during booting of the system

6.9 Answers to Self-Learning Exercise

Q.1 (b)

Q.2 (a)

Q.3 (c)

6.10 Exercise

- Q. 1 'whatis' command is used to
- display short description about the command.
 - display detailed description about the command.
 - display same information exactly same as the output of 'man' command.
 - modify the manual page of the command.
- Q.2 Command 'renice -10 -p 2546' will do
- restart the process with pid 2546.
 - update the pid of a process to 10.

- c) decrease the priority of a process with pid 2546 by 10.
 - d) increase the priority of a process with pid 2546 by 10.
- Q.3 Which one of processes is first process run by the kernel?
- a) getty
 - b) login
 - c) init
 - d) fork
- Q.4 Command used to kill a process with pid 2545 -
- a) killp 2545
 - b) kill p 2545
 - c) stopp 2545
 - d) kill 2545
- Q.5 'getty' process
- a) takes care of complete login process.
 - b) helps other processes find their ppid.
 - c) gets the return status of other processes.
 - d) receive the return status of jombie process.
- Q.6 Command 'nice -15 p10' does
- a) decreases the priority of p10 by 15
 - b) increases the priority of p10 by 15
 - c) increases the priority of p10 to 15
 - d) none of these
- Q.7 Which of the following statements is incorrect ?
- a) 'man' command is used to find the details about the devices installed
 - b) process id (pid) of 'init' process is 1.
 - c) 'manuseradd' command is used to display the manual page about 'useradd command'
 - d) To quit from manual page , we press 'q'.

6.11 Answers of Exercises

Q. 1 (c)

Q. 2 (c)

Q. 3 (c)

Q. 6 (b)

Q. 4 (d)

Q. 7 (a)

Q. 5 (a)

References and Suggested Readings

1. Linux: The Complete Reference 6th Edition, Richard Petersen, McGraw Hill Education
2. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
3. <http://www.linuxjournal.com/article/3121>
4. www.askubuntu.com
5. <http://www.linuxnix.com/how-login-process-work-in-linux/>
6. http://www.linfo.org/etc_skel.html
7. <https://www.howtogeek.com/108890/how-to-get-help-with-a-command-from-the-linux-terminal-8-tricks-for-beginners-pros-alike/>
8. http://linuxcommand.org/lc3_Its0060.php
9. <http://www.dummies.com/computers/operating-systems/linux/how-to-get-help-on-linux/>
10. <https://bash.cyberciti.biz/>
11. <https://www.howtogeek.com>
12. <http://internal.math.arizona.edu/services/computing/linux/man>
13. <https://www.unixmen.com/managing-your-services-and-processes-in-linux/>
14. <https://www.ibm.com/developerworks/library/l-config/>
15. http://how-to.wikia.com/wiki/Guide_to_linux_configuration_files
16. <https://bash.cyberciti.biz/guide/etc/profile>
17. www.linfo.org/
18. https://en.wikiversity.org/wiki/System_administration
19. www.cyberciti.biz/faq/linux-log-files-location-and-how-do-i-view-logs-files/

20. <https://www.javacodegeeks.com/2013/12/common-linux-log-files-name-and-usage.html>
21. <http://www.omnisecu.com/gnu-linux/redhat-certified-engineer-rhce/introduction-to-linux-user-administration.php>
22. <http://www.debianadmin.com/enable-and-disable-ubuntu-root-password.html>
23. <http://www.computerhope.com/unix/info.htm>

UNIT-7

Files in Linux

Structure of the Unit

- 7.0 Objective
- 7.1 Introduction
- 7.2 LINUX Hierarchical file system
- 7.3 File Types
- 7.4 File Permissions
- 7.5 Links, Size and Space, Date and Time
- 7.6 Self Learning Exercise
- 7.7 Summary
- 7.8 Glossary
- 7.9 Answers to Self Learning Exercise
- 7.10 Exercise
- 7.11 Answers to Exercise

7.0 Objective

In this chapter we shall focus upon the following topics

- Files and its structure in system
- File types
- File Permissions
- Links of files
- File size and space with date and time

7.1 Introduction

A simple explanation of the UNIX system, also valid to Linux, is this:

"Everything is a file on a UNIX system; if something is not a file, it is a process."

This statement is true as there are special files that are more than just files (liked pipes and sockets, for example), but to keep things simple, saying that everything is a file. A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are reflected to be files, according to the system.

7.2 LINUX Hierarchical file system

A filesystem of LINUX is a set of data structure that usually be located in on part of a disk and it holds directories and files. Filesystems collect user and system data that are the basis of users' work on the system and the system's presence. Linux piles data and programs in **files**. These are structured in directories. In a simple way, a directory is just a file that contains other files (or directories).

A *hierarchical* assembly frequently takes the shape of a pyramid. One example of this type of structure is found by tracing a family's lineage: A couple has a child, who may in turn have several children, each of whom may have more children. This hierarchical structure is called a *family tree* is shown in figure 7.1.

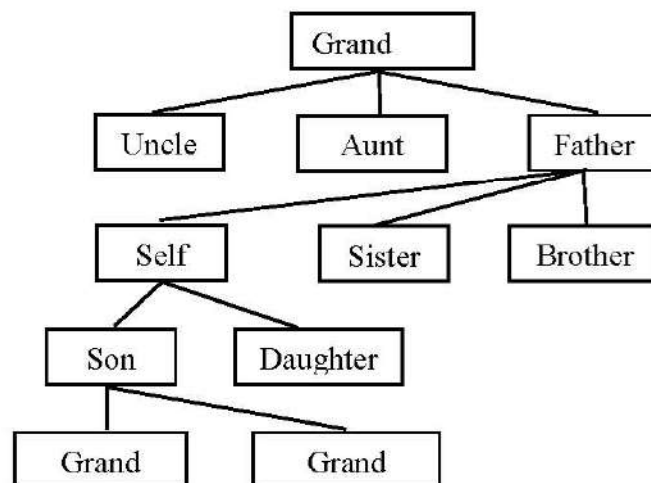


Figure 7.11

Like the family tree it look like, the Linux filesystem is called a *tree*. It consists of aset of linked files. This arrangement allows you to organize files so you can easilyfind any actual one. On a standard Linux system, each user switches with onedirectory, to which the user can add subdirectories to any desired level. By creatingnumerous levels of subdirectories, a user can enlarge the structure as needed. Typically each subdirectory is dedicated to a sole subject, such as a person,project, or event. The subject commands whether a subdirectory should be subdividedfurther.

7.3 File Types

In LINUX, a file is any source from which data can be read or any destination to which data can be written. Therefore, the keyboard, a source of input, is a file; the monitor, a destination for output, is a file; a printer, another destination for output, is a file; and a document stored on a disk, a source or destination of data, is also a file.

LINUX provides seven file types as shown in figure 7.2.

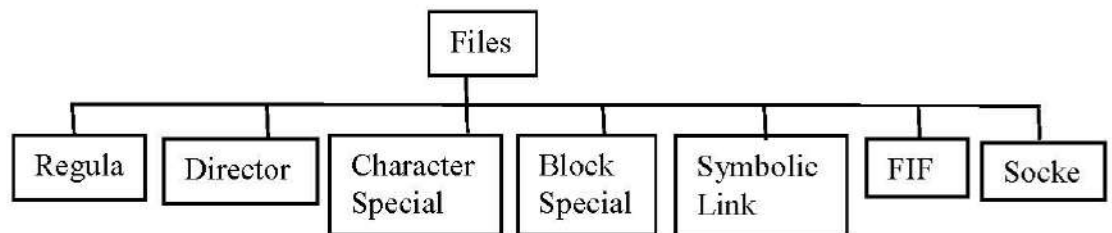


Figure 7.2

Regular Files:-Regular files cover user data that need to be accessible for future processing. Sometimes called ordinary files, regular files are the most common files originate in a system.

Directory files :- A directory is a file that encompasses the names and locations of all files stored on a physical device.

Character Special files :- A character special file characterizes a physical device, such as a terminal, that reads or writes one character at a time.

Block Special files :- A block special file embodies a physical device, such as a disk, that reads and writes data a block at a time.

Symbolic Link files :- A Symbolic link is a logical file that describes the location of another file somewhere else in the system.

FIFO files :- A first-in, first-out file is a file that is used for interposes communication.

Socket :- A socket is a distinct file that is used for network administration.

Regular Files

The most public file in Linux is the regular file. Regular files are separated by the physical format used to accumulate the data as text or binary. The physical format is controlled by the application program or utility that processes it.

Text Files:A text file is a file of characters drawn from the computer's character set. Linux computers use the ASCII character set. Because the Linux shells treat data almost universally as strings of characters, the text file is the most common Linux file.

Binary Files: A binary file is a assembly of data stored in the internal format of the computer. In general, there are two types of binary files: data files and program files. Data files contain application data. Program files contain instructions that make a program work. If you try to process a binary file with a text-processing utility, the output will look very strange because it is not in a format that can be read by people.

Directory Files

Like other operating system, Linux has a provision for organizing files by grouping them into directories. A directory performs the same function as a folder in a filing cabinet. It organize related files and subdirectories in one place.

In most systems, the directory hierarchy is designed as shown in figure 7.3. At the top of the directory structure is a directory called the *root*. Although its name is root, in the commands related to directories, it is typed as one slash (/). In turn, each directory can contain subdirectories and files.

Special directories

There are four special directories that play an important role in the directory structure. Root directory, home directory, working directory and parent directory.

Root directory: The root directory is the highest level of the hierarchy. It is the root of the whole file structure; therefore, it does not have a parent directory. In a LINUX environment, the root directory always has several levels of subdirectories. The root directory belongs to the system administrator and can

be changed by only the system administrator. The abbreviation of the root directory is / (slash).

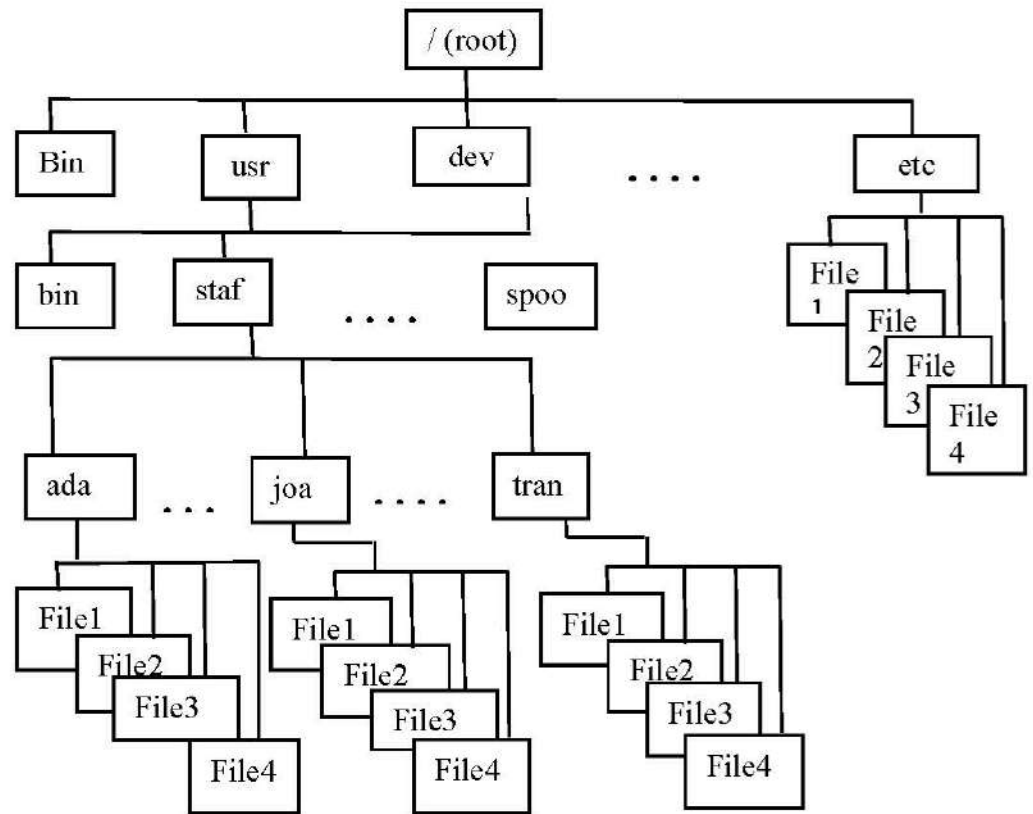


Figure 7.3

Home directory: We use home directory when we first log into the system. It contains any files we create while in it and may contain personal system files such as our profile file and command history. Each user has home directory. The name of the home directory is the user login id or the user id. The abbreviation of a user's home directory is the tilde (~). When we use the tilde, the shell uses the home directory pathname set for us by the system. When we need to refer to our home directory, we can use the tilde.

Working directory: The working, or current, directory is the one that we are in at any point in a session. When we start, working in our home directory. If we have subdirectories, we will most likely move from our home directory to one or more subdirectories as needed during a session. When we change directory, our working directory changes automatically. The abbreviation for the working directory is a dot (.).

Parent directory: The parent directory is immediately the working directory. When we are in our home directory, its parent is one of the system directories. When we move from our home directory to a subdirectory, our home directory becomes the parent directory. The abbreviation for the parent directory is two dots (..).

7.4 File Permissions

Linux supports two methods of controlling who can access a file and how they can access it: traditional Linux access permissions and Access Control Lists (ACLs). ACLs provide finer-grained control of access privileges. This section

describes traditional Linux access permissions.

Three types of users can access a file: the owner of the file (owner), a member of a group that the file is associated with and everyone else (other). A user can attempt to access an ordinary file in three ways: by trying to read from, write to, or execute it.

ls -l: Displays Permissions

When you call `ls` with the `-l` option and the name of one or more ordinary files, `ls`

displays a line of information about the file. The following example displays information for two files. The file `letter_spell` contains the text of a letter, and `check` contains a shell script, a program written in a high-level shell programming language:

```
$ ls -l letter_spell check
```

```
-rw-r--r-- 1 alex pubs 3355 May 2 10:52 letter_spell
-rwxr-xr-x 2 alex pubs 852 May 5 14:03 check
```

From left to right, the lines that `ls -l` command displays contain the following information.

- The type of file (first character).
- The file's access permissions (the next nine characters).
- The ACL flag (present if the file has an ACL).
- The number of links to the file.
- The name of the owner of the file (usually the person who created the file).

- The name of the group that the file is associated with.
- The size of the file in characters (bytes).
- The date and time the file was created or last modified.
- The name of the file.

The type of file (first column) for **letter_spell** is a hyphen (-) because it is an ordinary file (directory files have a **d** in this column).

The next three characters specify the access permissions for the *owner* of the file: **r** indicates read permission, **w** indicates write permission, and **x** indicates execute permission. A - in a column indicates that the owner does *not* have the permission that would have appeared in that position.

In a similar manner the next three characters represent permissions for the *group*, and the final three characters represent permissions for *other* (everyone else). In the preceding example, the owner of **letter_spell** can read from and write to the file, whereas the group and others can only read from the file and no one is allowed to execute it. Although execute permission can be allowed for any file, it does not make sense to assign execute permission to a file that contains a document, such as a letter. The **check** file is an executable shell script, so execute permission is appropriate for it. (The owner, group, and others have execute access permission.)

chmod: Changes Access Permissions

The **chmod** (change mode) command is used to set the permission of one or more files for all three categories of users (user, group and others). It can be run only by the user (the owner) and the superuser. The command can be used in two ways:

- In a relative manner by specifying to change the current permissions.
- In an absolute manner by specifying the final permissions.

Relative Permissions

When changing permissions in a relative manner, **chmod** only changes the permissions specified in the command line and leaves the other permissions unchanged. In this mode it uses the following syntax:

```
chmod category operation permission filename(s)
```

chmod takes as its argument an expression comprising some letters and symbols that completely describe the user category and the type of the permission being assigned or removed. The expression contains three components:

- User category (user, group, others)
- The operation to be performed (assign or remove a permission)
- The type of permission (read, write, execute)

By using suitable abbreviations for each of these components, user can frame a compact expression and then use it as an argument to chmod. The abbreviations used for these three components are shown in table 7.1

Table 7.1 Abbreviations used by chmod

Category	Operation	Permission
u---User	+---Assigns permission	r---read permission
g---Group	- ---Remove permission	w---Write permission
o---Others	=---Assign absolute permission	x---Execute
permission		
a---All (ugo)		

The owner of a file controls which users have permission to access the file and how they can access it. When you own a file, you can use the chmod (change mode) utility to change access permissions for that file. In the following example, chmod adds (+) read and write permissions (rw) for all (a) users:

```
$ chmoda+rwletter_spell
$ ls -l letter_spell
-rw-rw-rw- 1 alex pubs 3355 May 2 10:52 letter_spell
```

7.5 Links, Size and Space, Date and Time

Links

A *link* is a pointer to a file. Every time you make a file, you are putting a pointer in a directory. This pointer links a filename with a place on the disk.

When you state a filename in a command, you are ultimately pointing to the place on the disk that holds the information you want.

Sharing files can be beneficial when two or more people are working on the same project and need to share some information. You can make it easy for other users to access one of your files by producing additional links to the file.

To share a file with additional user, first give the user permission to read from and write to the file. You may also have to change the access permissions of the parent directory of the file to give the user read, write, or execute permission. Once the permissions are appropriately set, the user can create a link to the file so that each of you can access the file from your separate directory hierarchies.

A link can also be useful to a single user with a large directory hierarchy. To create a link, `ln` command is used and it has two options: force (Hard) and symbolic link.

Hard Link

A hard link to a file appears as another file. If the file appears in the same directory as the linked-to file, the links must have different filenames because two files in the same directory cannot have the same name. You can create a hard link to a file only from within the filesystem that holds the file. The default link type is hard.

The `ln` (link) utility (without the `-s` or `—symbolic` option) creates a hard link to an existing file using the following syntax:

`ln existing-file new-link`

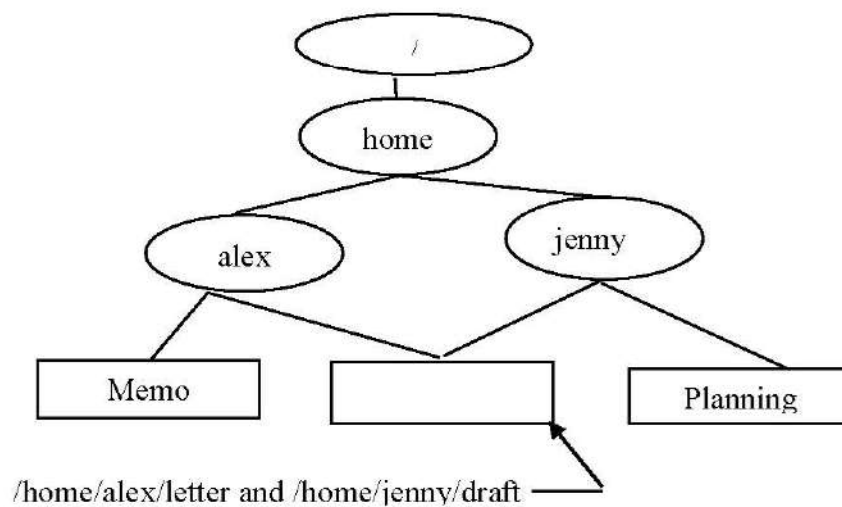


Figure 7.4

The next command makes the link shown in Figure 7.4 by creating a new link named `/home/alex/letter` to an existing file named `draft` in Jenny’s home directory:

```
$ pwd
/home/jenny
$ ln draft /home/alex/letter
```

The new link appears in the `/home/alex` directory with the filename `letter`. In practice, Alex may need to change the directory and file permissions so that Jenny will be able to access the file. Even though `/home/alex/letter` appears in Alex’s directory, Jenny is the owner of the file because she created it.

The `ln` utility creates an additional pointer to an existing file but it does *not* make another copy of the file. Because there is only one file, the file status information—such as access permissions, owner, and the time the file was last modified—is the same for all links; only the filenames differ. When Jenny modifies `/home/jenny/draft`, for example, Alex sees the changes in `/home/alex/letter`.

Symbolic Links

In addition to hard links, Linux supports symbolic links, also called soft links or `symlinks`. A hard link is a pointer to a file (the directory entry points to the inode), whereas a symbolic link is an indirect pointer to a file (the directory entry contains the pathname of the pointed-to file—a pointer to the hard link to the file).

Symbolic links were developed because of the limitations inherent in hard links. You cannot create a hard link to a directory, but you can create a symbolic link to a directory. A major advantage of a symbolic link is that it can point to a nonexistent file. This ability is useful if you need a link to a file that is periodically removed and recreated.

A hard link keeps pointing to a “removed” file, which the link keeps alive even after a new file is created. In contrast, a symbolic link always points to the newly created file and does not interfere when you delete the old file. Although they are more general than hard links, symbolic links have some disadvantages.

Whereas all hard links to a file have equal status, symbolic links do not have the same status as hard links. When a file has multiple hard links,

You use `ln` with the `—symbolic` (or `—s`) option to create a symbolic link. The following example creates a symbolic link `/tmp/s3` to the file `sum` in Alex's home directory.

When you use an `ls -l` command to look at the symbolic link, `ls` displays the name of the link and the name of the file it points to. The first character of the listing is `l` (for link).

```
$ ln --symbolic /home/alex/sum /tmp/s3
$ ls -l /home/alex/sum /tmp/s3
-rw-rw-r-- 1 alexalex 38 Jun 12 09:51 /home/alex/sum
lrwxrwxrwx 1 alexalex 14 Jun 12 09:52 /tmp/s3 -> /home/alex/sum
$ cat /tmp/s3
```

This is `sum`.

The sizes and times of the last modifications of the two files are different. Unlike a

hard link, a symbolic link to a file does not have the same status information as the

file itself.

Removes a link

When you create a file, there is one hard link to it. You can then delete the file or, using Linux terminology, remove the link with the `rm` utility. When you remove the last hard link to a file, you can no longer access the information stored there and the operating system releases the space the file occupied on the disk for subsequent use by other files.

This space is released even if symbolic links to the file remain. When there is more than one hard link to a file, you can remove a hard link and still access the file from any remaining link.

When you remove all hard links to a file, you will not be able to access the file through a symbolic link. In the following example, `cat` reports that the file `total` does not exist because it is a symbolic link to a file that has been removed:

```

$ ls -l sum
-rw-r--r-- 1 alex pubs 981 May 24 11:05 sum
$ ln -s sum total
$ rm sum
$ cat total
cat: total: No such file or directory
$ ls -l total

```

```
lrwxrwxrwx 1 alex pubs 6 May 24 11:09 total -> sum
```

When you remove a file, be sure to remove all symbolic links to it. Remove a symbolic link in the same way you remove other files:

```
$ rm total
```

Size and Space

A file has a number of attributes that are stored in the `inode` which is already discussed in previous section. We use `ls -l` command with additional options to display these attributes. We have already used the `ls` command with a number of options. It's the `-l` (long) option that reveals most. This option displays most attributes of a file---like its permission, size and ownership details. The output in Linux is often referred to as the listing. Sometimes we combine this options with other options for displaying other attributes, or ordering the list in a different sequence.

`ls` looks up the file's `inode` to fetch its attribute. Let's use `ls -l` to list seven attributes of all files in the current directory:

```

$ls -l
-rw-r--r--  1  kumar    metal 19514 May  10   13:45 chap01
-rw-r--r--  1  kumar    metal  4174 May  10   15:01 chap02
-rw-rw-rw-  1  kumar    metal   84 Feb   2   12:30 dept.lst

```

The fifth column shows the size of the file bytes, i.e., the amount of data it contains. The important thing to remember is that it is only a character count of the file and not a measure of disk space that it occupies. The space occupied by a file on disk is usually larger than this figure since files are written to disk in

blocks of 1024 bytes or more. In other words, even though the file `dept.lst` contains 84 bytes, it would occupy 1024 bytes on the system that uses a block size of 1024 bytes.

Date and Time

The **date** command displays the system date and time. If the system is local—that is, one in your own area—it is the current time. If the system is remote, such as across the country somewhere, the reply will contain the time where the system is physically located. Each date response indicates what time zone is being used.

The input for **date** is the system itself; the date is actually maintained in the computer as a part of the operating system. Most modern hardware also has a hardware date and time clock that is often updated automatically to ensure that it is accurate. The **date** command sends its response to the monitor.

If you enter the **date** command without any options, it displays the current date and time as shown in the following example:

```
$date
```

```
Fri Aug 19 13:29:23 IST 2008
```

The **date** command has only one user option and one argument. If no option is used, the time is local time. If a `-u` option is used, the time is GMT which is as follows:

```
$date -u
```

```
Fri July 23 17:34:12 GMT 2009
```

The **date** command arguments allow you to customize the format of the date. To create your own format, you use arguments. Command arguments are modified to the command. Because different arguments have different requirements, each command requires a unique set of arguments. For the **date** command, the format, is a plus sign (+) followed by text and a series of former codes all enclosed in double quote marks. Each code is preceded by a percentage sign (%) that identifies it as a code. The output display follows the command on the console. For example, the following arguments prints the date and text as shown:

```
$ date "+Today's date is: %D. the time is: %T"
```

7.6 Self Learning Exercise

- Q.1 Binary or executable files are
- a) Regular files
 - b) Device files
 - c) Special files
 - d) Directory files
- Q.2 The file permission 764 means:
- a) Everyone can read, group can execute only and the owner can read and Write.
 - b) Everyone can read and write, but owner alone can execute,
 - c) Everyone can read, group including owner can write, owner alone can Execute.
 - d) Everyone can read and write and execute.
- Q.3 When you use the ln command, which of the following occurs?
- a) A file is created that points to an existing file.
 - b) A file is created that is a copy of an existing file.
 - c) A file is moved from one location to another.
 - d) A file is renamed.

7.7 Summary

Linux has a hierarchical, or treelike, file structure that makes it possible to organize files so that you can find them quickly and easily. The file structure contains directory files and ordinary files.

Directories contain other files, including other directories; ordinary files generally contain text, programs, or images. The ancestor of all files is the root directory named /.

Linux recognizes seven file types: regular, directory, character special, block special, symbolic link, FIFO and socket. Regular files can be either text files or binary files.

The directory is a type of the file that is used to recognize other files into groups. The directory hierarchy is designed as an upside down tree with a special directory, called the root, at the highest level.

There are four special directories with abbreviated labels: root (/), home (~), working or current (.), and parent (..).

To uniquely identify a file in the directory hierarchy, we use the pathname of the file. There are two ways we can use the pathnames: absolute and relative. The absolute pathname starts from the root directory. The relative pathname starts from the working directory.

Linux defines two types of links: hard and symbolic. A hard link is a direct link. A symbolic link (or soft) link is an indirect link. Size of file specifies volume of data that it contains. Date command shows the system date and time.

7.8 Glossary

absolute pathname—any LINUX pathname that begins with a slash (/) indicating that path starts from the root.

relative pathname--- the pathname relative to the current directory. A relative pathname should not be stated with a slash (/).

7.9 Answers to Self-Learning Exercise

Q.1 (a) Q.2 (c) Q.3 (a)

7.10 Exercise

Q.1 Deleting a soft-link

- a) Deletes the destination file
- b) Deletes both the softlink and the destination file
- c) Deletes just the softlink
- d) backup of the destination is automatically created

Q.2 Which command is used to change permissions of files and directories?

- a) mv
- b) chgrp
- c) chmod
- d) set

- Q.3 The permission -rwxr-r- represented in octal expression will be
- 777
 - 666
 - 744
 - 711
- Q.4 Which command is used to assign read-write permission to the owner?
- chmoda+r file
 - chmodo+r file
 - chmodog-r file
 - chmod u=rw file
- Q.5 Given the command
- ```
$ chmod o-w file
```
- sets write permission to everyone for file
  - sets write permission to others for file
  - clears write permission to everyone for file
  - clears write permission to others for file
- Q.6 Which column gives details of size of a file in output of ls -l command?
- Second column
  - Third column
  - Fourth column
  - Fifth column
- Q.7 date command displays
- System date
  - System time
  - Both system date and time
  - None of these

## 7.11 Answers of Exercise

- |          |          |
|----------|----------|
| Q. 1 (c) | Q. 4 (d) |
| Q. 2 (c) | Q. 5 (d) |
| Q. 3 (c) | Q. 6 (d) |

Q. 7 (c)

### References and Suggested Readings

1. A Practical Guide to Ubuntu Linux by Mark G. Sobell, Prentice Hall, 2007.
2. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
3. UNIX Shell Programming by Yashwant Kanetkar, BPB Publications, 2003.
4. <http://www.uio.no/studier/emner/matnat/fys/FYS3120/v08/undervisning/smateriale/LectureNotes3120.pdf>

# UNIT-8

## Working with Files

### Structure of the Unit

- 8.0 Objective
- 8.1 Introduction
- 8.2 Creating and reading a file
- 8.3 Copying, moving and editing a file
- 8.4 Renaming, deleting and linking a file
- 8.5 Self Learning Exercise
- 8.6 Summary
- 8.7 Glossary
- 8.8 Answers to Self-Learning Exercise
- 8.9 Exercise
- 8.10 Answers to Exercise

### 8.0 Objective

In this chapter we shall focus upon the following topics

- Introduction
- Commands to create and read a file
- Commands to copy, move and edit a file
- Commands to rename, delete and link a file

### 8.1 Introduction

Linux looks at everything as a file and any Linux system has thousands of files. If you write a program, you add one more file to the system. When you compile it, you add some more. Files grow rapidly, and Linux organizes its own files in directories and expects you to do that as well. The file is a container for storing



information. As a first approximation, we can treat it simply as a sequence of characters. If you name a file `f1` and write four characters `w`, `x`, `y`, `z` into it, then `f1` will contain only the string `wxyz` and nothing else. Unlike the DOS files, a Linux file doesn't contain the eof (end-of-file) mark. A file's size is not stored in the file, nor even its name. All files attributes are kept in a separate area of the hard disk, not directly accessible to users, but only to the kernel.

Linux treats directories and devices as files as well. A directory is simply a folder where you store filenames and other directories. All physical devices like the hard disk, memory, and modem are treated as file. The shell is also a file and so is the kernel.

## 8.2 Creating and reading a file

**cat** is one of the well-known command of the Linux system. It is mainly used to display the contents of a small file on the terminal. It is also used for creating a file and overwriting a file.

**cat** is also useful for creating a file. The following example shows the procedure to create a file.

```
$ cat> book
```

A `>` symbol following the command means that the output goes to the filename following it.

```
[ctrl-d]
```

```
$ _ prompt reappear
```

When the command line is terminated with `[Enter]`, the prompt vanishes. **cat** now waits to take input from the user. Enter the three lines, each followed by `[Enter]`. Finally press `[ctrl-d]` to signify the end of input to the system. This is the eof character used by Linux systems.

**cat** command is also used to display or read the contents of a file on the terminal.

```
$ cat book
```

A `>` symbol following the command means that the output goes to the filename following it.

**cat** also accepts more than one filename as arguments.

```
$ cat file1 file2 file3
```

This is file1...

This is file2...

This is file3...

The contents of second file and third file are shown immediately after the first file without any header information.

### **cat options (-v and -n)**

There are two **cat** options that you may find useful in displaying the contents of file.

*Displaying Nonprinting Characters (-v):* **cat** is normally used for displaying text files only. **cat** display executable files as junk. If you have nonprinting ASCII characters in your input, you can use **cat** with the `-v` option to display these characters.

*Numbering Lines (-n):* The `-n` option numbers lines. C compilers indicate the line numbers where errors are detected, and this numbering facility often helps a programmer in debugging programs.

```
$ cat -n book
```

1: A > symbol following the command means that

2: the output goes to the filename following it.

## **8.3 Copying, moving and editing a file**

### **cp: Copying a File**

The copy (**cp**) command copies a file or a group of files. It creates duplicate of a file, a set of files, or a directory. If the source is a file, the new file contains an exact copy of the data in the source file. It creates an exact image of the file on the disk with different name. If the source is a directory, all of the files in the directory are copied to the destination, which must be a directory. If the destination file already exists, its contents are replaced by the source file content. The **cp** command copies both text and binary files. The syntax requires at least two filenames to be specified in the command line.

```
$ cp file11 file21
```

If the destination file (file21) doesn't exist, it will first be created before copying takes place. If not, it will simply be overwritten without any warning from the system.

If there is only one file to be copied, the destination can be either an ordinary or directory. You then have an option of choosing your destination filename. The following example shows the procedure.

```
$ cp file11 DirA/file21 file11 copied to file21 under DirA
```

```
$ cp file11 file21 file11 retains its name under DirA
```

If we need to copy a specific file from one directory to another directory, the following syntax is used. The filename is same as source filename.

```
$ cp DirA/file11 DirB
```

**cp** can also be used to copy more than one file with a single call of the command. In that case, the last filename must be a directory, to copy the files file11, file21 and file31 to the DirA directory, you have to use **cp** like this:

```
$ cp file11 file21 file31 DirA
```

The file retain their original names in DirA. If these files are already resident in DirA, they will be overwritten. For above command to work, the DirA directory must exist because **cp** won't create it.

Wildcards can be used to copy files as long as the destination is another directory. You cannot use wildcards if you are copying to and from the same directory.

```
$ cp DirA/f* DirB
```

In the above command, all files beginning with letter f is copied DirA directory to DirB directory.

### **cp Options**

The copy command has three options: preserve attributes, interactive and recursive.

**Preserve Attributes option (-p):** when the destination file exists, its permission, owner and group are used rather than the source file attributes. We can force the permissions, owner and group to be changed, however, by using the preserve (-p) option. The example shows the working of -p option.

```
$ ls -l
```

```
-rw-r--r-- 1 gitberg staff 120 May 28 17:45 file1
-rw----- 1 gitberg staff 120 May 29 19:34 file2
```

```
$ cp -p file1 file2
$ ls -l
-rw-r--r-- 1 gitberg staff 120 May 28 17:45 file1
-rw-r--r-- 1 gitberg staff 120 May 28 17:45 file2
```

**Interactive option (-i):** we can guard against a file being accidentally deleted by a copy command by using the interactive (-i) option. When the interactive option is specified, copy asks if we want to delete an existing file. If we reply y or yes, the file is replaced. If we reply n or no, the copy is cancelled. The following example shows the procedure.

```
$ cp -i file11 file21
cp: overwrite file21 (yes/no) ? yes
```

The interactive option is especially useful and highly recommended when you are copying multiple files.

**Recursive Copy (-r):** We can copy a collection of files with the recursive (-r) copy. While the wildcard copies the matching files in a directory, the recursive copy copies the whole directory and all of its subdirectories to a new directory. The following example shows the procedure.

```
$ ls
DirA
$ lsDirB
Cannot access DirB; No such file or directory.
$ cp -r DirA DirB
$ ls
DirA DirB
$ ls DirB
file11 file21
```

### **mv: Moving a File**

The move (**mv**) command is used to move either an individual file, a list of files or a directory. After a move, the old file name is gone and the new file name is found at the destination. This is the difference between a move and a copy. After a copy, the file is physically duplicated; it exists in two places. The move format appears as shown in following example:

```
$ ls -i DirA/file1
89451 DirA/file1
```



```

$ ls -i DirB
$
$ mv DirA/file1 DirB
$ ls -i DirA
$
$ ls -i DirB
89451 DirB/file1

```

The **mv** command renames (moves) files. It has two distinct functions:

1. It renames a files (or directory).
2. It moves a group of files to a different directory.

**mv** doesn't create a copy of the file; it merely renames it. No additional space is consumed on disk during renaming.

### **mv Options**

Move has only two options: interactive (-i) and force (-f).

**Interactive option (-i):** if the destination file already exists, its contents are destroyed unless we use the interactive flag (-i) to request that **move** warn us. When the interactive flag is on, move asks if we want to destroy the existing file. The following example shows the process:

```

$ls mvDir1
file11 gitberg file31
$ mv -i file1 mvDir1
Overwrite mvDir1/file1? (yes/no): n

```

**Force option (-f):** when we are not allowed to write a file, we are asked if we want to destroy the file or not. If we are sure that we want to write it, even if it already exists, we can skip the interactive message with the force (-f) option. The following example shows the process:

```

$ ls -l mvDir1
-rw-r--r-- 1 gitberg staff 120 May 28 17:45 file1
-rw----- 1 gitberg staff 120 May 29 19:34 file2
$ mv -f file1 mvDir1
$ ls -l mvDir1
-rw-r--r-- 1 gitberg staff 120 June 30 19:20 file1
-rw-r--r-- 1 gitberg staff 120 May 28 17:45 file2

```

## vi: Editing a file

To edit a file in Linux, many editors are available but vi editor is most common editor. Vi is a powerful text editor included with most Linux systems, even embedded ones. Sometimes you'll have to edit a text file on a system that doesn't include a friendlier text editor, so knowing vi is essential.

**Getting Started:** Vi is a terminal application, so you'll have to start it from a terminal window. Use the `vi /path/to/file` command to open an existing file with vi. The `vi /path/to/file` command also works if the file doesn't exist yet; Vi will create a new file and write it to the specified location when you save. Figure 8.1 shows the details.

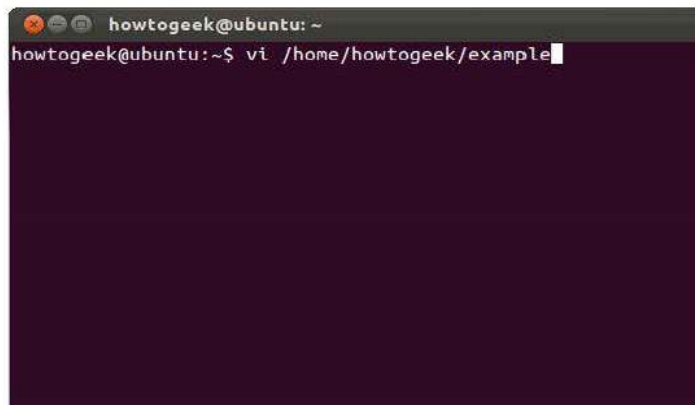
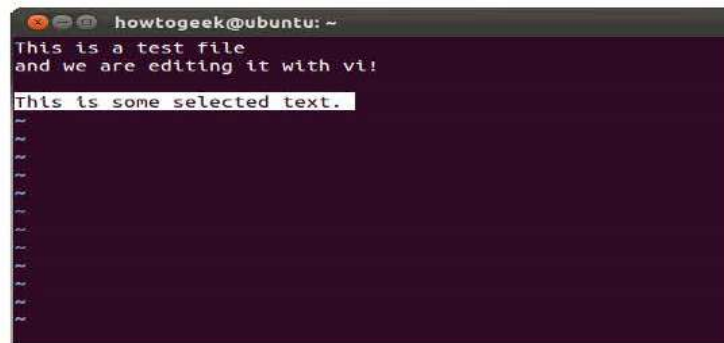


Figure 8.1

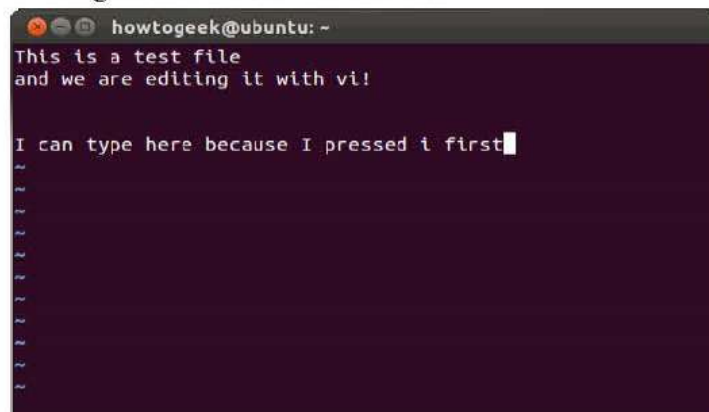
**Command Mode:** This is what you'll see when you open a file in vi. It looks like you can just start typing, but you can't. vi is a modal text editor, and it opens in command mode. Trying to type at this screen will result in unexpected behavior. While in command mode, you can move the cursor around with the arrow keys. Press the `x` key to delete the character under the cursor. There are a variety of other delete commands — for example, typing `dd` (press the `d` key twice) deletes an entire line of text.

You can select, copy, cut and paste text in command mode. Position the cursor at the left or right side of the text you want to copy and press the `v` key. Move your cursor to select text, and then press `y` to copy the selected text or `x` to cut it. Position your cursor at the desired location and press the `p` key to paste the text you copied or cut. Figure 8.2 shows the details.



**Figure 8.2**

**Insert Mode:** Aside from command mode, the other mode you need to know about is insert mode, which allows you to insert text in Vi. Entering insert mode is easy once you know it exists — just press the `i` key once after you've positioned the cursor in command mode. Start typing and Vi will insert the characters you type into the file rather than trying to interpret them as commands. Once you're done in insert mode, press the escape key to return to command mode. Figure 8.3 shows the details.



**Figure 8.3**

**Saving and Quitting:** You can save and quit vi from command mode. First, ensure you're in command mode by pressing the escape key (pressing the escape key again does nothing if you're already in command mode.) Type: `wq` and press enter to write the file to disk and quit vi. You can also split this command up—for example, type: `w` and press enter to write the file to disk without quitting or type: `q` to quit vi without saving the file. Fig 8.4 shows the details.





**rm: Deleting a file**

The remove (rm) command or utility deletes an entry from a directory by destroying its link to the file. Remember, though, that there can be multiple links to a physical file. This means that a remove does not always physically delete a file. The file is deleted only if, after the remove, there are no more links to it. The syntax of rm command is as follows:

```
$ rm file11
```

The following command deletes three files:

```
$ rm file11 file21 file31
```

A file once deleted can't be recovered. rm won't normally remove a directory but it can remove files from one. You can remove two files from a directory without having to "cd" to it.

```
$ rm DirA/file11 DirA/file21
```

You may sometimes need to delete all files in a directory as part of cleanup operation. The \*, when used by itself, represents all files, and you can use then rm like as follows:

```
$ rm *
```

```
$ __
```

To delete a file, we must have write permissions. If we try to remove a file that does not have its write flag, Linux asks for confirmation. In the following example, we will use file2, which can only be read permission. Because we don't have write permission, when we try to remove it, Linux asks for a confirmation. After completing the remove, we try to list the file to prove that it is in fact gone.

```
$ ls -l file11
```

```
-r--r--r-- 2 gitberg staff 125 June 20 15:40 file11
```

```
$ rm file11
```

```
file11: 444 mode. Remove ? (yes/no) : y
```

```
$ ls -l file11
```

cannot access file: No such file or directory

### rm Options

There are three options for the remove command: interactive, force and recursive.

**Interactive Deletion option (-i):** Like in `cp`, the `-i` (interactive) option makes the command ask the user for confirmation before removing each file:

```
$ rm -i file11 file21 file31
rm: remove file11 (yes/no) ? y
rm: remove file21 (yes/no) ? y
rm: remove file31 (yes/no) ? [Enter]
```

A `y` removes the file, any other response leaves the file undeleted.

**Recursive Deletion (-r or -R):** the recursive removal removes all files and empty directories in the path from the source directory. Files are deleted first, then the directory, so a directory can have files before the remove command. A directory is considered empty if all files are deleted. If a write-protected file is found in the path, remove asks for confirmation before completing the remove. If the response is no, the file is not deleted, but the recursive remove command continues with other files and directories. The following example shows the procedure:

```
$ ls -Rl DirAset
Total 4
drwxr-wr-w 2 gitberg staff 256 May 20 15:30 DirA
drwxr-wr-w 2 gitberg staff 256 May 20 15:30 DirB
drwxr-wr-w 2 gitberg staff 256 May 20 15:30
emptyDir
-r--r-- 1 gitberg staff 60May 20 15:30 file11
DirAset/DirA
Total 1
-rw-r--r-- 1 gitberg staff 60 May 20 15:30 file21
```

```
DirAset/DirB
```

```
Total 0
```

```
DirAset/emptyDir
```

```
Total 0
```

```
$ rm -r DirAset
```

```
DirAset/file1: 444 mode. Remove ? (yes/no): y
```

```
$ lsDirAset
```

```
fCannot access DirAset: No such file or directory.
```

**Force Removal(-f):rm** prompts for removal if a file is write-protected. The `-f` option overrides this minor protection and force removal. When you combine it with the `-r` option, it could be the most risky thing to do:

```
$ rm -rf *
```

Above command deletes everything in the current directory and below.

### In: linking a file

A link is a logical relationship between an inode and a file that relates the name of the file to its physical location. In other words, a file can have, multiple filenames, we say the file has more than one link. We can access the file by any of its links. All names provided to a single file have one thing in common; they all have the same inode number. Linux defines two types of links: hard links and symbolic links. This topic has been discussed in the last chapter.

### Hard links

In a hard link structure, the inode in the directory links the filename directly to the physical file. While this may sound like an extra level of structure, it provides the basis for multiple file linking.

A file is linked with `ln` command, which takes two filenames as arguments. The command can create both a hard and soft link.

```
$ ln stud.lst student
```

The `-i` option to `ls` shows that they have the same inode number, meaning that they are actually one and the same file.

```
$ ls -li stud.lst student
```

```
29845 -rwxr-xr-x 2 vijay steel 816 June 9 08:55 stud.lst
29845 -rwxr-xr-x 2 vijay steel 816 June 9 08:55 student
```

### Symbolic links

Hard links creates multiple names for a file. But it has two limitations:

1. You can't have two linked filenames in two file systems. In other words, you can't link a filename in the file system to another file system.
2. You can't link a directory even within the same file system.

This serious limitation was overcome when symbolic links made their entry. Unlike the hard link, a symbolic link doesn't have the file's contents, but simply provides the pathname of the file that actually has the contents. Being more flexible, a symbolic link is also known as a soft link.

The `ln` command creates symbolic links also, except that you have to use the `-s` option. This time the listing tells you a different story:

```
$ ln -s notel notel.sym
$ ln -li notel notel.sym
8795 -rw-r--r-- 1 vijay group 90 March 17 21:34 notel
8799 lrw-r--r-- 1 vijay group 4 March 17 22:30 notel.sym-
>notel
```

You can identify symbolic links by the character `l` (`el`) seen in the permission field. The pointer notation `->` note suggests that `notel.sym` contains the pathname for the filename `notel`. It's `notel` and not `notel.sym`, that actually contains the data. When you see `cat not1.sym`, you don't actually open the symbolic link, `notel.sym`, but the file the link points to. Observe that the size of the symbolic link is 4; this is the length of the pathname it contains (`notel`).

It's important you realize that this time we indeed have two "files", and they are not identical. Removing `notel.sym` won't affect us much because we can easily recreate the link. But if we remove `notel`, we would lose the file containing the data. In that case, `notel.sym` would point to a nonexistent file and become a *dangling* symbolic link.

Symbolic links can also be used with relative pathnames. Unlike hard links, they can also span multiple file system and also link directories. If you have to



link all filenames in a directory to another directory, it makes sense to simply link the directories. Like other files, a symbolic link has a separate directory entry with its own inode number. This means that **rm** can remove a symbolic link even if it points to a directory.

## 8.5 Self Learning Exercise

- Q.1 Which option of **rm** command is used to remove a directory with all its subdirectories?
- a) **-b**
  - b) **-o**
  - c) **-p**
  - d) **-r**
- Q.2 What command is used to copy files and directories?
- a) **copy**
  - b) **cp**
  - c) **rn**
  - d) **cpy**
- Q.3 Deleting a soft-link
- a) Deletes the destination file
  - b) Deletes both the softlink and the destination file
  - c) Deletes just the softlink
  - d) Backup of the destination is automatically created

## 8.6 Summary

**cat** command is used to create file and displaying or reading the contents of file. Two option (**-v** and **-n**) are used with **cat** command.

**cp** command is used to copy a file or directory from one location to another. Location can be directory. In copying a file, source location the file as destination after performing the copying operation. Three options are used with **cp** command.

mv command is same as cp command with one exception. In moving operation, source did not contain the file only destination is contains the file. Two options are used with mv command. mv command can also be used to rename a file.

Vi editor is common editor to edit a file.

A file can be removed using rm command. If a file removes using rm command, it cannot be recovered. rm command is used with three option.

Links are logical connections between file and inode. Two types of links can be created : 1) hard link 2) soft link.

## 8.7 Glossary

**Link:** It's a logical relationship between an inode and a file that relates the name of a file to its physical location.

## 8.8 Answers to Self Learning Exercise

**Ans.1:** d

**Ans.2:** b

**Ans.3:** c

## 8.9 Exercise

**Q.1** What command is used to remove files?

- A. dm
- B. rm
- C. delete
- D. erase
- E. None of the above

**Q.2** Before removing file, interactive option of rm command

- A. ask for deletion
- B. ask for confirmation
- C. ask for editing
- D. ask for moving
- E. None of the above

**Q.3** mv command perform

- A. only renaming
  - B. only moving
  - C. renaming and moving both
  - D. only editing
  - E. None of the above
- Q.4 Explain the procedure of editing of a file.
- Q.5 Force option of mv command skips
- A. moving forcedly
  - B. interactive option
  - C. only recursively
  - D. both moving forcedly and interactive option
  - E. None of the above
- Q.6 cat command is used to
- A. create and displaying a file
  - B. create a file only
  - C. displaying a file only
  - D. edit a file
  - E. None of the above
- Q.7 In cp command,
- A. source and destination both contains the same file.
  - B. only source contains the file.
  - C. only destination contains the file.
  - D. link a file
  - E. None of the above

## 8.10 Answers to Exercise

**Ans.1: B**

**Ans.2: B**

**Ans.3: C**

**Ans.5: B**

**Ans.6: A**

**Ans.7: A**

## References and Suggested Readings

1. A Practical Guide to Ubuntu Linux by Mark G. Sobell, Prentice Hall, 2007.
2. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
3. UNIX Shell Programming by YashwantKanetkar, BPB Publications, 2003.
4. UNIX and Shell Programming, A Textbook by Behrouz A. Forouzan and Richard F. Gilberg, Cengage Learning, INDIA Edition, 2003.



# UNIT-9

## File Commands in Linux

### Structure of the Unit

- 9.0 Objective
- 9.1 Introduction
- 9.2 ls
- 9.3 rm
- 9.4 mkdir
- 9.5 rmdir
- 9.6 pwd
- 9.7 more
- 9.8 less
- 9.9 grep
- 9.10 awk
- 9.11 sort
- 9.12 cat
- 9.13 head
- 9.14 tail
- 9.15 wc
- 9.16 ps
- 9.17 tee
- 9.18 top
- 9.19 tar
- 9.20 unzip
- 9.21 nice
- 9.22 kill
- 9.23 chmod

- 9.24 cd
- 9.25 umask
- 9.26 Self Learning Exercise
- 9.27 Disk related commands
- 9.28 Summary
- 9.29 Answers to Self Learning Exercise
- 9.30 Exercise
- 9.31 Answers to Exercise

## 9.0 Objective

In this chapter we shall focus upon the following topics

- Basic file commands of Linux
- Filters like grep and awk
- Process commands ps, kill
- Disk related commands du, df

## 9.1 Introduction

As we have already discussed in chapter 4, that Linux treats everything as file. There are various operations that can be performed on files, so in this chapter we will focus on commands that operate on files.

Considering the file system discussed in chapter 4, we know that every user has its own home directory and the absolute path to any file or directory starts from root directory (/). A path is a unique location to a file or a folder in a file system of an Operating System. An absolute path is defined as the location of a file or directory from the root directory (/). So, we can say absolute path is a complete path from start of actual file system i.e. from root directory (/).

E.g. /home/usr1/dir1/prg1.c

Relative path is defined as path related to the current working directory. The relative path for file whose absolute path is denoted in above example (if we are already in directory usr1) will be

```
dir1/prg1.c
```

**Important Note:-** Whenever we are specifying any Linux command, it is assumed to be executed for current working directory or file present in current working directory. If we want to execute the command for any file or directory other than current working directory, we need to specify the path. Path may be absolute or relative.

## 9.2 ls

ls is a Linux shell command that lists directory contents (by default it lists for current working directory).

### Syntax

```
$ ls [options] [directory name with path]
```

If we write only ls, it will list files and directories contained within the current working directory.

```
$ls
dir1
prg1.c
```

By default, ls command list the output in alphabetical order.

If we write only ls, it does not display hidden files (beginning with a '.'). To list hidden files as well, we need to specify command as

```
$ ls -a
```

The output of above command will consist of all files and directories including the hidden files. But it will also display two more entries .and ..

. signifies the current directory, and .. denotes the parent of current directory.

### Wildcard

A wildcard is a character that can be used as a substitute for any of a class of characters. They provide flexibility and enhance the efficiency.

Three types of wildcards are used with Linux commands. The most frequently employed and usually the most useful is the **star wildcard**, which is the same as an asterisk (\*). The star wildcard has the broadest meaning of any of the wildcards, as it can represent zero characters, all single characters or any string.

E.g. To list all files starting with 'x'

```
$ls x*
```

To list all files having .c extension

```
$ls *.c
```

The **question mark** (?) is used as a wildcard character which represents exactly one character. For more than one character in succession, we need to have as many ? in succession. Thus, two question marks in succession would represent any two characters in succession.

```
$lsa?x
```

The above command will display all files (if any) whose name is of three characters, which starts with 'a', ends with 'x'.

```
$ls a??x
```

The above command will display all files (if any) whose name is of four characters, which starts with 'a', ends with 'x'.

The question mark wildcard can also be used in combination with other wildcards. For example, the following would return a list of all files in the current directory whose file extension is of three characters.

```
$ls *.*??
```

The third wildcard is a pair of square brackets, which can represent any of the characters enclosed in the brackets. To list all file whose name start with a lower case vowel

```
$ls [aeiou]*
```

When a hyphen is used between two characters in the square brackets wildcard, it indicates a range inclusive of those two characters. To list all three character filenames in the current working directory, whose first character is in the range of x to z, the second character is in the range of l to p and third character is in the range of 3 to 8

```
$ls [x-z][l-p][3-8]
```

ls command only displays the name, but if we want to know more details, we can give a variant of ls as

```
$ls -l
```



This is called long listing.

### 9.3 rm

The `rm` (i.e., remove) command is used to delete files and directories on Linux.

The general syntax for `rm` is:

```
$rm [options] [-r directories] filenames
```

The items in square brackets are optional. When used just with the names of one or more files, `rm` deletes all those files without requiring confirmation by the user. Thus, in the following example, `rm` would immediately delete the files named `file1`, `file2` and `file3`, assuming that all three are located in the current working directory.

```
rm file1 file2 file3
```

Error messages are returned if a file does not exist or if the user does not have the appropriate permission to delete it. Write-protected files prompt the user for a confirmation (with a `y` for yes and an `n` for no) before removal. Files located in write-protected directories can never be removed, even if those files are not write-protected.

The `-f` (i.e., force) option tells `rm` to remove all specified files, whether write-protected or not, without prompting the user. It does not display an error message or return error status if a specified file does not exist.

The `-i` (i.e., interactive) option tells `rm` to prompt the user for confirmation before removing each file and directory.

`rm` does not delete directories by default. To delete directories, it is necessary to use the `-r` option. This option recursively removes directories i.e., the specified directories will first be emptied of any subdirectories (including their subdirectories and files, etc.) and files and then removed. The user is normally prompted for removal of any write-protected files in the directories unless the `-f` option is used.

If a file encountered by `rm` is a symbolic link, the link is removed, but the file or directory to which that link refers will not be affected. A user does not need write permission to delete a symbolic link, as long as the user has write permission for the directory in which that link resides.

rm command actually deletes a link, file is actually deleted from storage if it is the last link of file.

## 9.4 mkdir

The mkdir command allows users to create directories. The mkdir command can create multiple directories at once and also set permissions when creating the directory. The user running the command must have appropriate permissions on the parent directory to create a directory or permission denied error will be generated.

```
$mkdir dir1 dir2 dir3
```

Above command will create three directories in current working directory by the names dir1, dir2 and dir3.

To create multiple generations of directories using single mkdir command we can use -p option.

```
$mkdir -p a/b/c
```

The above command will create directory a, within directory a, another directory b and in directory b, directory named c will be created.

To specify permissions when creating a directory we can use -m option.

```
$mkdir -m 777 newdir
```

This will create directory named newdir with permissions rwxrwxrwx, irrespective of umask value.

## 9.5 rmdir

The rmdir command is used to remove empty directories in Linux.

The syntax for rmdir is

```
rmdir [option] directory_names
```

When used without any options, rmdir will delete any empty directories whose names are supplied as arguments (i.e., inputs) regardless of whether such directories have write permission or not.

```
$rmdir dir1 dir2
```

Just like the `-p` option in `mkdir`, this option can be used in `rmdir` to remove the parent directories of the specified directory if each successive parent directory will, in turn, become empty and if each parent directory has write permission. Thus, for example, the following would remove `dir5`, `dir4` and `dir3` if `dir5` were empty, `dir4` only contained `dir5` and `dir3` only contained `dir4` (which, in turn, contained `dir5`):

```
$rmdir -p dir3/dir4/dir5
```

## 9.6 pwd

'`pwd`' stands for 'Print Working Directory'. It prints the absolute path to current working directory.

```
$pwd
/home/usrl/dir1
```

## 9.7 more

`more` command of Linux allows us to view text files or other output in a scrollable manner. It displays the text one screen at a time, and lets us scroll backwards and forwards through the text, it also facilitates to perform search in the text.

It is usually used to display the contents of text file.

```
$more /etc/passwd
```

Now we can see the file contents on screen, and also can scroll through the file output. We can use the `[UpArrow]` and `[DownArrow]` keys to scroll through the display. Following keys can be used to move through the output:

`[Space]` - scrolls the display, one screen of data at a time.

`[Enter]` - scrolls the display one line.

`[b]` - scrolls the display backwards by one screen.

`[/]` - allows us to search the text.

`more` command can also be used with pipe

```
ls|more
```

## 9.8 less

It is similar to more, but allows backward and forward movement in the file. It does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like vi. This command is used to view files instead of opening the file.

It can be used to perform number of operations

1. To open the file(`less file_name`)

```
$less abc.txt
```

2. To Clear screen before displaying(`less -c file_path`)

```
$less -c abc.txt
```

It clears the screen and shown only the contents of the file.

3. To Starts up the file from the given number(`less +number file_path`)

```
$less +3 abc.txt
```

It displays the contents of file abc.txt from line number 3.

4. Just like more, it can also be used with pipe

```
$ls-l | less
```

5. It can also be used for navigation

Search navigation keys help us in forward and backward search.

Forward search

`/` : search for a pattern for the next occurrence

`n` : search for next match

`N` : search for previous match

Backward search

`?` : search for a pattern for the next occurrence

`n` : for next match in backward direction

`N` : for previous match in forward direction

Screen Navigation

`Ctrl + f` : forwards one window



Ctrl + d : forwards half window  
Ctrl + b : backwards one window  
Ctrl + u : backwards half window

Line Navigation is used to move forward or backward line by line

j : forward by one line  
k : backward by one line

Other Navigations

G : used to go to end of the file  
g : used to go to start of the file  
q or ZZ : to exit

Count Keys

10j : 10 lines forward  
10k : 10 lines backward

Marked Navigation

ma : mark current position with letter 'a'  
a : go to marked position

## 9.9 grep

grep stands for "globally search a regular expression print it". This command processes text line by line and prints any lines which match a specified pattern.

```
$ grep ex f1
```

The above command searches for pattern `ex` in file `f1`, and prints the line(s) containing `ex`.

It can also be used to search for pattern in multiple files

```
$ grep ex f1 f2 f3
```

The above command searches for pattern `ex` in file `f1`, `f2` and `f3` thus printing the line(s) containing `ex`.

For case insensitive search use `grep -I` and for number of lines use `-n`

Wildcards (Discussed in section 9.2) can also be used with `grep`

```
$grep c??knewfile
```

The above command searches for a pattern of four characters starting with c and ending with k in file named newfile and print the lines containing such pattern.

## 9.10 awk

The awk command is powerful method for processing or analyzing text files, in particular data files that are organized by lines (rows) and columns. Simple awk command can be run from the command line. More complex tasks should be written as awk programs (so-called awk scripts) to a file.

The syntax ofawk command is: -

```
awk 'pattern {action}' input-file > output-file
```

Take each line of the input file; if the line contains the pattern apply the action to the line and write the resulting line to the output-file.If the pattern is omitted, the action is applied to all lines.

For example: -

```
awk '{ print $5 }' table1.txt > output1.txt
```

This statement takes the element of the 5th column of each line and writes it as a line in the output file "output1.txt".

## 9.11 sort

Sort command is helpful to sort/order lines in text files. We can sort the data in text file and display the output on the screen, or redirect it to a file.Sorting is done by comparing the first character in each line of the file. If the first character matches, then second and so on.

```
$sort file1
```

The above command will sort the contents of file named file1 and display the sorted output on screen.

The following sort command checks whether text file data is sorted or not. If it is not, then it shows first occurrence with line number and disordered value.

```
$ sort -c file1
```

```
sort: test:3: disorder: 1
```

The following sort command sorts lines in test file in reverse order and displays sorted output.

```
$ sort -r test
```

## 9.12 cat

The cat (short for “concatenate”) command is one of the most frequently used command in Linux. cat command allows us to create single or multiple files, view contents of file, concatenate files and redirect output in terminal or files.

This command can be used to show contents of file.

```
$cat f1
```

It can also be used to display contents of multiple files

```
$cat f1 f2
```

With the help of this command, we can also create a new file

```
$cat >newfile
```

By the above command the cursor will be positioned in next line and awaits input from user. When user has typed desired text, the user needs to press CTRL+D (hold down Ctrl Key and type ‘d’) to exit. The text will be written in file named newfile.

With the help of cat command we can also redirect standard output of a file into a new file or an existing file with ‘>’(greater than) symbol. We must be careful because existing contents of file may get overwritten.

```
$cat f1 > f2
```

If the file f2 exists, the content of f2 will be overwritten by contents of f1. And if it does not exists, it will be created and contents of f1 will be redirected to f2.

It can also be used to append in existing file with ‘>>’(double greater than) symbol.

```
$cat f1>>newfile
```

Contents of file f1 will be appended at the end of filenewfile.

## 9.13 head

The head command reads the first few lines of any file given to it as an input and writes them to standard output. By default it prints the first 10 lines of each FILE to standard output.

`$head f1`

Display the first ten lines of f1.

`$head -15 f1`

Display the first fifteen lines of f1.

`$head f1 f2`

Display the first ten lines of both f1 and f2, with a header before each file which indicates the file name.

## **9.14 tail**

It displays the last few lines of a file. It is mostly used for viewing log file updates as these updates are appended to the log files. It can be used similar to head.

`$tail f1`

Display the last ten lines of f1.

`$tail -15 f1`

Display the last fifteen lines of f1.

## **9.15 wc**

The wc (word count) command is used to find out number of lines, words and characters in given file(s).

`wc -l` : Prints the number of lines in a file.

`wc -w` : prints the number of words in a file.

`wc -c` : Displays the count of bytes in a file.

`wc -m` : prints the count of characters from a file.

`wc -L` : prints only the length of the longest line in a file.

## **9.16 ps**



The `ps` (i.e., process status) command is used to provide information about the currently running processes, including their process identification numbers (PIDs). A process is a program under execution. Every process is assigned a unique PID by the system.

`$ps`

The output displays four items of information for at least two processes currently on the system: the shell and `ps`. A shell is a program that provides the traditional, text-only user interface in Unix-like operating systems for issuing commands and interacting with the system. `ps` itself is a process and it dies (i.e., is terminated) as soon as its output is displayed.

The four items are labeled PID, TTY, TIME and CMD. TIME is the amount of CPU (central processing unit) time in minutes and seconds that the process has been running. CMD is the name of the command that launched the process. TTY (which now stands for terminal type but originally stood for teletype) is the name of the console or terminal (i.e., combination of monitor and keyboard) that the user logged into, which can also be found by using the `tty` command.

## 9.17 tee

`tee` command is used to store and view (both at the same time) the output of any other command. Tee command writes to the `STDOUT`, and to a file at a time.

The following command displays output only on the screen (`stdout`).

`$ ls`

The following command writes the output only to the file and not to the screen.

`$ ls > fl`

The following command (with the help of `tee` command) writes the output both to the screen (`stdout`) and to the file.

`$ ls | tee fl`

## 9.18 top

`top` command is majorly used for administrative purpose. It displays processor activity and also tasks managed by kernel in real-time. It'll show processor and

memory being used and other information like running processes. This helps us to take correct action.

#top

Press 'q' to quit window.

## 9.19 tar

The “tar” stands for tape archive, which is used to deal with tape drives backup. The tar command is used to rip a collection of files and directories into highly compressed archive file. The tar is most widely used command to create compressed archive files that can be moved easily from one disk to another disk or machine to machine.

If we have a directory named dir1 in the current directory and we want to save it to a file named archive.tar.gz. , this can be done by executing following command: -

```
$tar -czvf archive.tar.gz dir1
```

-c: Create an archive.

-z: Compress the archive with gzip.

-v: Display progress in the terminal while creating the archive, also known as “verbose”

mode. The v is always optional in these commands.

-f: Allows us to specify the filename of the archive.

## 9.20 unzip

Unzip is used to extract all the files of a zip archive to our current directory.

```
$unzip data.zip
```

We can also test the .zip file ( ie make sure its ok or not) by the following command .The command prints a summary message indicating whether it is usable or broken.

```
$unzip -tq data.zip
```

## 9.21 nice

By default all processes are considered equally urgent and are allotted the same amount of CPU time. In order to enable the user to change the relative urgency of processes, Linux associates a priority parameter with each job that can be set or changed by the user. The Linux kernel then reserves CPU time for each process based on their relative priority values.

Linux process priorities based on the "nice" parameter which ranges from minus 20 to plus 19. It can take on only integer values. A value of minus 20 represents the highest priority level, whereas 19 represent the lowest priority level. The default nice value is zero.

The following command starts the process "xyz" setting the "nice" value to 12.

```
nice -12 xyz
```

Note that the dash in front of the 12 does not represent a minus sign. It has the usual function of marking a flag passed as argument to the nice command.

If we want to set the nice value to minus 12, we add another dash:

```
nice --12 xyz
```

Regular users can set lower priorities (positive nice values). In order use higher priorities (negative nice values) administrator privileges are required.

We can change the priority of a job that is already running using renice.

```
#renice 17 -p 1134
```

This changes the nice value of the job with process id 1134 to 17. In this case there is no dash for command option when specifying the nice value. The following command changes the nice value of process 1134 to -3:

```
#renice -3 -p 1134
```

## 9.22 kill

The kill command is used to terminate processes without having to log out or reboot (i.e., restart) the computer, thus it is particularly important for the stability of system. Each process is automatically assigned a unique process identification number (PID) when it is created for use by the system to reference the process. To terminate the process this PID needs to be specified.



\$kill 485

The above statement can be used to kill process with PID 485.

## 9.23 chmod

chmod is used to change the permissions of files or directories. As we have discussed in section 4.6 that there are three permissions (read, write and execute) and for three type of users (owner, group and others). The command name chmod stands for "change mode", and it is used to define the way a file can be accessed.

Permissions defines the permissions for the owner of the file (the "owner"), members of the group who owns the file (the "group"), and anyone else ("others"). Permissions can be represented with octal numbers (the digits 0 through 7).

4 stands for "read",

2 stands for "write",

1 stands for "execute", and

0 stands for "no permission."

If we want to change the permissions of a file named newfile, and we want to set its permissions such that: -

The user can read, write, and execute it; members of group can read and execute it; and others may only read it. We can perform this task by executing the following command

```
$chmod 754 newfile
```

So the permissions of newfile now will be, and it can be verified by executing ls -l command.

```
rwxr-xr--
```

## 9.24 cd

cd is used for change directory i.e. to change the current working directory.

If there is a directory named dir1 in our current working directory.

```
$cd dir1
```



The above command will make dir1 as current working directory. This can be verified by pwd command.

If we want to make any directory (not in current working directory) as current working directory, we need to specify the path (absolute or relative).

## 9.25 umask

umask stands for user file creation mask, and decides the default permissions of file or directory that is to be created. umask value tells, which of three permissions (read, write and execute) are to be denied rather than granted.

To know the current value of umask, following command is used

```
$umask
```

```
022
```

Whenever a file is created, it is assumed that the permission for this file would be 666. But our umask value is 022, so it is subtracted from 666 to get 644. So new file created will have permission 644.

For directories permission is assumed as 777. Again subtracting umask value from 777, we get 755. So a new directory created will have permissions 755.

We can change the umask value by executing command as follows

```
E.g.: - $umask 242
```

After execution of above statement, any new file that we create will have permission 424(666-242) and any directory that is created will have permission 535(777-242).

## 9.26 Self Learning Exercise

**Q.1** Which command is used to sort the lines of data in a file in reverse order?

(A) sort

(B) sh

(C) st

(D) sort -r

**Q.2** Explain various uses of cat command.

**Q.3** The Octal number to be given with chmod command to make a file readable, writable and executable to the owner, readable and executable to group and others is:

(A) 000

(B) 755

(C) 744

(D) 555

## 9.27 Disk Related Commands

System administrator also needs to manage the disk drive efficiently. It is required to regularly monitor the file system and the disk space available. Some commands which are helpful for disk are: -

### 1. **df**

It stands for disk free. It tells the free as well as the used disk space for all the file systems

installed on machine.

\$df

It tells the number of free disk blocks and free inodes for the file system. df counts

blocks in size of 512 byte, irrespective of the actual block size (which can be checked by

cmchk command).

### 2. **dfspace**

It is difficult to identify free space with inodes and blocks as reported by df. It would be

better if the disk space is reported in terms of bytes/megabytes/gigabytes instead of inode

and blocks.

\$ /etc/dfspace

### 3. **du**

It tells the disk space used by specified files and directories.

\$du

It reports the number of blocks used by the current directory and the directories contained

within the current directory.

#### **4. ulimit**

Stands for 'user limit', and denotes the maximum size of file that can be created by the user

in the file system.

`$ulimit`

The above command specifies the size (in bytes). User can not create a file larger than the

size specified by `ulimit`. An ordinary user can only reduce the `ulimit` value and can not

increase it. Root user (super user) can increase or decrease this value. If `ulimit` value is

changed, it is effective only for the current session.

### **9.28 Summary**

We have discussed various commands related to file and many other commands as well. It is advised to execute the commands on their own to fully understand the working of commands. There are numerous options available with any command and it is practically not possible to discuss each and every command with all options, so readers are advised to have an hands on experience for the commands.

### **9.29 Answers to Self Learning Exercise**

**Ans.1** D

**Ans.3** B

### **9.30 Exercise**

Q.1 How are hidden files, denoted in Linux?

Q.2 The permission 746 can be represented as

(A) `rw-rwx- -x`      (B) `rw- -w-r-x`

(C) rwxr-xr-x (D) rwxr- -rw-

- Q.3 The command pwd gives  
(A) Parent working directory (B) Password in encrypted form  
(C) Password in decrypted form (D) None
- Q.4 The command 'ulimit'  
(A) set a limit on specified resource for system users  
(B) set/show process resource limit  
(C) both (a) and (b)  
(D) none of the mentioned
- Q.5 Command used to count number of character in a file is  
(A) grep (B) wc  
(C) count (D) cut
- Q.6 How dfspace is better as compared to df command?
- Q.7 Explain ps and kill command.

### 9.31 Answers to Exercise

**Ans.1** files beginning with dot (.) **Ans.2** D

**Ans.3** D **Ans.4** C **Ans.5** B

### References and Suggested Readings

1. Unix Shell Programming by Yashwant Kanetkar; BPB Publication, First Edition 1996 REPRINTED 2009
2. Unix Shell Programming by Stephen G. Kochan, Patrick wood; Sams Publishing, Third Edition 2003
3. Unix Concepts and Applications by Sumitabha Das; Tata McGraw-Hill, Eighth Reprint 2008
4. A Practical Guide to Linux Commands, Editors, and Shell Programming by Mark G.
5. Sobell; Prentice Hall, Second Edition



# UNIT-10

## Programming with Shell Script

### Structure of the Unit

- 10.0 Objective
- 10.1 Introduction- The UNIX Shell
- 10.2 Variety of Shell and Comparison
- 10.3 Shell programming in Bash
- 10.4 Read Command
- 10.5 Shell variables
- 10.6 System Shell variables
- 10.7 Shell Keyword- Conditional, Case statements and looping statements
- 10.8 Parameter passing and arguments
- 10.9 Shell programs
- 10.10 Self Learning Exercise
- 10.11 Summary
- 10.12 Answers to Self-Learning Exercise
- 10.13 Exercise

### 10.0 Objective

After reading this chapter, you will be able to understand the following:

- Basic description of UNIX shell and types of Shells.
- Shell script and shell variables.
- Evaluating expression, uses of operators, conditional, loop statements.
- Command line arguments procedure.
- Understand the procedure to write Shell programs.

## 10.1 Introduction- the UNIX shell

The shell provides an interface between the user and the UNIX system. It gathers input from user and executes programs based on that input. When a program finishes executing, it displays program's output. A shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of shells, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

In UNIX, the shell is a program that interprets commands and acts as an intermediary between the user and the inner workings of the operating system as shown in figure 10.1. Providing a command-line interface (i.e., the shell prompt or command prompt), the shell is analogous to DOS and serves a purpose similar to graphical interfaces like Windows, Mac OS X, and the X Window System.

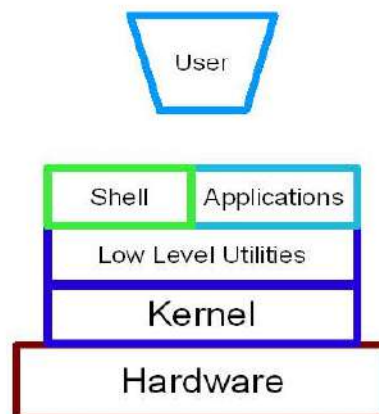


Figure 10.1

### Shell Prompt:

The prompt, \$, which is called command prompt, is issued by the shell. While the prompt is displayed, you can type a command. The shell reads user input after press Enter. It determines the command want executed by looking at the first word of input. A word is an unbroken set of characters. Spaces and tabs separate words. **Date** command, displays current date and time:

```
$date
```

```
Thu Jun 25 08:30:19 MST 2009
```

## 10.2 Variety of Shell and Comparison

There are many different shells that can be used in the UNIX environment. Each shell has its own features, strengths, and weaknesses. The C Shell is very useful and powerful in its own right, but there are other Shells that add functionality that can make working in the UNIX environment more enjoyable.

These command-line interfaces provide powerful environments for software development and system maintenance. Though shells have many commands in common, each type has unique features. Over time, individual programmers come to prefer one type of shell over another; some develop new, enhanced shells based on previous ones. UNIX also has an ecosystem of different shells.

### **Bourne Shell**

The original Bourne shell is named after its developer at Bell Labs, Steve Bourne. It was the first shell used for the UNIX operating system, and it has been largely surpassed in functionality by many of the more recent shells. However, all UNIX and many Linux versions allow users to switch to the original Bourne Shell, known simply as "sh," if they choose to forgo features such as file name completion and command histories that later shells have added.

### **C Shell**

The C shell, as its name might imply, was designed to allow users to write shell script programs using syntax very similar to that of the C programming language. It is known as "csh."

### **TC Shell**

TC shell is an expansion upon the C shell. It has all the same features, but adds the ability to use keystrokes from the Emacs word processor program to edit text on the command line. For example, users can press Esc-D to delete the rest of the highlighted word. It is also known as "tsh."

### **Korn Shell**

Korn Shell was also written by a developer at Bell Labs, David Korn. It attempts to merge the features of the C shell, TC shell and Bourne shell under one package. It also includes the ability for developers to create new shell commands as the need arises. It is known as "ksh."

## Bourne-Again Shell

The Bourne-Again shell is an updated version of the original Bourne shell that was created by the Free Software Foundation for its open source GNU project. For this reason, it is a widely used shell in the open source community.

Its syntax is similar to that used by the Bourne shell, however it incorporates some of the more advanced features found in the C, TC and Korn shells.

Among the added features that Bourne lacked are the ability to complete file names by pressing the TAB key, to remember a history of recent commands and to run multiple programs in the background at once. It is known as "bash".

**Table 10.1** A Comparison of various Shells

| <b>Feature</b>                                                                                                           | <b>sh</b> | <b>csb</b> | <b>bash</b> | <b>tcsh</b> |
|--------------------------------------------------------------------------------------------------------------------------|-----------|------------|-------------|-------------|
| Job Control: the jobs command                                                                                            | No        | Yes        | Yes         | Yes         |
| Aliasing: renaming complex commands with simple names                                                                    | No        | Yes        | Yes         | Yes         |
| Command History: re-execute frequently used commands quickly                                                             | No        | Yes        | Yes         | Yes         |
| Command line editing: correct a miss-spelled command name in a complicated command by using the arrow keys and backspace | No        | No         | Yes         | Yes         |
| Filename Completion: complete long filenames with a single keystroke                                                     | No        | Yes        | Yes         | Yes         |
| List Variables: the shell has a built-in list data type, useful when scripting                                           | No        | Yes        | No          | Yes         |
| Fully programmable completion: complete command names, hostnames, usernames, etc. with a single keystroke                | No        | No         | No          | Yes         |
| Can follow symbolic links invisibly                                                                                      | No        | No         | Yes         | No          |
| Custom Prompt (easily): for example, change the                                                                          | No        | No         | Yes         | Yes         |



|                                                                              |     |    |     |     |
|------------------------------------------------------------------------------|-----|----|-----|-----|
| prompt to display the current working directory                              |     |    |     |     |
| Can cope with large argument lists                                           | Yes | No | Yes | Yes |
| Freely available: download the shell and possibly the source code, for free! | No  | No | Yes | Yes |

### 10.3 Shell programming in Bash

As an interactive shell, bash is a terse language for initiating and directing computations. As a scripting language, bash is a domain-specific language for manipulating and composing processes and files. Bash is baroque. Bash is brittle. Bash is indispensable. Bash is an interactive shell:

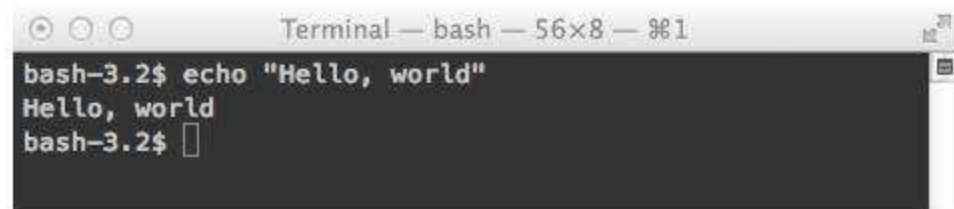


Figure 10.2

User types in command prompt. Bash executes them. UNIX users spend a lot of time in manipulating files at the shell. As a shell, it is directly available via the terminal in both Mac OS X (Applications > Utilities) and Linux/Unix. At the same time, bash is also a scripting language:

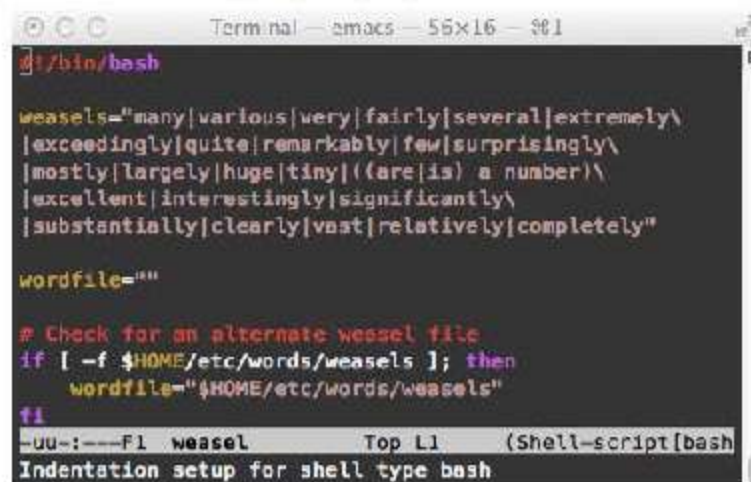


Figure 10.3

#### Bash as a scripting language

To create a bash script, you place `#!/bin/bash` at the top of the file. Then, change the permissions on the file to make it executable:

```
$ chmod u+x scriptname
```

To execute the script from the current directory, you can run `./scriptname` and pass any parameters you wish. When the shell executes a script, it finds the `#!/path/to/interpreter`.

It then runs the interpreter (in this case, `/bin/bash`) on the file itself. The `#!` convention is why so many scripting languages use `#` for comments.

### **Comments**

Comments in bash begin with `#` and run to the end of the line:

```
echo Hello, World. # prints out "Hello, World."
```

### **Variables/Arrays**

Variables in bash have a dual nature as both arrays and variables.

To set a variable, use `=`:

```
foo=3 # sets foo to 3
```

To reference the value of a variable, use a dollar sign, `$`:

```
echo $foo ; # prints the value of foo to stdout
```

There is no need to declare a variable as an array: every variable is an array.

You can start using any variable as an array:

```
foo[0]="first" # sets the first element to "first"
```

```
foo[1]="second" # sets the second element to "second"
```

You can also use parentheses to create an array:

```
foo=("a a a" "b b b" "c c c")
```

```
echo ${foo[2]} # prints "c c c"
```

```
echo $foo # prints "a a a"
```

### **String/array manipulation**

Bash has operators that operate on both arrays and strings.

For instance, the prefix operator `#` counts the number of characters in a string or the number of elements in an array.

```
ARRAY=(one two three)
```

```
echo ${#ARRAY} # prints 3 -- the length of the array?
```

### Strings and quoting

Strings in bash are sequences of characters.

To create a literal string, use single quotes; to create an interpolated string, use double quotes:

```
world=Earth
```

```
foo='Hello, $world!'
```

```
bar="Hello, $world!"
```

```
echo $foo # prints Hello, $world!
```

```
echo $bar # prints Hello, Earth!
```

In interpolated strings, variables are converted to their values.

### Expressions and arithmetic

It is possible to write arithmetic expressions in bash, but with some caution.

The command `expr` prints the result of arithmetic expressions, but one must take caution:

```
expr 3 + 12 # prints 15
```

```
expr 3 * 12 # (probably) crashes: * expands to all files
```

```
expr 3 * 12 # prints 36
```

### Control structures

Like most languages, bash supports control structures for conditionals, iteration and subroutines.

#### Conditionals

If-then-else-style conditionals exist in bash, as in other languages.

However, in bash the condition is a command, and an exit status of success (0) is "true," while an exit status of fail (non-zero) is "false.":

```
this will print:
```

```
if true
then
 echo printed
fi
```

## Iteration

The *while command; do commands; done* form executes *commands* until the test *command* completes with non-zero exit status:

# automatically restart the httpd in case it crashes:

```
while true
```

```
do
 httpd
done
```

## Subroutines

Bash subroutines are somewhat like separate scripts.

There are two syntaxes for defining a subroutine, first is -

```
function name {
commands
}
```

And other is:

```
name () {
commands
}
```

## Sample Example

Putting this all together allows us to write programs in bash.

Here is a subroutine for computing factorial:

```
function fact {
 result=1
 n=$1
```



```
while ["$n" -ge 1]
do
 result=$(expr $n * $result)
 n=$(expr $n - 1)
done
echo $result
}
```

## 10.4 Read Command

The read command is used to get a line of input into a variable. Each argument must be a variable name without the leading "\$". The built in command reads a line of input and separates the line into individual words using the "IFS" inter field separator. By default the "IFS" is set to a space. Each word in the line is stored in a variable from left to right.

The first word is stored in the first variable, the second word to the second variable and so on. If there are fewer variables than words, then all remaining words are then assigned to the last variable. If you have more variables than words defined, then any excess variables are set to null. If no variable names are supplied to the read line, then the read uses the default variable REPLY.

The read example is as follows:

```
#!/bin/bash
#
read first middle last
echo "Hello $first $middle $last"
```

### Output from above read example

```
john@john-desktop:~/scripts$./read1.sh
land of linux
Hello land of linux
```

In the above "read" example, we passed the values "land", "of" and "linux" into the variables "first", "middle" and "last". These are then displayed via the echo command.

### Options available to the read command

| Option            | Description                                                                                    |
|-------------------|------------------------------------------------------------------------------------------------|
| <b>-a ANAME</b>   | Words are assigned sequentially to the array variable ANAME                                    |
| <b>-d DELIM</b>   | The first character of DELIM is used to terminate the input line                               |
| <b>-e</b>         | readline is used to get line                                                                   |
| <b>-n NCHARS</b>  | read returns after reading NCHARS                                                              |
| <b>-p PROMPT</b>  | Display PROMPT without a trailing newline. Prompt is only displayed if coming from a terminal. |
| <b>-r</b>         | Backslash does not act as an escape character                                                  |
| <b>-s</b>         | Silent Mode. Characters are not echoed coming from a Terminal                                  |
| <b>-t TIMEOUT</b> | read will timeout after TIMEOUT seconds. Only from a Terminal                                  |
| <b>-u FD</b>      | read input from File Descriptor FD                                                             |

## 10.5 Shell Variables

A Shell variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data. A shell variable is a pointer to the actual data. The shell enables you to create, assign, and delete variables.

### Variable Names

The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character ( \_ ).

By convention, UNIX shell variables will have their names in UPPERCASE.

The following examples are valid variable names:

```
_ALI, TOKEN_A, VAR_1, VAR_2
```

### Defining Variables

Variables are defined as follows –

```
variable_name=variable_value
```

For example –NAME="VijayChouhan"

The above example defines the variable NAME and assigns the value "Vijay Chouhan" 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, add prefix its name with the dollar sign (\$) –For example, the following script will access the value of defined variable NAME and print it on STDOUT –

```
#!/bin/sh
NAME="Vijay Chouhan"
echo $NAME
```

The above script will produce the following value –

```
Vijay Chouhan
```

### Unsetting Variables

Unsetting or deleting a variable sends the shell to remove the variable from the list of variables that it tracks. Once user unset a variable, user cannot access the stored value in the variable.

Following is the syntax to unset a defined variable using the **unset** command –

```
unset variable_name
```

The above command unsets the value of a defined variable. Here is a simple example that demonstrates how the command works –

```
#!/bin/sh
NAME="Vijay Chouhan"
unset NAME
echo $NAME
```

### Variable Types

When a shell is running, three main types of variables are present –

- **Local Variables** – A local variable is a variable that is existing 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 accessible 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 needed by the programs that it runs.
- **Shell Variables** – A shell variable is a distinct 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.

## 10.6 System Shell Variables

Every UNIX process runs in a specific *environment*. An environment consists of a table of *environment variables*, each with an assigned value. When you log in certain *login files* are executed. They initialize the table holding the environment variables for the process. When this file passes the process to the shell, the table becomes accessible to the shell. When a (parent) process starts up a child process, the child process is given a copy of the parent process' table. Environment variable names are generally given in upper case by convention.

The shell also maintains a set of internal variables known as *shell variables*. These variables cause the shell to work in a particular way. Shell variables are local to the shell in which they are defined; they are not available to the parent or child shells. By convention, shell variable names are generally given in lower case in the C shell family and upper case in the Bourne shell family.



## Shell Variables

### *\$ dollar sign*

Another important character interpreted by the shell is the dollar sign\$. The shell will look for an environment variable named like the string following the dollar sign and replace it with the value of the variable (or with nothing if the variable does not exist). These are some examples using \$HOSTNAME, \$USER, \$UID, \$SHELL, and \$HOME.

```
[peter@RHELv4u3 ~]$ echo This is the $SHELL shell
```

```
This is the /bin/bash shell
```

```
[peter@RHELv4u3 ~]$ echo This is $SHELL on computer $HOSTNAME
```

```
This is /bin/bash on computer RHELv4u3.localdomain
```

```
[peter@RHELv4u3 ~]$ echo The userid of $USER is $UID
```

```
The userid of paul is 500
```

```
[peter@RHELv4u3 ~]$ echo My homedir is $HOME
```

```
My homedir is /home/paul
```

### *Case sensitive*

This example shows that shell variables are case sensitive!

```
[peter@RHELv4u3 ~]$ echo Hello $USER
```

```
Hello paul
```

```
[peter@RHELv4u3 ~]$ echo Hello $user
```

```
Hello
```

### *Creating variables*

This example creates the variable \$MyVar and sets its value. It then uses echo to verify the value.

```
[peter@RHELv4u3 gen]$ MyVar=555
```

```
[peter@RHELv4u3 gen]$ echo $MyVar
```

```
555
```

```
[peter@RHELv4u3 gen]$
```

### ***Quotes***

Notice that double quotes still allow the parsing of variables, whereas single quotes prevent this.

```
[peter@RHELv4u3 ~]$ MyVar=555
[peter@RHELv4u3 ~]$ echo $MyVar
555
[peter@RHELv4u3 ~]$ echo "$MyVar"
555
[peter@RHELv4u3 ~]$ echo '$MyVar'
$MyVar
```

### ***Set***

You can use the `set` command to display a list of environment variables. On Ubuntu and Debian systems, the `set` command will also list shell functions after the shell variables. Use `set | more` to see the variables then.

### ***Unset***

Use the `unset` command to remove a variable from your shell environment.

```
[peter@RHEL4b ~]$ MyVar=8472
[peter@RHEL4b ~]$ echo $MyVar
8472
[peter@RHEL4b ~]$ unset MyVar
[peter@RHEL4b ~]$ echo $MyVar
[peter@RHEL4b ~]$
```

### ***\$PATH***

The `$PATH` variable is where the shell is looking for commands to execute. This variable contains a list of directories, separated by colons.

```
[peter@RHEL4b ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
```

The shell will not look in the current directory for commands to execute. If you want the shell to look in the current directory, then add a `.` (dot) at the end of your `$PATH`.

```
[peter@RHEL4b ~]$ PATH=$PATH:.
[peter@RHEL4b ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:.
[peter@RHEL4b ~]$
```

### ***\$env***

The `env` command without options will display a list of exported variables. The difference with `set` with options is that `set` lists all variables, including those not exported to child shells. The `env -i` command clears the environment for the subshell.

Notice that in this screenshot that `bash` will set the `$SHELL` variable on startup.

```
[peter@RHEL4b ~]$ bash -c 'echo $SHELL $HOME $USER'
/bin/bash /home/paul paul
[peter@RHEL4b ~]$ env -i bash -c 'echo $SHELL $HOME $USER'
/bin/bash
[peter@RHEL4b ~]$
```

### ***\$export***

You can export shell variables to other shells with the `export` command. This will export the variable to child shells.

```
[peter@RHEL4b ~]$ var3=three
[peter@RHEL4b ~]$ var4=four
[peter@RHEL4b ~]$ export var4
[peter@RHEL4b ~]$ echo $var3 $var4
three four
[peter@RHEL4b ~]$ bash
[peter@RHEL4b ~]$ echo $var3 $var4
four
```

## 10.7 Shell Keywords-Conditional, Case statements and looping statements

UNIX conditional statements perform different computations or actions depending on whether a programmer-specified boolean condition evaluates to true or false. These statements are used to execute different parts of your shell program depending on whether certain conditions are true. The ability to branch makes shell scripts powerful.

### *Conditional Statements*

In UNIX, we have the following conditional statements:

1. `if..then..fi` statement (Simple If)
2. `if..then..else..fi` statement (If-Else)
3. `if..elif..else..fi` statement (Else If ladder)
4. `if..then..else..if..then..fi..fi..`(Nested if)

### *If..then..fi statement*

```
if [conditional expression]
then
 statement1
 statement2
fi
```

This if statement is also called as simple if statement. If the given conditional expression is true, it enters and executes the statements enclosed between the keywords “then” and “fi”. If the given expression returns zero, then consequent statement list is executed.

### *If..then..else..fi statement*

```
If [conditional expression]
then
 statement1
 statement2
else
```



```
 statement3
 statement4
fi
```

If the conditional expression is true, it executes the statement1 and 2. If the conditional expression returns zero, it jumps to else part, and executes the statement3 and 4. After the execution of if/else part, execution resume with the consequent statements.

*If..elif..else..fi*

```
If [conditional expression1]
then
 statement1
 statement2
elif [conditional expression2]
then
 statement3
 statement4
else
 statement5
fi
```

You can use this if .. elif.. if , if you want to select one of many blocks of code to execute. It checks expression 1, if it is true executes statement 1,2. If expression1 is false, it checks expression2, and if all the expression is false, then it enters into else block and executes the statements in the else block.

*If..then..else..if..then..fi..fi..*

```
If [conditional expression1]
then
 statement1
 statement2
else
```

```

 if [conditional expression2]
 then
 statement3
 fi
 fi

```

If statement and else statement could be nested in bash. The keyword “fi” indicates the end of the inner if statement and all if statement should end with the keyword “fi”.

### ***Case Statements***

User can use multiple **if...elif** statements to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable. Shell supports **case...esac** statement which handles exactly this situation, and it does so more efficiently than repeated if...elif statements.

### **Syntax**

The basic syntax of the **case...esac** statement is to give an expression to evaluate and to execute several different statements based on the value of the expression.

The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

*case word in*

*pattern1)*

*Statement(s) to be executed if pattern1 matches*

*::*

*pattern2)*

*Statement(s) to be executed if pattern2 matches*

*::*

*pattern3)*

*Statement(s) to be executed if pattern3 matches*

*::*

*esac*

Here the string word is compared against every pattern until a match is found. The statement(s) following the matching pattern executes. If no matches are found, the case statement exits without performing any action.

There is no maximum number of patterns, but the minimum is one.

When statement(s) part executes, the command ;; indicates that the program flow should jump to the end of the entire case statement. This is similar to break in the C programming language.

### ***Looping Statements***

A loop is a powerful programming tool that enables you to execute a set of commands repeatedly. In this chapter, we will examine the following types of loops available to shell programmers –

- The while loop
- The for loop
- The until loop
- The select loop

#### ***The while Loop***

The while loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.

Syntax

*while command*

*do*

*Statement(s) to be executed if command is true*

*done*

Here the Shell command is evaluated. If the resulting value is true, given statement(s) are executed. If command is false then no statement will be executed and the program will jump to the next line after the done statement.

#### ***The for Loop***

The for loop operates on list of items. It repeats a set of commands for every item in a list.

Syntax

*for var in word1 word2 ... word N*

*do Statement(s) to be executed for every word.*

*done*

Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the value of the variable var is set to the next word in the list of words, word1 to wordN.

*The until Loop*

The while loop is perfect for a situation where you need to execute a set of commands while some condition is true. Sometimes you need to execute a set of commands until a condition is true.

Syntax

*until command*

*do*

*Statement(s) to be executed until command is true*

*done*

Here the Shell command is evaluated. If the resulting value is false, given statement(s) are executed. If the command is true then no statement will be executed and the program jumps to the next line after the done statement.

*The select Loop*

The select loop provides an easy way to create a numbered menu from which users can select options. It is useful when you need to ask the user to choose one or more items from a list of choices.

Syntax

*select var in word1 word2 ... wordN*

*do*

*Statement(s) to be executed for every word.*

*done*

Here var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the



value of the variable `var` is set to the next word in the list of words, `word1` to `wordN`.

## 10.8 Parameter passing and arguments

Functions enable you to break down the overall functionality of a script into smaller, logical subsections, which can then be called upon to perform their individual tasks when needed.

Using functions to perform repetitive tasks is an excellent way to create **code reuse**. This is an important part of modern object-oriented programming principles. Shell functions are similar to subroutines, procedures, and functions in other programming languages.

### Creating Functions

To declare a function, simply use the following syntax –

```
function_name () {
 list of commands
}
```

The name of your function is **function\_name**, and that's what you will use to call it from elsewhere in your scripts. The function name must be followed by parentheses, followed by a list of commands enclosed within braces.

### Passing Parameters to a Function

You can define a function that will accept parameters while calling the function. These parameters would be represented by **\$1**, **\$2** and so on.

Following is an example where we pass two parameters *Zara* and *Ali* and then we capture and print these parameters in the function.

```
#!/bin/sh

Define your function here
Hello () {
 echo "Hello World $1 $2"
}

Invoke your function
```

```
Hello Hasan Ali
```

Upon execution, you will receive the following result –

```
./test.sh
```

```
Hello World Hasan Ali
```

### Returning Values from Functions

If you execute an exit command from inside a function, its effect is not only to terminate execution of the function but also of the shell program that called the function.

If you instead want to just terminate execution of the function, then there is way to come out of a defined function.

Based on the situation you can return any value from your function using the **return** command whose syntax is as follows –

```
return code
```

Here **code** can be anything you choose here, but obviously you should choose something that is meaningful or useful in the context of your script as a whole.

### Example

Following function returns a value 1 –

```
#!/bin/sh
Define your function here
Hello () {
 echo "Hello World $1 $2"
 return 10
}
Invoke your function
Hello Hasan Ali
Capture value returned by last command
ret=$?
echo "Return value is $ret"
```

Upon execution, you will receive the following result –

```
./test.sh
```

```
Hello World Hasan Ali
```

```
Return value is 10
```

## 10.9 Shell Programs

```
#-----
printing of string constant
#-----
#!/bin/sh
echo 'hello'
echo "hello"
echo hello
#-----
reading of text line from keyboard
#-----
#!/bin/sh
read x
echo $x
#-----
if ... else sample programs
#-----
#!/bin/sh
echo 'number=?'
read x
if [$x -eq 5]
then
 echo "five"
```

```

else
 echo "not 5"
fi
#-----
while loop - print first 10 integers from 0
#-----
#!/bin/bash
x=0
while [$x -lt 10]
do
 echo $x
 let x=$x+1
done
#-----
strings concatenation
#-----
#!/bin/sh
echo "Input string=?"
read str1
echo "Input second string=?"
read str2
s3=$str1$str2 # it works!
echo $s3
s4=${str1}${str2} # it works too!
echo $s4

```



## 10.10 Self Learning Exercise

- Q.1 Shell is
- a) interface
  - b) Command
  - c) function
  - d) file
- Q. 2 Which is not shell in following options:
- a) Bourne Shell
  - b) TC shell
  - c) J shell
  - d) Korn shell
- Q.3 The read command is used to
- a) get a line of input into a variable.
  - b) get a line of output into a variable.
  - c) get a line of output and output into a variable.
  - d) get a line of access into a variable.

## 10.11 Summary

Shell is the software installed on your system that works as command-line interpreter i.e. it takes the commands you type, interpret them and gives them to operating system (kernel) through system calls to execute. There are many different shells available in Linux. Types of shell are bash, C, TC, Korn shell etc. Bash is most common used shell in programming.

The read command is used to become a line of input into a variable. A variable is nothing more than a pointer to the actual data. The shell enables you to create, assign, and delete variables. Types of variables are local, environment and shell. Shell variables are used to know the status of shell. Shell keywords are reserve words and used by shell only. Keywords are like if...else, case...esac, while etc used for conditional, case and looping operations. Functions are part of each programming languages to repetitive execution of statements.

## 10.12 Answers to Self-Learning Exercise

- Q.1 (a)
- Q.2 (c)
- Q.3 (a)

## 10.13 Exercise

- Q.1 Define variable. What are the different types of variables used in a shell Script?
- Q.2 Write a shell script to compare two numbers and find greater number.
- Q.3 What is function? Write procedure to create a function in shell script with suitable example.
- Q.4 Write a shell script to find prime number among first 100 natural numbers.
- Q.5 Write a shell script to factorial of number which is inputted from user.

## References and Suggested Readings

1. Yashwant Kantekar, UNIX Shell Programming, BPB Publication, Third Edition.
2. Sumit Abhadhas, UNIX Concepts and applications, McGraw Hill Publications, Fourth Edition.
3. Mark. G. Sobell, A Practical Guide to Ubuntu LINUX, Prentice Hall, Fourth Edition.
4. Behrouz A. Forouzan and Richard F. Gilberg, UNIX and Shell Programming: A Textbook, Cengage Learning India, First Edition.

# UNIT-11

## System Administration

### Structure of the Unit

- 11.0 Objective
- 11.1 Introduction
- 11.2 Common administrative tasks
- 11.3 Identifying administrative files – configuration and log files
- 11.4 Managing user accounts-adding & deleting users
- 11.5 Changing permissions and ownerships
- 11.6 Creating and managing groups, modifying group attributes
- 11.7 Self Learning Exercise
- 11.8 Summary
- 11.9 Glossary
- 11.10 Answers to Self Learning Exercise
- 11.11 Exercise
- 11.12 Answers to Exercise

### 11.0 Objective

In this chapter we shall focus upon the following topics

- Common administrative tasks
- Identifying administrative files – configuration and log files.
- Managing user accounts-adding & deleting users
- changing permissions and ownerships
- Creating and managing groups, modifying group attributes

## 11.1 Introduction

System administration can be termed as managing system resources to ensure the smooth running of the system. In this unit, we shall study how to perform administrative tasks in Linux like managing packages/software, user management, group management, changing ownerships and permissions etc. We shall also see important configuration and log files that are essential for better system management.

## 11.2 Common administrative tasks

Common administrative tasks in Linux are installing, updating and removing packages, User management, group management, data backups, disk usage management etc. Let us see all these tasks in short one by one.

**Installing, updating and removing packages-** Most of Linux distributions provide the facility for installing, updating and removing packages in graphical user mode and using commands as well. **Ubuntu software center** and **Synaptic package manager** can be used to perform the tasks of installing, updating and removing packages in user-friendly graphical user mode. Using terminal we can use commands to execute these tasks. To install a package for a popular media player, VLC, we can use following command in Ubuntu-

```
sudo apt-get install vlc
```

We can use following command to remove (uninstall) a package for VLC. It may be noted that a software is organized as a package of files in Linux.

```
sudo apt-get remove vlc
```

To update the packages installed on the system to the recent version, we can use the following command-

```
sudo apt-get update
```

We have already seen user management and group management in the unit “Getting started with Linux”. We shall see some more details about user and group management in the later part of this unit.

Most of Linux distributions provide the facility for backing up the data and managing disk. Using the ‘**Backups**’ utility available in Ubuntu one can restore



the system and back up the important data. Using the utility ‘**Disk usage analyzer**’ one can analyze the used space and free space on the disk for each disk partition. To format a partition and make other changes we can use the utility ‘**Disks**’. ‘**Startup Disk creator**’ may help us creating a bootable startup Disks like bootable USB drives, DVDs that can be used to install Linux on a system. These utilities can be accessed from the start menu of Ubuntu that is also known as ‘**Dash**’.

### 11.3 Identifying administrative files - configuration and log

Now we shall study about administrative files that are mainly configuration and log files. Most of the configuration files are stored in `/etc` directory in Linux. Let’s first discuss important configuration files. The following table briefs up the main function of these configuration files-

| Configuration file/directory | Function                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>/etc/group</i>            | Contains the valid group names and the users included in the specified groups. A single user can be present in more than one group if he performs multiple tasks. For example, is a "user" is the administrator as well as a member of the project group "project 1", then his entry in the group file will look like: <code>user: * : group-id : project1</code> |
| <i>/etc/passwd</i>           | Stores the user database with fields giving the username, home directory, User ID, Group ID, path of shell and other Information about each user. We may use the utilities <code>useradd</code> , <code>usermod</code> and <code>userdel</code> to edit                                                                                                           |



|                         |                                                                                                                                                                                                                                                                                                   |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | this file.                                                                                                                                                                                                                                                                                        |
| <i>/etc/shadow</i>      | Stores actual password in encrypted format for user's account with additional properties related to user password i.e. it stores secure user account information. All fields are separated by a colon (:) symbol. It contains one entry per line for each user listed in <i>/etc/passwd</i> file. |
| <i>/etc/resolv.conf</i> | To configure the system's Domain Name System (DNS) resolver. The file is a plain-text file usually created by the network administrator or by applications that manage the configuration tasks of the system.                                                                                     |
| <i>/etc/inittab</i>     | Is a configuration file for init(first process started during booting of the computer system. It is a daemon process that continues running until the system is shut down), controls startup run levels, determines scripts to start with.                                                        |
| <i>/etc/networks</i>    | Lists names and addresses of networks accessible from the network to which the machine is connected. Used by route command. Allows use of name for network.                                                                                                                                       |

|                     |                                                                                                                                                                                                                                                                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>/etc/fstab</i>   | Contains information about major file systems on the system. It takes its name from file systems table. It is created automatically when the operating system is installed, and it is modified automatically when changes are made to the file systems. It can also be modified by using specialized programs or manually by using a text editor. |
| <i>/etc/sudoers</i> | Has list of users with special privileges along with the commands they can execute. It also contains the rules that users must follow when using the sudo command.                                                                                                                                                                                |
| <i>/etc/profile</i> | Contains Linux system wide environment and startup programs. It is used by all users with bash, ksh, sh shell. Usually used to set PATH variable, user limits, and other settings for user. It only runs for login shell. If we want to make large changes or application specific changes we may use <i>/etc/profile.d</i> directory.            |
| <i>/etc/skel</i>    | Allows a system administrator to create a default home directory for all new users on a computer or                                                                                                                                                                                                                                               |

|                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                            | <p>network and thus to make certain that all users begin with the same settings or environment. Several user configuration files are placed in <code>/etc/skel</code> by default when the operating system is installed. Typically they might include <code>.bash_profile</code>, <code>.bashrc</code>, <code>.bash_logout</code> etc. The name of the directory <code>skel</code> is derived from the word skeleton, because the files it contains form the basic structure for users' home directories.</p> |
| <p><i>/etc/crontab and the /etc/cron.* directories</i></p> | <p>Configuration of tasks that need to be executed periodically - backups, updates of the system databases, cleaning of the system, rotating logs etc.</p>                                                                                                                                                                                                                                                                                                                                                    |
| <p><i>/etc/sndconfig or /etc/sound</i></p>                 | <p>Configuration of the sound card and sound events.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <p><i>/etc/ssh</i></p>                                     | <p>Directory containing the config files for secure shell client and server.</p>                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <p><i>/etc/sysconfig</i></p>                               | <p>Directory containing the system configuration files: mouse, keyboard, network, desktop, system clock, power management etc. (specific to RedHat)</p>                                                                                                                                                                                                                                                                                                                                                       |

|                   |                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------|
| <i>/etc/magic</i> | <b>Configuration file for file types. Contains the descriptions of various file formats for the file command.</b> |
| <i>/etc/mtab</i>  | <b>A list of currently mounted file systems. Setup by boot scripts and updated by the mount command.</b>          |

After learning about important configuration files, we shall see log files now. Almost all log files are located under `/var/log` directory and its sub-directories on Linux. One needs to have administrative privileges to view or access log files. When our system is running smoothly, we should take some time to learn and understand the content of various log files, which will help us when there is a crisis and we have to look through the log files to identify the issue.

The following are few different log files that are located under `/var/log/` directory. Some of these log files are distribution specific. For example, we can see `dpkg.log` on Debian based distribution systems (for example, on Ubuntu).

- a) **`/var/log/messages`**– Contains global system messages, including the messages that are logged during system startup. There are several things that are logged in `/var/log/messages` including mail, cron, daemon, kern, auth, etc.
- b) **`/var/log/dmesg`**– Contains kernel ring buffer information. When the system boots up, it prints number of messages on the screen that displays information about the hardware devices that the kernel detects during boot process. These messages are available in kernel ring buffer and whenever the new message comes the old message gets overwritten. We can also view the content of this file using the `dmesg` command.
- c) **`/var/log/auth.log`** – Contains system authorization information, including user logins and authentication mechanism that were used.
- d) **`/var/log/boot.log`** – Contains information that are logged when the system boots
- e) **`/var/log/daemon.log`** – Contains information logged by the various background daemons that runs on the system



- f) **/var/log/dpkg.log** – Contains information that are logged when a package is installed or removed using dpkg command
- g) **/var/log/kern.log** – Contains information logged by the kernel. Helpful to troubleshoot a custom-built kernel.
- h) **/var/log/lastlog**– Displays the recent login information for all the users. This is not an ascii file. We should use lastlog command to view the content of this file.
- i) **/var/log/maillog /var/log/mail.log** – Contains the log information from the mail server that is running on the system. For example, sendmail logs information about all the sent items to this file.
- j) **/var/log/Xorg.x.log** – Log messages from the X
- k) **/var/log/alternatives.log** – Information by the update-alternatives are logged into this log file. On Ubuntu, update-alternatives maintains symbolic links determining default commands.
- l) **/var/log/btmp**(lastb command; shows all bad login attempts)  
**/var/log/wtmp**(displays all users logged in and out since the file is created...last command;login attempts)– This file contains information about failed login attempts. We use the last command to view the btmp file. For example, “last -f /var/log/btmp | more”
- m) **/var/log/cups**– All printer and printing related log messages
- n) **/var/log/anaconda.log** – When we install Linux, all installation related messages are stored in this log file
- o) **/var/log/yum.log** – Contains information that are logged when a package is installed using yum
- p) **/var/log/cron**– Whenever cron daemon(or anacron) starts a cron job, it logs the information about the cron job in this file
- q) **/var/log/secure**– Contains information related to authentication and authorization privileges. For example, sshd logs all the messages here, including unsuccessful login.
- r) **/var/log/wtmp**or **/var/log/utmp**– Contains login records. Using wtmp we can find out who is logged into the system. who command uses this file to display the information.



- s) **/var/log/faillog**– Contains user failed login attempts. We use faillog command to display the content of this file.

Apart from the above log files, /var/log directory may also contain the sub-directories depending on the application that is running on our system. Viewing huge log files for trouble shooting is a tedious routine task. We may use commands like head,tail, sed ,grep etc. to view selected parts and manipulate huge log files.

## 11.4 Managing user accounts-adding & deleting users

Managing user accounts in Linux is an important administrative task. We have already seen in Unit-6 how to create user accounts in GUI mode as well as using commands. To recap the things, ‘adduser’ or ‘useradd’ commands can be used to create user accounts. ‘adduser’ is more user-friendly to use while adding the account details for new users. User accounts can also be created from ‘User accounts’ options in ‘System settings’ in GUI mode. The administrator may set the type of new account to be created as ‘Administrator’ or ‘Standard’.

Before going into the details of process of deleting accounts let’s check how can we see all user accounts that have already been created on the system. ‘ls /home’ command simply lists out the human user accounts who have home directory. Essential information about the user accounts are stored in ‘/etc/passwd’ file.

The /etc/passwd contains one entry per line for each user (or user account) of the system. All fields are separated by a colon (:) symbol. Let’s pick one line from passwd file for user ‘test2’ and learn what the different fields used for.

```
test2:x:1002:1005:Test Kumar,20,,:/home/test2:/bin/bash
```

- a) Username:** It is used when user logs in. It should be between 1 and 32 characters in length. Here ‘test2’ is user name in our case.
- b) Password:** An x character indicates that encrypted password is stored in /etc/shadow file. Encrypted password is not stored in ‘\etc\passwd’ for security reasons.

- c) **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. Here '1002' is user id in our case.
- d) **Group ID (GID):** The primary group ID (stored in /etc/group file). Here '1005' is group-id in our case.
- e) **User ID Info:** The comment field. It allow us to add extra information about the users such as user's full name, phone number etc. Here 'Test Kumar,20,,' is information about the user. A comma means complete details about the user have not been filled but could be filled in some cases.
- f) **Home directory:** The absolute path to the directory the user will be in when they log in. Here '/home/test2' is home directory of the user in our case.
- g) **Command/shell:** The absolute path of a command or shell (/bin/bash). Typically, this is a shell.

User accounts are used not only for actual, human users, but also to run system services and sometimes as owners of system files. '/etc/passwd' contains the entries for all types of users. Now we shall see how to delete the user accounts using commands and in GUI mode.

'**userdel**' or '**deluser**' can be used to remove the user accounts. '**deluser**' is more user-friendly in comparison to '**userdel**'. Suppose we want to remove a user 'test2' forcefully though the user logged in. We may use the following command to do so.

```
$ deluser -force test2
```

To delete user account 'test2' along with his home directory , we may use the command-

## \$ deluser –remove-home test2

To delete user account 'test2' along with his home directory and his personal files which are located in different locations which we are not aware, the following command may be used-

## deluser –remove-all-files test2

To delete user account 'test2' and take backup of his files to a directory (/var/backup) for future use by the company.

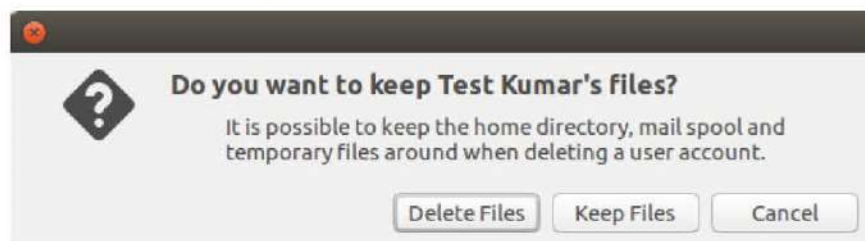
## deluser –backup-to /var/backup test2

To remove the user accounts in GUI mode, we may use the option 'User accounts' in 'System settings'. Fig 11.1 displays the window to manage the user accounts.



Fig 11.1

First we need to click on 'Unlock' option given on right top and enter the password. Then select the user-account to be deleted. Suppose the user-account we want to remove is 'Test Kumar'. Now we need to click on '-' (minus) option given on the left bottom. The system shows the prompt to confirm if the administrator wants to keep user's files as shown in Fig 11.2





**Fig 11.2**

In such a way , a particular user account may be deleted. The screen-shots shown are taken in Ubuntu Linux operating system. The procedure is similar in other distributions of Linux.

If we check the file '/etc/passwd' after deleting a user-account, we can easily see that the entry corresponding to that user is removed from the file.

## **11.5 Changing permissions and ownerships**

Linux is a multi-user operating system that can be used in mainframes and servers.

This raises security concerns as unsolicited or **malign user can corrupt, change**

**or remove crucial data.** For effective security , Linux divides authorization into two

levels-**Permission and Ownership**

The concept of permissions and ownership is crucial in Linux . Let us recap permissions and ownerships in brief .

**Three types of permissions-**

- a) Read -authority to open and read a file, ability to list contents of a directory. Abbreviated as 'r'.
- b) Write - authority to modify the contents of a file, authority to add , remove and rename files stored in the directory. Abbreviated as 'w'.
- c) Execute- authority to run or execute the program. Abbreviated as 'x'.

To represent all types of permissions , we use 'a' .

**Three types of Owners-**

- a) User(or owner)- person who creates a file/directory becomes the user/owner of the file. Abbreviated as 'o'
- b) Group- all users belonging to a group will have the same access permissions to the file.
- c) Other- other people except file owner and group members.



Let us use command 'ls -l' to see the permissions assigned for a file called 'abc'

```
$ ls -l abc
-rw-rw-r-- 1 amreshamresh 12 Mar 14 15:49 abc
```

First part of the output contains permissions assigned to different types of owners. First '-' represents that it is a regular file and rest of nine symbols represents read, write and execute permissions for owner, group members and others respectively. If some symbol is missing and a '-' is given there, it means that permission has not been given to that type of owner.

Now let us see how changing permissions and ownerships work.

**Changing permissions-** permissions can be changed with help of 'chmod' command that can be changed in two ways- **absolute mode and relative mode.**

**Changing permission in absolute mode-** In this mode file permissions are not represented as characters but a three digit octal number. General syntax of using chmod in absolute mode is -

#### **SchmoddddFileOrDirName**

Three 'd's are octal digits representing permissions for owner, group members, others respectively. The table below, shows octal number representation for different permissions types.

| <b>Octal Number</b> | <b>Permission Type</b> | <b>Symbol</b> |
|---------------------|------------------------|---------------|
| 0(000)              | No Permission          | ---           |
| 1 (001)             | Execute                | --x           |
| 2 (010)             | Write                  | -w-           |
| 3 (011)             | Execute +              | -wx           |

|         |                             |     |
|---------|-----------------------------|-----|
|         | Write                       |     |
| 4 (100) | Read                        | r-- |
| 5 (101) | Read +<br>Execute           | r-x |
| 6 (110) | Read<br>+Write              | rw- |
| 7 (111) | Read +<br>Write<br>+Execute | rwx |

Let us use `chmod` command to change the permissions to '764' for file 'abc' in absolute mode

```
$chmod 764 abc
$ls -l abc
-rwxrw-r-- 1 amreshamresh 12 Mar 14 15:49 abc
```

Assigning absolute permission as 764 means – 7(111=rwx), all types of permissions to owner , 6(110=rw-), read and write permissions to group members,

4(100=-x),execute permission only to others. The same can be observed from the

output of 'ls -l' command above. This is how we can change the permissions on a

file by assigning an absolute number. Setting permissions to 666 allows everyone

to read and write to a file or directory. Setting permissions to 777 allows everyone

read, write, and execute permission. These permissions could allow tampering with

sensitive files, so in general, it is not a good idea to use these settings.

**Changing permission in relative mode-** In the Absolute mode we change permissions for all types of owners. In the relative mode we can modify specific permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions

| Operator | Description                                                    |
|----------|----------------------------------------------------------------|
| +        | Adds a permission to a file or directory                       |
| -        | Removes the permission                                         |
| =        | Sets the permission and overrides the permissions set earlier. |

Suppose we want to remove the write permission from the group members for the file 'abc'.

```
$chmod g-w abc
```

```
$ ls -l abc
```

```
-rwxr--r-- 1 amreshamresh 12 Mar 14 15:49 abc
```

We can easily see that write permission from group members is gone.

So general syntax for using chmod in relative mode can be written as

```
$ chmod u*p fileOrDirName
```

where 'u' may be replaced with any of o(owner),g(group members) , o(others) and a(all). '\*' may be replaced with any of +(to add permission), -(to remove permission) , =(sets the permission and overrides the permissions set earlier).

We should remember that file permissions are a security feature. Whenever we allow anyone else to read, write to, and execute files, we are increasing the risk of files being tampered with, altered, or deleted. As a rule, we should only grant read , write and execute permissions to those who truly need them.

### **changing ownerships-**

For changing the ownership of a file/directory , we may use the 'chown' command in following manner:

```
chown userfile Name
```

In case we want to change the user as well as group for a file or directory, we may use the command

**chownuser:groupfileName**

Let us understand this with help of an example. We check the file owner and group owner of file 'abc'.

```
$ ls -l abc
```

```
-rwxr--r-- 1 amreshamresh 12 Mar 14 15:49 abc
```

Here file owner's name is 'amresh' and group owner's name is also 'amresh'.

Let's make test2 as the owner of this file 'abc'.

```
$ chown test2 abc
```

```
$ ls -l abc
```

```
-rwxr--r-- 1 test2 amresh 12 Mar 14 15:49 abc
```

In the output of 'ls' command ,we can see that ownership of the file 'abc' has been given to 'test2'.

In case , we want to change group-owner only , we may use the 'chgrp' command in following manner

**chgrpNewGOwnerFileName**

'chgrp' stands for change group.

To assign the group ownership of 'abc' to newGroup, we may use command

```
$ chgrpnewGroupabc
```

```
$ ls -l abc
```

```
-rwxr--r-- 1 test2 newGroup 12 Mar 14 15:49 abc
```



## 11.6 Creating and managing groups, modifying group attributes

**Creating and managing groups, modifying group attributes are core systems administration tasks.** The `/etc/group` file contains a list of groups, each on a separate line. Each line is a four field, colon delimited list including the following information:

- **Group name** — The name of the group. Used by various utility programs as a human-readable identifier for the group.
- **Group password** — If a lower case `x` is in this field, then shadow group passwords are being used.
- **Group ID (GID)** — The numerical equivalent of the group name. It is used by the operating system and applications when determining access privileges.
- **Member list** — A comma delimited list of the users belonging to the group.

Here is an example line from `/etc/group`:

```
group5:x:502:Ram,Shyam,Geeta
```

This line shows that the `group5` group is using shadow passwords, has a GID of 502, and that `Ram,Shyam` and `Geeta` are members.

We have already seen in Unit 6 how to create a new group in Linux using commands and in GUI mode. Let's recap those things and see how to manage groups and how to modify group attributes. `'addgroup'` or `'groupadd'` can be used to create a new group.

```
$ sudoaddgroup group10
[sudo] password for amresh:
Adding group `group10' (GID 1003) ...
```

**Done.**

Here we have created a new group `'group10'`. If we want to add a new group with specific group-id, we can do in following manner.

```
$ sudoaddgroup -gid 3450 def
Adding group `def' (GID 3450) ...
```

**Done.**

Almost each Linux distribution provides the facility to create groups and manage existing groups in GUI mode. Ubuntu has utility called ‘User Settings’ that helps us to manage groups and users in very simple way. This tool can also be installed using command ‘sudo apt-get install gnome-system-tools’ if it is not already there.

### **Managing groups and modifying group attributes-**

‘deluser’ or ‘userdel’ can be used to remove an existing group. To delete a group called ‘def’ we may use the command

```
$ sudodelgroupdef
```

```
[sudo] password for amresh:
```

```
Removing group `def' ...
```

**Done.**

To modify the group attributes, we use command ‘groupmod’. With ‘-g’ option we can change the group-id of a group using ‘groupmod’ command.

```
$ groupmod -g 789 groupName
```

The above command will update the group-id of group called ‘groupName’ to 789. With ‘-n’ option we can change the groupname of a group. The following command will change the name of group called ‘OldGroup’ to ‘NewGroup’

```
$groupmod -n OldGroupNewGroup
```

## **11.7 Self Learning Exercise**

- Q.1 ‘/etc/passwd’ directory contains :
- a) password of all users in plaintext
  - b) user home directory only
  - c) user ID number only
  - d) user name , user ID number , user home directory and other details
- Q.2 Using ‘chmod664fl’ means:
- a) Everyone can read, group can execute only and the owner can read

and write 'fl'.

b) Everyone can read, owner and group owner can read and write 'fl'.

c) Everyone can read, group including owner can write, owner alone can execute 'fl'.

d) Everyone can read, owner and group owner can write 'fl'.

Q.3 What is the purpose of using 'chgrp' command?

a) To rename a group.

b) To change group-id of a group.

c) To change the group ownership of a group.

d) To delete a group.

## 11.8 Summary

In this unit we saw in brief how we can perform basic administrative tasks like installing, updating and removing a software using commands and in GUI mode. Every Linux distribution has /etc directory that is more or less equivalent to control panel in Windows operating system. It contains configuration files for managing different types of services. /etc/passwd, /etc/shadow and /etc/group are important configuration files that have stored details related to users, encrypted passwords and groups respectively.

New users can be created using adduser/useradd commands and existing users can be removed using deluser/userdel commands. To create new group, addgroup/groupadd can be used and delgroup/groupdel can be used to remove a group. 'groupmod' command can be used to modify group attributes.

There are three types of ownership defined as owner, group owner and others for files and three types of permissions are defined as read, write and execute. 'chown' command is used to change the ownership and 'chmod' is used to change the permissions assigned to three types of owners.

## 11.9 Glossary

|          |     |                                    |
|----------|-----|------------------------------------|
| Chown    | --- | command to change the ownership    |
| Chmod    | --- | command to change the permissions  |
| Groupmod | --- | command to modify group attributes |

|                   |     |                                  |
|-------------------|-----|----------------------------------|
| adduser/useradd   | --- | command to create new user       |
| addgroup/groupadd | --- | command to create new group      |
| deluser/userdel   | --- | command to delete existing user  |
| delgroup/groupdel | --- | command to delete existing group |

## 11.10 Answers to Self-Learning Exercise

- Q.1 (d)  
 Q.2 (b)  
 Q.3 (c)

## 11.11 Exercises

- Q.1 Command that can be used to change group attributes of a group:  
 a) groupmod  
 b) gmod  
 c) modgrp  
 d) chmod
- Q.2 Which command is used to change ownerships of files and directories?  
 a) change  
 b) chgrp  
 c) chown  
 d) set
- Q.3 The permission string `-rwxrwx-r--` represented in absolute mode will be  
 a) 746  
 b) 764  
 c) 744  
 d) 711
- Q.4 File `/etc/shadow` contains-  
 a) password of users in encrypted form  
 b) password of users in plaintext  
 c) configuration files related to booting  
 d) child-parent relationship of all processes



- Q.5 The following command does-  
**\$ chown tuser xyz**
- assigns file-ownership to 'tuser' for file 'xyz'
  - resets group-ownership to 'tuser' for file 'xyz'
  - changes file-ownership and group-ownership both for file 'xyz'
  - changes read and write permissions for file 'xyz'
- Q.6 Which of the following statements is incorrect?
- A package called 'weka' can be installed using command –  
`sudo apt-get install weka`
  - Most of Linux distributions provide the facility to manage the users and groups using commands and GUI mode as well.
  - 'useradd' command can be used to add user and 'userdel' command can be used to remove an existing user.
  - '/etc/group' contains the password of group members of all groups in plaintext.
- Q.7 File '**/var/log/auth.log**' contains
- mail server logs
  - kernel logs
  - authentication logs
  - boot logs

### 11.12 Answers of Exercises

- |         |         |
|---------|---------|
| Q.1 (a) | Q.5 (a) |
| Q.2 (c) | Q.6 (d) |
| Q.3 (b) | Q.7 (c) |
| Q.4 (a) |         |

## References and Suggested Readings

1. Linux: The Complete Reference 6th Edition, *Richard Petersen*, McGraw Hill Education
2. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
3. [www.askubuntu.com](http://www.askubuntu.com)
4. [http://www.linfo.org/etc\\_skel.html](http://www.linfo.org/etc_skel.html)
5. [http://linuxcommand.org/lc3\\_lts0060.php](http://linuxcommand.org/lc3_lts0060.php)
6. <https://bash.cyberciti.biz/>
7. <https://www.howtogeek.com>
8. <http://internal.math.arizona.edu/services/computing/linux/man>
9. <https://www.unixmen.com/managing-your-services-and-processes-in-linux/>
10. <https://www.ibm.com/developerworks/library/l-config/>
11. [http://www.linuxtopia.org/online\\_books/introduction\\_to\\_linux/linux\\_The\\_mostimportant\\_configuration\\_files.html](http://www.linuxtopia.org/online_books/introduction_to_linux/linux_The_mostimportant_configuration_files.html)
12. [http://how-to.wikia.com/wiki/Guide\\_to\\_linux\\_configuration\\_files](http://how-to.wikia.com/wiki/Guide_to_linux_configuration_files)
13. <https://bash.cyberciti.biz/guide/etc/profile>
14. [www.linfo.org/](http://www.linfo.org/)
15. [https://en.wikiversity.org/wiki/System\\_administration](https://en.wikiversity.org/wiki/System_administration)
16. [www.cyberciti.biz/faq/linux-log-files-location-and-how-do-i-view-logs-files/](http://www.cyberciti.biz/faq/linux-log-files-location-and-how-do-i-view-logs-files/)
17. <https://www.javacodegeeks.com/2013/12/common-linux-log-files-name-and-usage.html>
18. [https://access.redhat.com/documentation/en-US/Red\\_Hat\\_Enterprise\\_Linux/4/html/System\\_Administration\\_Guide/](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/System_Administration_Guide/)
19. <http://www.guru99.com/file-permissions.html>
20. [https://www.tutorialspoint.com/unix\\_commands/groupmod.htm](https://www.tutorialspoint.com/unix_commands/groupmod.htm)

# UNIT-12

## User Accounts

### Structure of the Unit

- 12.0 Objective
- 12.1 Introduction
- 12.2 Disable user's account
- 12.3 Creating File System
- 12.4 Mounting File System
- 12.5 Checking and monitoring system performance
- 12.6 File Security and permissions
- 12.7 Super User su control
- 12.8 Self Learning Exercise
- 12.9 Summary
- 12.10 Answers to Self-Learning Exercise
- 12.11 Exercise

### 12.0 Objective

After reading this chapter you will be able to understand the following:

- User account and its disable process
- File system creation and mounting
- System performance checking and monitoring
- Security of file and its permissions
- Controlling using su as Super User

## 12.1 Introduction

Linux is a multi-user operating system, which means that more than one user can use Linux at the same time. Linux provides a beautiful mechanism to manage users in a system. One of the most important roles of a system administrator is to manage the users and groups in a system. A user or account of a system is uniquely identified by a numerical value called the UID (unique identification number). There are two types of users – the root or super user and normal users. A root or super user can access all the files, while the normal user has limited access to files. A super user can add, delete and modify a user account. The full account information is stored in the */etc/passwd* file and a hash password is stored in the file */etc/shadow*.

## 12.2 Disable user's account

Linux systems allow you to disable access to particular user account without changing anything from the account. This might be useful if you do not want to remove user account permanently but, you just want it disabled and no longer able to use the system. The disabled user will still receive emails for example, but he will not be able to login and check them out. A user account can be temporarily disabled or permanently removed.

### Temporary disable user account:

#### 1. First method editing */etc/shadow*.

Administrator can disable or lock a user account temporarily by just putting an asterisk "\*" at the beginning of the second field in the file */etc/shadow*. This means that "\*" won't permit login for this account. Whenever you want to enable the account, just erase the asterisk and the user account is back in operation with its old password.

For example you want to disable user "Mukesh" then you can do this as follows:

```
#vi /etc/shadow
Mukesh:*1arMEFm6$fhA1puOU422HiSL5aggLI:11193:0:99999:7:-1:-
1:134539228
```

Here, the second field is the encrypted password.



You can replace the password with “\*” or “!”. This will render user account inaccessible and it will mean that no login is permitted for the user.

```
#vi /etc/shadow
Mukesh*:13852:0:99999:7:::
```

However, main disadvantage of this method is that the password will be lost in the case we will want to enable it again later.

## 2. Second method using passwd command

Passwd command can be used to disable the user account.

```
#passwdMukesh -l
```

Output

```
“Password changed.”
```

Above command changes the shadow file and adds “!” in front of the user password:

```
Mukesh:!!1eFd7EIOg$EeCk6XgKktWSUgi2pGUpk.:13852:0:99999:7:::
```

Now in case, if you want enable the account just unlock it using `-u` option as follows:

```
#passwd Tom -u
```

You can also enable account by removing manually the “!” character from the user’s password line in `/etc/shadow`.

## 3. Permanently remove user account

You can permanently remove the user; just run `userdel` command.

```
#userdelMukesh
```

Or

```
#userdel -r Mukesh
```

Make sure to check home of the user before running this command.

## 12.3 Creating File System

A **File System** is a way of storing content inside a disk partition. There are many kinds of file systems, each with different characteristics. Linux supports a

huge number. Proprietary operating systems generally only support their own types of file systems.

Linux uses the `mkfs` command to create file systems and `mkswap` command to swap space. The `mkfs` command is actually a front end to several filesystem-specific commands such as `mkfs.ext3` for ext3, `mkfs.ext4` for ext4 and `mkfs.reiserfs` for ReiserFS.

To find out what file system support is already installed on your system? Use the `ls /sbin/mk*` command. An example is shown below-

Filesystem creation commands

```
[ian@echidna ~]$ ls /sbin/mk*
/sbin/mkdosfs /sbin/mkfs.ext2 /sbin/mkfs.ntfs
/sbin/mke2fs /sbin/mkfs.ext3 /sbin/mkfs.vfat
/sbin/mkfs /sbin/mkfs.ext4 /sbin/mkfs.xfs
/sbin/mkfs.btrfs /sbin/mkfs.ext4dev /sbin/mkhomedir_helper
/sbin/mkfs.cramfs /sbin/mkfs.msdos /sbin/mkswap
```

You will notice various forms of some commands. For example, you will usually find that the files `mke2fs`, `mkfs.ext2`, and `mkfs.ext3` are identical, as are `mkreiserfs` and `mkfs.reiserfs`. Filesystems that may be needed to boot the system will usually use hard links to provide the different names for the same file. Filesystems that cannot be used for the `/filesystem` in Linux, such as `vfat` or `msdos`, may use symbolic links instead.

There are a few common options for all `mkfs` commands. Options that are specific to the type of filesystem being created are passed to the appropriate creation command, based on the type of filesystem specified in the `-type` option. Our examples use `mkfs -type`, but you may use the other forms directly with equal effect. For example, you may use `mkfs -type ext2`, `mk2fs`, or `mkfs.ext2`. For the manual pages for a specific filesystem, use the appropriate `mkfs` command as the name, for example, `man mkfs.ext3`. Many of the values displayed in the output examples below can be controlled by options to `mkfs`.

## 12.4 Mounting File System

Data on a computer, as you may know, is stored in binary form as a series of 1s and 0s. The way these are stored on a device and their structure is called the "filesystem". In Linux devices are referenced in /dev. Data is not actually stored on a device so you cannot access this data by going into /dev, this is because it is stored inside the filesystem on the device so you need to access these filesystems somehow. Accessing such filesystems is called "mounting" them, and in Linux (like any UNIX system) you can mount filesystems into any directory, that is, make the files stored in that filesystem accessible when you go into a certain directory. These directories are called the "mount points" of a filesystem. In other systems this is done differently.

Linux only has one top-level volume which is kept in the system's RAM. It too has a root, but since there is only one top-level volume there is no point giving it a label (as there is nothing to distinguish it from) which means that all files in a Linux system are accessed via simply "/". This top-level volume is kept in RAM, but the files themselves are stored on various drives.

### Mount/Unmount Filesystems

#### *Mounting*

The command

```
$mount
```

is responsible for mounting filesystems. The syntax for this command is quite simple (remember that mount must be run with super user privileges to change the system) so:

```
$sudo mount /dev/sda1 /mnt
```

will mount the filesystem on /dev/sda1 (which may be a USB drive, a SATA drive or a SCSI drive) into the folder /mnt. This means that going into /mnt will show you the filesystem which is on /dev/sda1.

Many options can be given to mount. A useful option is the "type" option, when automatic filesystem-type detection fails. An example would be:

```
$sudo mount /dev/fd0 /floppy -t vfat
```

This command tells mount to put the filesystem on the first floppy disk into the folder /floppy, and tells it to treat the filesystem as a FAT filesystem. If the wrong type is given then mount will not mount the filesystem and you will be told of the error.

To mount a filesystem contained in a file using the loopback device the command would look like this

```
$sudo mount Filesystem.img /home/user/MyFilesystem -o loop
```

To create a filesystem image you can either "dump" an existing filesystem into a file, for example by using the command:

```
$dd if=/dev/hdc3 of=/home/user/Filesystem.img
```

Alternatively you can create an empty file by using:

```
$dd if=/dev/zero of=/home/user/MyFilesystem.img bs=512 count=100000
```

and then creating a filesystem on this file as if it were a drive by using the command:

```
$mkfs.ext2 -j MyFilesystem.img
```

An interesting ability of mount is that it's ability to move specific parts of a filesystem around. For example:

```
$sudo mount --bind /mnt/Files/Music /home/user/Music
```

will let the folder "/mnt/Files/Music" also be accessible in /home/user/Music. If you wish to "move" a folder (no data is copied or removed, it is merely displayed in a different place) then use:

```
$sudo mount --move /mnt/Files/Music /home/user/Music
```

### ***Unmounting***

Unmounting is done through the "umount" command, which can be given a device or a mount point so:

```
$sudount /mnt
```

```
$sudount /dev/hda1
```

would both unmount the filesystem on /dev/hda1 if it is mounted on /mnt.



Remember that a filesystem cannot be in use when it is unmounted, otherwise unmount will give an error. If you know it is safe to unmount a filesystem you can use:

```
$sudo umount -l /mountpoint
```

## 12.5 Checking and monitoring system performance

Computer operating systems monitor their resources constantly. In a system, processes are the main resource owners, and as such, most monitoring is done at the process level. This information is used by operating systems while they are running to perform effective memory management, scheduling, multiprogramming, and many other important decisions.

In addition, performance monitoring is useful while developing and refining systems and it provides user support during everyday operation. Records of operating system and process performance can be used to quantify changes to the system and allow accurate comparisons to other systems. They can also be used to predict the performance of similar systems and type of performance gains may be expected in the future.

This list of commands shown here are very enough for you to pick the one that is suitable for your monitoring scenario.

### 1. Top – Linux Process Monitoring

Linux Top command is a performance monitoring program which is used frequently by many system administrators to monitor Linux performance and it is available under many Linux/Unix like operating systems. The top command used to display all the running and active real-time processes in ordered list and updates it regularly. It display CPU usage, Memory usage, Swap Memory, Cache Size, Buffer Size, Process PID, User, Commands and much more as shown in figure 12.1. It also shows high memory and CPU utilization of running processes.

```
$top
```

```

top - 11:36:04 up 1 day, 22:51, 2 users, load average: 0.06, 0.11, 0.09
Tasks: 141 total, 1 running, 139 sleeping, 0 stopped, 1 zombie
Cpu(s): 0.7%us, 0.5%sy, 0.0%ni, 98.8%id, 0.0%wa, 0.0%hi, 0.0%st, 0.0%wt
Mem: 1021108k total, 982904k used, 38204k free, 134576k buffers
Swap: 2046968k total, 0k used, 2046968k free, 599576k cached

 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
25454 root 20 0 397m 107m 13m S 2.3 10.8 25:19.18 skype
269 root 20 0 0 0 0 S 0.3 0.0 0:51.24 scsi_ah_1
1 root 20 0 2872 1400 1260 S 0.0 0.1 0:00.89 init
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root RT 0 0 0 0 S 0.0 0.0 0:00.16 migration/0
4 root 20 0 0 0 0 S 0.0 0.0 0:51.23 ksoftirqd/0
5 root RT 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
6 root RT 0 0 0 0 S 0.0 0.0 0:00.33 watchdog/0
7 root RT 0 0 0 0 S 0.0 0.0 0:00.10 migration/1
8 root RT 0 0 0 0 S 0.0 0.0 0:00.00 migration/1
9 root 20 0 0 0 0 S 0.0 0.0 6:44.34 ksoftirqd/1
10 root RT 0 0 0 0 S 0.0 0.0 0:00.27 watchdog/1
11 root 20 0 0 0 0 S 0.0 0.0 0:04.05 events/0
12 root 20 0 0 0 0 S 0.0 0.0 0:04.87 events/1
13 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cgroup
14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khelper
15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 netns
16 root 20 0 0 0 0 S 0.0 0.0 0:00.00 async/mgr
17 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pm
18 root 20 0 0 0 0 S 0.0 0.0 0:00.23 sync_supers

```

Figure 12.1

## 2. VmStat – Virtual Memory Statistics

Linux VmStat command used to display statistics of virtual memory, kernel threads, disks, system processes, I/O blocks, interrupts, CPU activity and much more. By default vmstat command is not available under Linux systems you need to install a package called sysstat that includes a vmstat program. The common usage of command format is as shown in figure 12.2.

```

vmstat

procs -----memory----- ---swap-- ----io---- --system--
-----cpu-----

r bswpd free inact active si so bi bo in cs us
sy id wast

1 0 0 810420 97380 70628 0 0 115 4 89 79
1 6 90 3 0

```

Figure 12.2

## 3. Lsof – List Open Files

Lsof command used in many Linux/Unix like system that is used to display list of all the open files and the processes. The open files included are disk files, network sockets, pipes, devices and processes. One of the main reasons for using this command is when a disk cannot be unmounted and displays the error that files are being used or opened. With this command you can easily identify which files are in use. The most common format for this command is as shown in figure 12.3.

```

lsof
COMMAND PID USER FD TYPE DEVICE SIZE
NODE NAME
init 1 root cwd DIR 104,2 4096 2 /
init 1 root rtdDIR 104,2 4096 2 /
init 1 root txt REG 104,2 38652
177110339 /sbin/init
init 1 root mem REG 104,2 129900
196453 /lib/ld-2.5.so
init 1 root mem REG 104,2 1693812
196454 /lib/libc-2.5.so
init 1 root mem REG 104,2 20668
196479 /lib/libdl-2.5.so
init 1 root mem REG 104,2 245376
196419 /lib/libsepol.so.1

```

**Figure 12.3**

#### 4. Netstat – Network Statistics

Netstat is a command line tool for monitoring incoming and outgoing network packets statistics as well as interface statistics as shown in figure 12.4. It is very useful tool for every system administrator to monitor network performance and troubleshoot network related problems.

```

netstat-a | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address
State
tcp 0 0 *:mysql *: *
LISTEN
tcp 0 0 *:sunrpc *: *
LISTEN
tcp 0 0 *:realm-rusd *: *
LISTEN
tcp 0 0 *:ftp *: *
LISTEN
tcp 0 0 localhost.localdomain:ipp *: *
LISTEN

```



```

tcp 0 0 localhost.localdomain:smtp *:*
LISTEN

tcp 0 0 localhost.localdomain:smtp *:*
localhost.localdomain:42709 TIME_WAIT

tcp 0 0 localhost.localdomain:smtp *:*
localhost.localdomain:42710 TIME_WAIT
tcp 0 0 *:* *:http
: LISTEN
tcp 0 0 *:* *:ssh
: LISTEN
tcp 0 0 *:* *:https
: LISTEN

```

Figure 12.4

## 5. Iotop – Monitor Linux Disk I/O

Iotop is also much similar to top command and Htop program, but it has accounting function to monitor and display real time Disk I/O and processes as shown in figure 12.5. This tool is much useful for finding the exact process and high used disk read/writes of the processes.

\$iotop

| TID   | PRIO | USER    | DISK READ | DISK WRITE | SWAPIN | IO%    | COMMAND                              |
|-------|------|---------|-----------|------------|--------|--------|--------------------------------------|
| 20869 | be/4 | mongod  | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | mongod -f /etc/mongod.conf           |
| 23239 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | konqueror -minety-ctory file:///root |
| 7183  | be/4 | ubersvn | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | java -Djava.util.-up.Bootstrap start |
| 1805  | be/3 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | [cmic_wq]                            |
| 1812  | be/0 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | [bnx2i_thread/0]                     |
| 7125  | be/4 | ubersvn | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | java -Djava.util.-up.Bootstrap start |
| 7152  | be/4 | ubersvn | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | java -Djava.util.-up.Bootstrap start |
| 25635 | be/4 | clamav  | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | clamd                                |
| 9160  | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | perl /usr/libexec-bmin/miniserv.conf |
| 18100 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | automount                            |
| 8406  | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | squid -D                             |
| 31923 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | Xvnc :1 -desktop - -rfbport 5901 -ph |
| 32127 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | nm-applet --sm-disable               |
| 2125  | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | xulrunner-bin ./application.ini      |
| 8355  | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | crond                                |
| 32168 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | gconfd-2 ?                           |
| 25636 | be/4 | clamav  | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | clamd                                |
| 31932 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | sh /usr/bin/startkde                 |
| 1965  | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | klauncher [kdeinit] --nev-startup    |
| 2064  | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | bt-applet --sm-disable               |
| 32125 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | bt-applet --sm-disable               |
| 2146  | be/4 | apache  | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | php-fpm: pool www                    |
| 8478  | be/4 | xymon   | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | hobbitd_channel --og hobbitd_history |
| 13846 | be/4 | mysql   | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | mysqld --basedir=/b/mysql/mysql.sock |
| 30015 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | (nccsa_auth) /etc/squid/passwd       |
| 32091 | be/4 | root    | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | kdeinit Running...                   |
| 13831 | be/4 | mysql   | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | mysqld --basedir=/b/mysql/mysql.sock |
| 13832 | be/4 | mysql   | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | mysqld --basedir=/b/mysql/mysql.sock |
| 13833 | be/4 | mysql   | 0.00 B/s  | 0.00 B/s   | 0.00 % | 0.00 % | mysqld --basedir=/b/mysql/mysql.sock |

Figure 12.5

## 6. Iostat – Input/Output Statistics



IoStat is simple tool that will collect and show system input and output storage device statistics as shown in figure 12.6. This tool is often used to trace storage device performance issues including devices, local disks, and remote disks such as NFS.

```
iostat
Linux 2.6.18-238.9.1.el5 (tecmint.com) 09/13/2012
avg-cpu: %user %nice %system %iowait %steal %idle
2.60 3.65 1.04 4.29 0.00 88.42
Device: tpsBlk_read/s Blk_wrtn/s Blk_readBlk_wrtn
cciss/c0d0 17.79 545.80 256.52 855159769
401914750
cciss/c0d0p1 0.00 0.00 0.00 5459
3518
cciss/c0d0p2 16.45 533.97 245.18 836631746
384153384
cciss/c0d0p3 0.63 5.58 3.97 8737650
6215544
cciss/c0d0p4 0.00 0.00 0.00 8
0
cciss/c0d0p5 0.63 3.79 5.03 5936778
7882528
cciss/c0d0p6 0.08 2.46 2.34 3847771
3659776
```

**Figure 12.6**

**7. NetHogs – Monitor Per Process Network Bandwidth**

NetHogs is an open source nice small program (similar to Linux top command) that keeps a tab on each process network activity on your system as shown in figure 12.7. It also keeps a track of real time network traffic bandwidth used by each program or application.

```

tecmint@tecmint: ~
NetHogs version 0.8.0

```

| PID  | USER    | PROGRAM                 | DEV  | SENT  | RECEIVED     |
|------|---------|-------------------------|------|-------|--------------|
| 2551 | tecmint | ./r/lib/firefox/firefox | eth0 | 0.448 | 0.744 KB/sec |
| ?    | root    | ..57229-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57228-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57227-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57226-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57213-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57212-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57211-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57210-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57209-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57208-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57205-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57165-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57164-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57163-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57162-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57161-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57153-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57152-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57151-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57150-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57149-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57148-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | ..57145-172.16.16.69:80 |      | 0.000 | 0.000 KB/sec |
| ?    | root    | unknown TCP             |      | 0.000 | 0.000 KB/sec |

Figure 12.7

## 8. iftop – Network Bandwidth Monitoring

iftop is another terminal-based free open source system monitoring utility that displays a frequently updated list of network bandwidth utilization (source and destination hosts) that passes through the network interface on your system as shown in figure 12.8. iftop is considered for network usage, what ‘top‘ does for CPU usage. iftop is a ‘top‘ family tool that monitor a selected interface and displays a current bandwidth usage between two hosts.

```

172.16.25.126 => 172.15.25.125 2.673b 2.84Kb 2.84Kb
<= 1.67b 240c 240b
172.16.25.126 => mddc-01.midcorp.mid-d 269b 861c 86.3b
<= 803b 1.50Kb 1.50Kb
172.16.31.255 => 172.15.21.185 7h 0c 0b
<= 7b 240c 240b
172.16.31.255 => 172.15.21.152 7b 0c 0b
<= 7b 229c 229b
172.16.31.255 => xsus.midcorp.mid-day. 7b 0c 0b
<= 7b 229c 229b
172.16.31.255 => 172.15.21.79 7b 0c 0b
<= 91fb 229c 229b
172.16.31.255 => 172.15.13.159 7b 0c 0b
<= 7b 156c 156b
255.255.255.255 => 172.15.13.9 7b 0c 0b
<= 7b 68c 68b

```

| TX:    | rx:    | pkts: | bytes: | sent:  | recv:  | total: |
|--------|--------|-------|--------|--------|--------|--------|
| 0.000  | 3.68KB | peak: | rates: | 2.953b | 3.68Kb | 3.68Kb |
| 0.000  | 2.93KB |       |        | 1.633b | 2.93Kb | 2.93Kb |
| TOTAL: | 6.61KB |       |        | 4.583b | 6.60Kb | 6.60Kb |

Figure 12.8

## 12.6 File Security and Permissions

Linux file security is quite simplistic in design, yet quite effective in controlling access to files and directories. Directories and the files which are stored in them are arranged in a hierarchical tree structure. Access can be controlled for both

the files and the directories allowing a very flexible level of access. On a Linux system, every file is owned by a user and a group user. There is also a third category of users, those that are not the user owner and don't belong to the group owning the file. For each category of users, read, write and execute permissions can be granted or denied.

LINUX assigns file permissions in three broad categories:

1. The user that owns the file
2. The group that owns the file
3. Everyone else (a.k.a 'world' or 'other')

Each permission category can be assigned three permissions:

21. (r)ead
22. (w)rite
23. e(x)ecute

LINUX permissions can be represented as symbols and as octal digits. Table 1 shows a table of possible values:

*Table 1: Table of Permission Combinations*

| <b>Octal</b> | <b>Symbol</b> | <b>Permission</b>        |
|--------------|---------------|--------------------------|
| 0            | ---           | No Permissions           |
| 1            | --x           | Execute                  |
| 2            | -w-           | Write                    |
| 3            | -wx           | Write and Execute        |
| 4            | r--           | Read                     |
| 5            | r-x           | Read and Execute         |
| 6            | rw-           | Read and Write           |
| 7            | rwX           | Read, Write, and Execute |

The effect of these permissions varies by file type. There are numerous file types, but this section will only deal with the following types:

1. 'l' — Regular file
2. 'd' — Directory

When applied to a regular file, UNIX permissions have the following effect:

1. Read permission allows the file to be opened
2. Write permission allows the file to be modified
3. Execute permission allows the file to be executed directly from the shell, if it is a script or binary file

When applied to a directory, UNIX permissions have the following effect:

1. Read permission allows the directory contents to be listed
2. Write permission allows files or directories to be created or deleted within the directory
3. Execute permission allows the directory contents to be accessed

LINUX permissions closest to the user take precedence. User permissions take priority over group permissions for the user who owns a file or directory, and both user and group permissions take priority over the 'other' permissions for the user and group who own the file.

### ***Setting LINUX File Permissions***

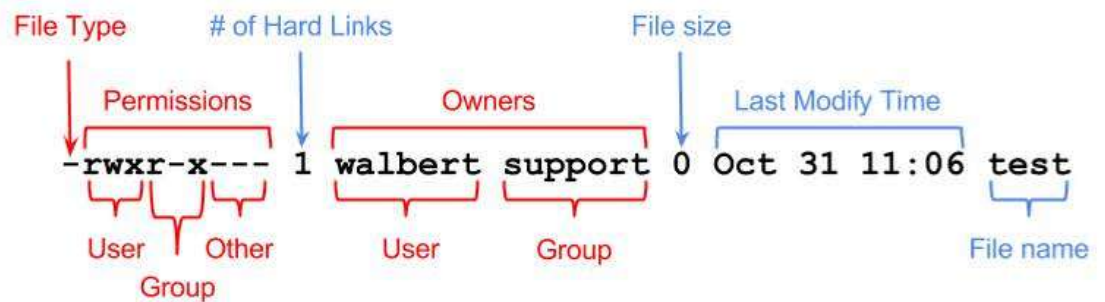
There are three utilities used for managing LINUX permissions:

1. 'ls -l' - Displays a detailed list of directory contents
2. 'chmod' - Changes permissions
3. 'chgrp' - Changes group owners

#### ***'ls -l' — Displays a detailed list of directory contents***

The ls commands outputs a list of files and directories within a given directory, and the '-l' option provides a detailed list of the directory contents (including permissions). Fig. 12.9 shows the output of 'ls -l' on a test file, with each section labeled. The portions of the output related to UNIX file system permissions are colored red:





The output of Figure tells us the following:

1. The '-' file type indicates that this is a regular file
2. 'rwx' permissions for user indicates that the user who owns this file has read, write, and execute permissions for the file
3. 'r-x' permissions for group indicates that the group who owns this file has read and execute permissions for the file
4. '---' permissions for 'other' (i.e., everyone else) means that everyone other than the user or group who owns the file has no access to the file.
5. The file is owned by user 'walbert' and group 'support'

### *'chmod'- change permissions*

The `chmod` command changes the permission on a given file or directory. `chmod` sets permissions in two ways. 1. Using symbols, 2. Using octal values

When using octal values, a series of three digits defines the permissions for user, group, and other in that order. For example, a three-digit octal value of 640 would translate to read and write permissions for the user, read permission for the group, and no permission for others. When using octal values to define permissions, the new octal value entered will replace the existing permissions on the file or directory. The following examples show common uses of `chmod`.

#### ➤ Access for Only Yourself

```
$ chmod 600 test_file
$ ls -l test_file
-rw----- 1 walbertugrad 0 Nov 5 15:58 test_file
```

#### ➤ Write Access for Yourself and Read Access for Your Group

```
$ chmod 640 test_file
```

```
$ ls -l test_file
-rw-r----- 1 walbertugrad 0 Nov 5 15:58 test_file
```

➤ Write Access for Yourself and Read Access for Everyone Else

```
$ chmod 644 test_file
$ ls -l test_file
-rw-r--r-- 1 walbertugrad 0 Nov 5 15:58 test_file
```

➤ Write and Execute Access for Yourself Only

```
$ chmod 700 test_file
$ ls -l test_file
-rwx----- 1 walbertugrad 0 Nov 5 15:58 test_file
```

➤ Write and Execute Access for Yourself, and Execute Access for Everyone

```
$ chmod 755 test_file
$ ls -l test_file
-rwxr-xr-x 1 walbertugrad 0 Nov 5 15:58 test_file
```

➤ Write Access to a Directory for Yourself, and Read Access for Your Group

```
$ chmod 750 test_directory
$ ls -l -d test_directory
drwxr-x--- 2 walbertugrad 2.0K Nov 5 15:59 test_directory/
```

**'chgrp' — Changes group owners**

The `chgrp` command changes the group ownership on a given file or directory. Changing group ownership may be necessary when sharing files and directories with other members of your research group.

➤ Changing the Group Ownership of a File

```
$ chgrp support test_file
$ ls -l test_file
-rw-rw---- 1 walbert support 0 Nov 5 15:58 test_file
```

## 12.7 Super User (su) control

To use su to gain root privileges, you must unlock the root account as follows:

Except for a few instances, there is no need to unlock the root account on Linux suggests that you do not do so. The following command unlocks the root account by assigning a password to it:

```
$ sudo passwd root
```

Enter new UNIX password:

Retype new UNIX password:

```
passwd: password updated successfully
```

If you decide you want to lock the root account after unlocking it, give the command

```
sudo passwd -l root. You can unlock it again with the preceding command.
```

To use su to gain root privileges, you must unlock the root account.

The su (substitute user) utility can spawn a shell or execute a program with the identity and privileges of a specified user. Follow su on the command line with the name of a user; if you are working with root privileges or if you know the user's password, you will then take on the identity of that user. When you give the su command without an argument, su defaults to spawning a shell with root privileges.

When you give the su command to work as root, su spawns a new shell, which displays the # prompt. You can return to your normal status (and your former shell

and prompt) by terminating this shell: Press CONTROL-D or give an exit command. Giving the su command by itself changes your user and group IDs but makes minimal changes to the environment. For example, PATH has the same value as it did before you gave the su command. When you give the command su - (you can use -l or --login in place of the hyphen), you get a root login shell: It is as though you logged in as root. Not only do the shell's user and group IDs match those of root, but the environment is identical to that of root.

The id utility displays the changes in your user and group IDs and in the groups you are associated with:

```
$ id
uid=1002(sam) gid=1002(sam) groups=117(admin),1002(sam)
$ su
Password:
id
uid=0(root) gid=0(root) groups=0(root)
```

## 12.8 Self Learning Exercise

- Q.1 There is following types of users in LINUX
- a) Super user and normal users
  - b) Only super user
  - c) Only normal users
  - d) None of above
- Q.2 Which file is used by Linux distribution to store the encrypted user passwords.
- a) /etc/passwords
  - b) /etc/shadow
  - c) /etc/bin
  - d) /etc/user
- Q.3 mount command is used
- a) to mount a file system.
  - b) to mount a directory system.
  - c) to mount network settings.
  - d) to mount audio and video settings.

## 12.9 Summary

Linux is a multi-user system. In Linux, there are two types of users: system users and regular users. Traditionally, system users are used to run non-interactive or background processes on a system, while regular users used for logging in and running processes interactively. User account can be disabled temporary using various ways.



The `mount` command mounts a storage device or filesystem, making it accessible and attaching it to an existing directory structure. The `umount` command "unmounts" a mounted filesystem, informing the system to complete any pending read or write operations, and safely detaching it.

Monitoring your Linux system is essential in order to be able to improve its performance, locate the source of a problem and take more targeted corrective actions. As is always the case with Linux, there are quite a few tools and many different ways you can utilize to monitor different aspects of your system's performance.

From an end user's perspective, file security is one of the most critical areas of security. Some file security is built into Linux: `chmod` gives you basic security control. `chgrp` is used to change permission of the file to achieve security.

The `superuser` is a special user account used for system administration. Depending on the operating system (OS), the actual name of this account might be `root`, `administrator`, `admin` or `supervisor`. In Linux-like computer OSes, `root` is the conventional name of the user who has all rights or permissions (to all files and programs) in all modes (single- or multi-user).

## 12.10 Answers to Self-Learning Exercise

- Q.1 (a)
- Q.2 (b)
- Q.3 (a)

## 12.11 Exercise

- Q.1 Write the different procedure steps to disable user account temporarily.
- Q.2 Explain the `mount` and `umounts` command with suitable example.
- Q.3 Describe the command `chmod` which is used to change the permissions.
- Q. 4 Explain the command which are used to monitor and check the system performance.

## References and Suggested Readings

1. A Practical Guide to Ubuntu Linux by Mark G. Sobell, Prentice Hall, 2007.
2. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
3. UNIX Shell Programming by Yashwant Kanetkar, BPB Publications, 2003.

# UNIT-13

## Getting System Information

### Structure of the Unit

- 13.0 Objective
- 13.1 Introduction
- 13.2 Getting system information with
  - 13.2.1 uname
  - 13.2.2 host name
  - 13.2.3 disk partitions and sizes
  - 13.2.4 users, kernel
- 13.3 Backup and restore files
  - 13.3.1 Backup Utilities
- 13.4 File reconfiguration hardware with kudzu
- 13.5 Installing and removing packages in Linux
- 13.6 Self Learning Exercise
- 13.7 Summary
- 13.8 Glossary
- 13.9 Answers to Self-Learning Exercise
- 13.10 Exercise
- 13.11 Answers to Exercise

### 13.0 Objective

In this chapter we shall focus upon the following topics

- Getting system information using various utilities
- Backup and Restore Files
- File configuration with kudzu

- Installing and removing packages

## 13.1 Introduction

Since Unix comes in many flavors, the vendors have customized a number of commands to behave in the way they wanted. System information get from various commands like `uname`, `hostname` and `disk partitions` commands. In Linux, user can take the backup and restore the files using related commands. Different packages can be install and uninstalled in Linux using related commands. Kudzu is a hardware probing program (written by Red Hat Linux) which relies on a library of hardware device information. When the computer boots, kudzu detects changes in the running system's hardware configuration, if any, and activates the newly detected hardware (or removal of hardware).

## 13.2 Getting system information

### 13.2.1 Uname

This command displays information about the system. Without arguments, this utility displays the name of the operating system (Linux). The `uname` command within Linux allows you to view system information about your Linux environment.

With the `-a` (all) option, it displays the operating system name, hostname, version number and release date of the operating system, and type of hardware you are using:

```
$ uname -a
```

```
Linux Brother 2.6.22-10-generic #2 SMP Thu Jun 9 18:19:32 UTC 2011 i686
GNU/Linux
```

With `-s` option, it shows the kernel name on its own.

```
$ uname -s
```

```
Linux
```

With `-n` option, this command shows the node name of your computer.

```
$ uname -n
```

```
Brother
```



With `-r` option, it shows the kernel release.

```
$ uname-r
```

```
2.6.22-10-generic
```

With `-v` option, it shows the version of kernel.

```
$ uname-v
```

```
#2 SMP Thu Jun 9 18:19:32 UTC 2011
```

With `-p` option, this command shows the processor type.

```
$ uname-p
```

```
i686
```

With `-o` option, this command shows the operating system.

```
$ uname-o
```

```
GNU/Linux
```

### **13.2.2 Host Name**

The `hostname` utility displays the name of the system you are working on. Use this utility if you are not sure that you are logged in on the right machine.

```
$ hostname
```

```
bravo.example.com
```

With `-i` option, it print the IP address of the computer.

```
$ hostname-i
```

```
192.168.134.128
```

With `-d` option, it prints domain name.

```
$ hostname-d
```

```
example.com
```

With `-s` option, it prints short hostname.

```
$ hostname-s
```

```
bravo
```

### **13.2.3 Disk partitions and sizes**

Like other operating systems, Linux requires a formatted hard disk. Normally, the formatting operation lies outside the operating system's domain and is carried out by special utilities supplied by the vendor. After the disk has been formatted, the next step is to divide it into several *partitions*. Each partition can be considered to be a logically independent disk and accessed by its own device file. The various components of the Linux system are organized into one or more of these partitions.

`fdisk` stands (for "fixed disk or format disk") is most commonly used command-line based disk manipulation utility for a Linux/Unix systems. With the help of `fdisk` command you can view, create, resize, delete, change, copy and move partitions on a hard drive using its own user friendly text based menu driven interface.

The `fdisk` with '-l' option lists all existing disk partitions on your system. The '-l' argument (stands for listing all partitions) is used with `fdisk` command to view all available partitions on Linux. The partitions are displayed by their device names.

```
$fdisk -l
```

```
Disk /dev/sda: 637.8 GB, 637802643456 bytes
```

```
255 heads, 63 sectors/track, 77541 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes
```

| Device    | Boot | Start | End  | Blocks    | Id | System |
|-----------|------|-------|------|-----------|----|--------|
| /dev/sda1 | *    | 1     | 13   | 104391    | 83 | Linux  |
| /dev/sda2 |      | 14    | 2624 | 20972857+ | 83 | Linux  |
| /dev/sda3 |      | 2625  | 4582 | 15727635  | 83 | Linux  |

A specific partition can be seen by writing the device name with command like following example.

```
$fdisk -l /dev/sda
```

```
Disk /dev/sda: 637.8 GB, 637802643456 bytes
```

```
255 heads, 63 sectors/track, 77541 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes
```

| Device | Boot | Start | End | Blocks | Id | System |
|--------|------|-------|-----|--------|----|--------|
|--------|------|-------|-----|--------|----|--------|

```

/dev/sda1 * 1 13 104391 83 Linux
/dev/sda2 14 2624 20972857+ 83 Linux
/dev/sda3 2625 4582 15727635 83 Linux

```

If you would like to view all commands which are available for fdisk. Simply use the following command by mentioning the hard disk name such as /dev/sda as shown below. The following command will give you output similar to below.

```
$fdisk /dev/sda
```

Type 'm' to see the list of all available commands of fdisk which can be operated on /dev/sda hard disk. After, We enter 'm' on the screen, you will see the all available options for fdisk that you can use on the /dev/sda device.

```
$fdisk /dev/sda
```

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to switch off the mode (command 'c') and change display units to sectors (command 'u'). Command (m for help): m

**Command action**

- a toggle a bootable flag
- b edit bsddisklabel
- c toggle the dos compatibility flag
- d delete a partition
- l list known partition types
- m print this menu

Command (m for help):

User can print all partitions in Linux writing p option in help like as follows. you must be on command mode of specific hard disk say /dev/sda. From the command mode, enter 'p' instead of 'm' as we did earlier. As I enter 'p', it will print the specific /dev/sda partition table.

```
$fdisk /dev/sda
```

Command (m for help): p

```
Disk /dev/sda: 637.8 GB, 637802643456 bytes
```

255 heads, 63 sectors/track, 77541 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

| Device    | Boot | Start | End   | Blocks     | Id | System   |
|-----------|------|-------|-------|------------|----|----------|
| /dev/sda1 | *    | 1     | 13    | 104391     | 83 | Linux    |
| /dev/sda2 |      | 14    | 2624  | 20972857+  | 83 | Linux    |
| /dev/sda3 |      | 2625  | 4582  | 15727635   | 83 | Linux    |
| /dev/sda4 |      | 4583  | 77541 | 586043167+ | 5  | Extended |
| /dev/sda5 |      | 4583  | 5887  | 10482381   | 83 | Linux    |

If you would like to delete a specific partition (i.e /dev/sda9) from the specific hard disk such as /dev/sda. You must be in fdisk command mode to do this. Next, enter 'd' to delete any given partition name from the system. As I enter 'd', it will prompt me to enter partition number that I want to delete from /dev/sda hard disk. Suppose I enter number '4' here, then it will delete partition number '4' (i.e. /dev/sda4) disk and shows free space in partition table. Enter 'w' to write table to disk and exit after making new alterations to partition table. The new changes would only take place after next reboot of system. This can be easily understood from the below output.

```
$fdisk /dev/sda
```

```
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
```

```
switch off the mode (command 'c') and change display units to sectors (command 'u').
```

```
Command (m for help): d
```

```
Partition number (1-4): 4
```

```
Command (m for help): w
```

```
The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.
```

```
WARNING: Re-reading the partition table failed with error 16: Device or resource busy. The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8) Syncing disks.
```



If you've free space left on one of your device say /dev/sda and would like to create a new partition under it. Then you must be in fdisk command mode of /dev/sda. Type the following command to enter into command mode of specific hard disk. After entering in command mode, now press "n" command to create a new partition under /dev/sda with specific size. This can be demonstrated with the help of following given output. While creating a new partition, it will ask you two options 'extended' or 'primary' partition creation. Press 'e' for extended partition and 'p' for primary partition. Then it will ask you to enter following two inputs.

First cylinder number of the partition to be created.

Last cylinder number of the partition to be created (Last cylinder, +cylinders or +size).

```
$fdisk /dev/sda
```

```
WARNING: DOS-compatible mode is deprecated. It's strongly recommended
to switch off the mode (command 'c') and change display units to sectors
(command 'u').
```

```
Command (m for help): n
```

```
Command action
```

```
e extended
```

```
p primary partition (1-4)
```

```
e
```

```
Command (m for help): w
```

```
The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.
```

```
WARNING: Re-reading the partition table failed with error 16: Device or
resource busy.
```

```
The kernel still uses the old table. The new table will be used at the next reboot
or after you run partprobe(8) or kpartx(8) Syncing disks.
```

After the new partition is created, it is necessary to format the newly created partition using 'mkfs' command. Type the following command in the terminal to format a partition. Here /dev/sda4 is my newly created partition.

```
$ mkfs.ext4 /dev/sda4
```

After formatting new partition, check the size of that partition using flag 's' (displays size in blocks) with fdisk command. This way you can check size of any specific device.

```
$fdisk -s /dev/sda2
```

```
5194304
```

### 13.2.4 Users

In Linux, everyone who logs on to the system is called a **user**. Users are known to the system by their user-ids. In Linux, however, not every user is created equal. Some users have more capabilities than others. They are known as **super users**. Also known as system administrators, super users have the maximum set of capabilities in the system; they can even change the system itself.

In Linux, there are few commands from which information related to user can be get.

users: This command print the user names of users currently logged in to the current host without showing any much information about source, login time or any other relevant detail

```
$users
```

```
deepakruchikapankajsureshbharat
```

last: This command searches back through the file /var/log/wtmp and displays a list of all users logged in (and out) since that file was created. Names of users and tty's can be given, in which case last will show only those entries matching the arguments.

```
$last -a
```

```
deepakpts/3 Tue Oct 16 18:01 - 18:01 (00:00) 10.10.10.30
```

```
rootpts/2 Tue Oct 16 17:51 still logged in 10.10.10.30
```

```
rootpts/1 Tue Oct 16 14:29 - 18:03 (03:34) 10.10.10.30
```

```
rootpts/3 Tue Oct 16 11:10 - 13:11 (02:00) 10.10.10.30
```

```
rootpts/1 Mon Oct 15 20:30 - 13:21 (16:51) 10.10.10.30
```

```
rootpts/3 Mon Oct 15 18:02 - 18:37 (00:34) 10.10.10.30
```

finger: If no arguments are specified, finger will print an entry for each user currently logged into the system.

```
$finger
```

```
Login Name TtyIdle Login Time Office Office Phone
```

```
deepakpts/3 Oct 16 18:01 (10.10.10.30)
```

```
rootroottty1 4d Oct 12 17:56 (:0)
```

```
rootrootpts/0 6:51 Oct 12 17:57 (:0.0)
```

```
rootrootpts/1 2:08 Oct 16 14:29 (10.10.10.30)
```

who: This command shows currently logged in users with time details

```
$who -u
```

```
root tty1 2012-10-12 17:56 old 1960 (:0)
```

```
rootpts/0 2012-10-12 17:57 06:51 2376 (:0.0)
```

```
rootpts/1 2012-10-16 14:29 02:09 3094 (10.10.10.30)
```

whoami: Prints the user name associated with the current effective user ID

```
$ whoami
```

```
Root
```

### 13.3 Backup and restoring the files

One of the most oft-neglected tasks of system administration is making backup copies of files on a regular basis. The backup copies are vital in three instances: when the system malfunctions and files are lost, when a catastrophic disaster (fire, earthquake, and so on) occurs, and when a user or the system administrator deletes or corrupts a file by accident. It is a good idea to have a written backup policy and to keep copies of backups offsite (in another building, at home, or at a different facility or campus) in a fireproof vault or safe.

The time to start thinking about backups is when you partition the disk. Make sure the capacity of the backup device and your partition sizes are comparable.

You must back up filesystems regularly. Backup files are usually kept on magnetic tape, external hard disk, or another removable medium. Alternatively, you can keep backup files on a remote system.

The backup procedure typically slows the system for users, takes a certain amount of your time, and requires that you have and store the media holding the backup.

Another question is when to take the backup. Depending on the use of the system, sometime in the middle of the night can work well. Then the backup is least likely to affect users, and the files are not likely to change as they are being read for backup.

### **13.3.1 Backup Utilities**

A number of utilities are available to help you back up a system, and most work with any media. Most Linux backup utilities are based on one of the archive programs—tar or cpio—and augment these basic programs with bookkeeping support for managing backups conveniently.

You can use any of the tar, cpio, or dump/restore utilities to construct full or partial backups of a system. Each utility constructs a large file that contains, or archives, other files. In addition to file contents, an archive includes header information for each file it holds. This header information can be used when extracting files from the archive to restore file permissions and modification dates. An archive file can be saved to disk, written to tape, or shipped across the network while it is being created.

#### **tar: Archives Files**

The tar (tape archive) utility writes files to and retrieves files from an archive; it can compress his archive to conserve space. If you do not specify an archive device, tar writes to standard output and reads from standard input. With the `-f` (`--file`) option, tar uses the argument to `-f` as the name of the archive device. You can use this option to refer to a device on another system on the network.

The following command displays a complete list of options:

```
$ tar --help | less
```

Most options for tar can be given either in a short form (a single letter) or as a descriptive word. Descriptive-word options are preceded by two hyphens, as in `--help`. Single-letter options can be combined into a single command line



argument and need not be preceded by a hyphen (for consistency with other utilities, it is good practice to use the hyphen anyway). Although the following two commands look quite different, they specify the same tar options in the same order. The first version combines single-letter options into a single command line argument; the second version uses descriptive words for the same options:

```
$ sudo tar -ztvf /dev/st0
```

```
$ sudo tar --gzip --list --verbose --file /dev/st0
```

Both commands tell tar to generate a (v, verbose) table of contents (t, list) from the tape on /dev/st0 (f, file), using gzip (z, gzip) to decompress the files. Unlike the original Unix tar utility, the GNU version strips the leading / from absolute pathnames.

The options in Table 13-1 tell the tar program what to do. You must include exactly one of these options in a tar command. The -c, -t, and -x options are used most frequently.

Table 17-1 tar options

| Option                | Effect                                     | Option               | Effect                                                                                        |
|-----------------------|--------------------------------------------|----------------------|-----------------------------------------------------------------------------------------------|
| <b>--append (-r)</b>  | Appends files to archive                   | <b>--help</b>        | <b>Displays a help list of tar options</b>                                                    |
| <b>--create (-c)</b>  | Creates a new archive                      | <b>--list (-t)</b>   | <b>Lists the files in an archive</b>                                                          |
| <b>--delete</b>       | Deletes files in an archive (not on tapes) | <b>--update (-u)</b> | <b>Like the -r, but the file is not appended if a newer version is already in the archive</b> |
| <b>--extract (-x)</b> | <b>Extracts files from an archive</b>      |                      |                                                                                               |

### cpio: Archives Files

The cpio (copy in/out) program is similar to tar but can read and write archive file in various formats, including the one used by tar. Normally cpio reads the names of the files to add to the archive from standard input and produces the

archive file as standard output. When extracting files from an archive, it reads the archive as standard input.

As with tar, some options can be given in both a short, single-letter form and a more descriptive word form. However, unlike with tar, the syntax of the two forms in cpio differs by additional information. In the short form, you must include a SPACE between the option and the additional information; with the word form, you must separate the two with an equal sign and no SPACES.

Running cpio with the `—help` option displays a complete list of options.

### Performing a Simple Backup

When you prepare to make a major change to a system, such as replacing a disk drive, upgrading to a new release, or updating the Linux kernel, it is a good idea to archive some or all of the files so you can restore any that become damaged if something goes wrong. For this type of backup, tar or cpio works well. For example, if you have a SCSI tape drive as device `/dev/st0` (or it could be a hard disk at `/dev/hdb`) that is capable of holding all the files on a single tape, you can use the following commands to construct a backup tape of the entire system:

```
$ cd /
$ sudo tar -cf /dev/st0 .
```

All the commands in this section start by using `cd` to change to the root directory so you are sure to back up the entire system. The tar command then creates an archive (c) on the device `/dev/st0` (f). To compress the archive, replace the preceding tar command with the following command, which uses `j` to call `bzip2`:

You can back up a system with a combination of `find` and `cpio`. The following commands create an output file and set the I/O block size to 5120 bytes (the default is 512 bytes):

```
$ cd /
$ sudo find . -depth | cpio -oB> /dev/st0
```

The next command restores the files in the `/home` directory from the preceding backup. The options extract files from an archive (`-i`) in verbose mode, keeping the modification times and creating directories as needed.

```
$ cd /
```

```
$ sudo cpio -ivmd /home/* < /dev/st0
```

### **Back Up and Restore Filesystems (dump and restore)**

The dump utility (part of the dump package) first appeared in UNIX version 6. It backs up either an entire ext2 or ext3 filesystem or only those files that have changed since a recent dump. The restore utility can then restore an entire filesystem, a directory hierarchy, or an individual file. You will get the best results if you perform a backup on a quiescent system so that the files are not changing as you make the backup.

The next command backs up all files (including directories and special files) on the root (/) partition to SCSI tape 0. Frequently there is a link to the active tape drive, named /dev/tape, which you can use in place of the actual entry in the /dev directory.

```
$ sudo dump -0uf /dev/st0 /
```

The next command makes a partial backup containing all files that have changed since the last level 0 dump. The first argument (1) specifies a level 1 dump:

```
$ sudo dump -1uf /dev/st0 /
```

To restore an entire filesystem from a dump backup, first restore the most recent complete (level 0) backup. Perform this operation carefully because restore can overwrite the existing filesystem. Change directories to the directory the filesystem is mounted on and give a restore command as shown following:

```
$ cd /xxx
```

```
$ sudo restore -if /dev/st0
```

## **13.4 File reconfiguration hardware with kudzu**

Kudzu is a hardware probing program (written by Red Hat Linux) which relies on a library of hardware device information. When the computer boots, kudzu detects changes in the running system's hardware configuration, if any, and activates the newly detected hardware (or removal of hardware).

Kudzu only runs at boot time, and then exits. There is no performance penalty during normal operation. (Since Fedora release 9, kudzu is supplemented by HAL) kudzu detects and configures new and/or changed hardware on a system.

When started, kudzu detects the current hardware, and checks it against a database if one exists. It then determines if any hardware has been added or removed from the system. If so, it gives the users the opportunity to configure any added hardware, and unconfigure any removed hardware.

The kudzu program is a hardware autodetection and configuration tool that runs automatically at boot time. If you like, you can also start kudzu while Red Hat Linux is running. In either case, here is what kudzu does:

1. It checks the hardware connected to your computer.
2. It compares the hardware it finds to the database of hardware information stored in the `/etc/sysconfig/hwconf` file.
3. It prompts you to change your system configuration, based on new or removed hardware that was detected.

To run kudzu, either reboot (during the reboot, kudzu runs automatically) or switch to a virtual terminal (`Ctrl+Alt+F2`), log in as root, and run the kudzu command. For any hardware that has been added or removed since the last time kudzu was run, you are asked if you want to configure it, not configure it, or do nothing.

### **Configuring modules**

Linux comes with tools for configuring the drivers that stand between the programs you run (such as CD players and Web browsers) and the hardware they use (such as CD-ROM drives and network cards). The intention is to have the drivers your system needs most often built into the kernel; these are called resident drivers. Other drivers that are added dynamically as needed are referred to as loadable modules.

### **Finding available modules**

If you have installed the Linux kernel source code (kernel-source package), source code files for available drivers are stored in subdirectories of the `/usr/src/linux-2.4/drivers` directory. There are several ways of finding information about these drivers:

**makexconfig**— With `/usr/src/linux-2.4` as your current directory, type `make xconfig` from a Terminal window on the desktop. Select the category of module you are interested in and click Help next to the driver that interests you. The



help information that appears tells you the module name and a description of the driver.

**Documentation** — The `/usr/src/linux-2.4/Documentation` directory contains lots of plain-text files describing different aspects of the kernel and related drivers. Of particular interest is the `modules.txt` file (which describes how to work with modules) and the `Configure.help` file (which contains all the help files hardware drivers).

**kernel-doc** — The `kernel-doc` software package (available on CD #3 of the Red Hat Linux distribution) contains a large set of documents describing the kernel and drivers. These documents are stored in the `/usr/share/doc/kernel-doc*` directory.

### Listing loaded modules

To see which modules are currently loaded into the running kernel on your computer, you can use the `lsmod` command. Here's an example

```
$ lsmod
Module Size Used by
sr_mod 15120 0 (autoclean)
es1371 26784 0 (autoclean)
ac97_codec 8704 0 (autoclean) [es1371]
gameport 1920 0 (autoclean) [es1371]
soundcore 4112 4 (autoclean) [es1371]
binfmt_misc 6272 1
nuscstcp 17200 0 (unused)
```

### Loading modules

You can load any module that has been compiled and installed (to the `/lib/modules` directory) into your running kernel using the `insmod` command. The most common reasons for loading a module are that you want to use a feature temporarily (such as loading a module to support a special file system on a floppy you want to access) or to identify a module that will be used by a particular piece of hardware that could not be autodetected.

Here is an example of the `insmod` command being used to load the `parport` module. The `parport` module provides the core functions to share parallel ports with multiple devices.

```
insmodparport
```

```
Using /lib/modules/2.4.20-2.48/kernel/drivers/parport/parport.o
```

### **Removing modules**

You can remove a module from a running kernel using the `rmmod` command. For example, to remove the module `parport_pc` from the current kernel, type the following:

```
rmmodparport_pc
```

## **13.5 Installing and removing packages in Linux**

Linux provides different methods for installing software. You can install software from the standard Ubuntu software repositories using the Ubuntu Software Center, from outside of the standard Ubuntu software repositories, or by compiling source code.

If you've installed software from the Ubuntu software repositories using the Ubuntu Software Center, you can use the Ubuntu Software Center to uninstall that software as well. However, if you're more comfortable using the command line, we'll show you an easy way to see what's installed on your system and uninstall programs.

You can use the "`dpkg`" command to see a list of all installed packages on your computer, press "`Ctrl + Alt + T`" to open a Terminal window. Type the following command at the prompt and press "`Enter`" as shown in figure 13.1.

```
$ dpkg --list
```

```

lori@lori-VirtualBox: ~
lori@lori-VirtualBox:~$ dpkg --get-selections
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name Version Architecture Description
+++-+-----+-----+-----+-----+
ii account-plugin 3.12.9-1ubun amd64 Messaging account plugin for AIM
ii account-plugin 0.12+15.04.2 all GNOME Control Center account plug
ii account-plugin 0.12+15.04.2 all GNOME Control Center account plug
ii account-plugin 0.12+15.04.2 all GNOME Control Center account plug
ii account-plugin 3.12.9-1ubun amd64 Messaging account plugin for Jabb
ii account-plugin 3.12.9-1ubun amd64 Messaging account plugin for Loca
ii account-plugin 0.12+15.04.2 all GNOME Control Center account plug
ii account-plugin 3.12.9-1ubun amd64 Messaging account plugin for Yaho
ii accountsservic 0.6.37-1ubun amd64 query and manipulate user account
ii acl 2.2.52-2 amd64 Access control list utilities
ii acpi-support 0.142 amd64 scripts for handling many ACPI ev
ii acpid 1:2.0.23-1ub amd64 Advanced Configuration and Power
ii activity-log-m 0.9.7-0ubuntu amd64 blacklist configuration user inte
ii adduser 3.113+nmu3ub all add and remove users and groups
ii adium-theme-ub 0.3.4-0ubuntu all Adium message style for Ubuntu
ii adwaita-icon-t 3.14.0-2ubun all default icon theme of GNOME (smal
ii alsleriot 1:3.14.2-0ub amd64 GNOME solitaire card game collect
ii alsa-base 1.0.25+dfsg- all ALSA driver configuration files

```

Figure 13.1

Scroll through the list of installed packages in the Terminal window to find the one you want to uninstall as shown in figure 13.2. Note the full name of the package.

```

lori@lori-VirtualBox: ~
ii gedit-common 3.10.4-0ubun all official text editor of the GNOME
ii genisoimage 9:1.1.11-3ub amd64 Creates ISO-9660 CD-ROM filesyste
ii geoclue 0.12.99-4ubu amd64 Geographic information framework
ii geoclue-ubuntu 1.0.2+14.04. amd64 Provide positioning for GeoClue v
ii geoip-database 20150209-1 all IP lookup command line tools that
ii gettext 0.19.2-2ubun amd64 GNU Internationalization utilitie
ii gettext-base 0.19.2-2ubun amd64 GNU Internationalization utilitie
ii ghostscript 9.15+dfsg-0u amd64 interpreter for the PostScript la
ii ghostscript-x 9.15+dfsg-0u amd64 interpreter for the PostScript la
ii gimp 2.8.14-1ubun amd64 The GNU Image Manipulation Progra
ii gimp-data 2.8.14-1ubun all Data files for GIMP
ii gimp-help-comm 2.8.2-0ubunt all Data files for the GIMP documenta
ii gimp-help-en 2.8.2-0ubunt all Documentation for the GIMP (Engli
ii gir1.2-account 1.18+15.04.2 amd64 typelib file for libaccounts-glib
ii gir1.2-appindi 12.10.1+15.0 amd64 Typelib files for libappindicator
ii gir1.2-atk-1.0 2.14.0-1ubun amd64 ATK accessibility toolkit (Gobjec
ii gir1.2-atspi-2 2.14.0-1ubun amd64 Assistive Technology Service Prov
ii gir1.2-cbusmen 12.10.3+15.0 amd64 typelib file for libdbusmenu-glib
ii gir1.2-dee-1.0 1.2.7+15.04. amd64 GObject introspection data for th
ii gir1.2-freedes 1.42.0-2.2 amd64 Introspection data for some FreeD
ii gir1.2-gdata-0 0.16.1-1 amd64 GObject introspection data for th
ii gir1.2-gdkpixb 2.31.3-1ubun amd64 GDK Pixbuf library - GObject-Intr
ii gir1.2-glib-2. 1.42.0-2.2 amd64 Introspection data for GLib, Gobj
ii gir1.2-gmenu-3 3.10.1-0ubun amd64 GObject introspection data for th

```

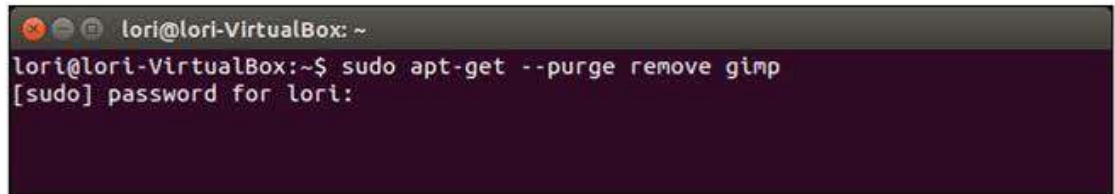
Figure 13.2

To uninstall a program, use the “apt-get” command, which is the general command for installing programs and manipulating installed programs. For example, the following command uninstalls gimp and deletes all the configuration files, using the “--purge” (there are two dashes before “purge”) command.



```
$ sudo apt-get --purge remove gimp
```

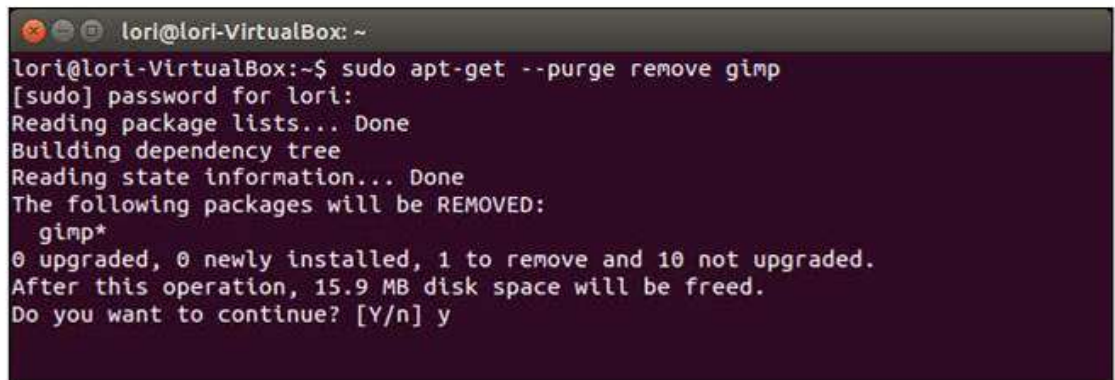
Enter your password when prompted and press “Enter”.



```
lori@lori-VirtualBox: ~
lori@lori-VirtualBox:~$ sudo apt-get --purge remove gimp
[sudo] password for lori:
```

Figure 13.3

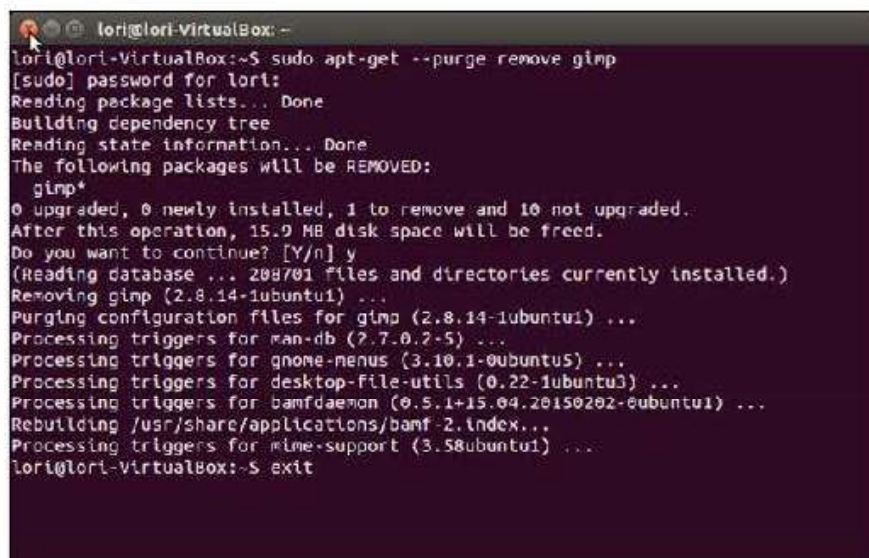
The uninstallation process begins and a summary of the actions to be taken displays. When asked if you want to continue, type a “y” and press “Enter”.



```
lori@lori-VirtualBox: ~
lori@lori-VirtualBox:~$ sudo apt-get --purge remove gimp
[sudo] password for lori:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
 gimp*
0 upgraded, 0 newly installed, 1 to remove and 10 not upgraded.
After this operation, 15.9 MB disk space will be freed.
Do you want to continue? [Y/n] y
```

Figure 13.4

The un-installation process continues. When it’s done, type “exit” at the prompt and press “Enter” to close the Terminal window, or click the “X” button in the upper-left corner of the window.



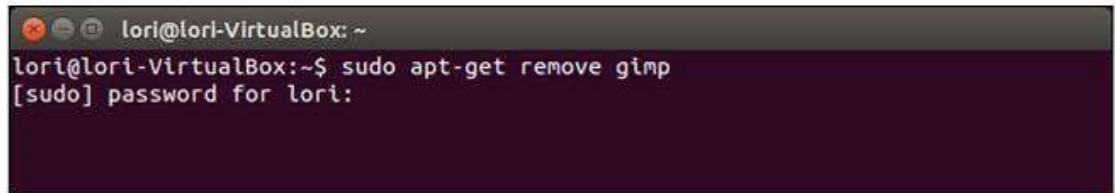
```
lori@lori-VirtualBox: ~
lori@lori-VirtualBox:~$ sudo apt-get --purge remove gimp
[sudo] password for lori:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
 gimp*
0 upgraded, 0 newly installed, 1 to remove and 10 not upgraded.
After this operation, 15.9 MB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 208701 files and directories currently installed.)
Removing gimp (2.8.14-1ubuntu1) ...
Purging configuration files for gimp (2.8.14-1ubuntu1) ...
Processing triggers for man-db (2.7.0.2-5) ...
Processing triggers for gnome-menus (3.10.1-0ubuntu5) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu3) ...
Processing triggers for bamfdaemon (0.5.1+15.04.20150202-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.58ubuntu1) ...
lori@lori-VirtualBox:~$ exit
```

Figure 13.5



If you don't want to remove the configuration files, simply leave out the "--purge" command, as shown in the following command.

```
$ sudo apt-get remove gimp
```

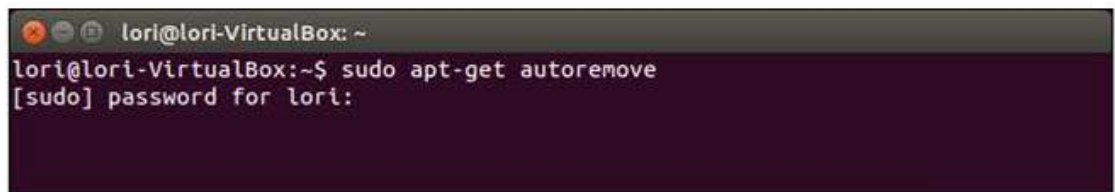


```
lori@lori-VirtualBox:~$ sudo apt-get remove gimp
[sudo] password for lori:
```

Figure 13.6

When you uninstall a program, there may be packages that the uninstalled program depended upon that are no longer used. To remove any unused packages, use the "autoremove" command, as shown in the following command.

```
$ sudo apt-get autoremove
```

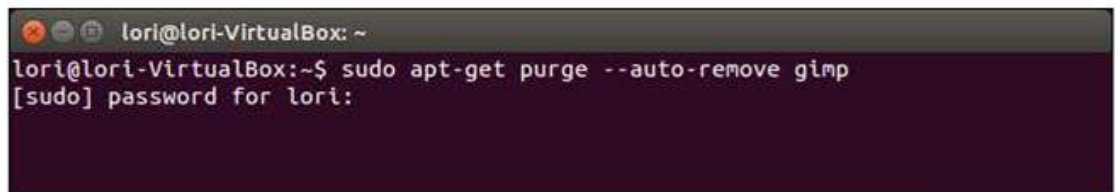


```
lori@lori-VirtualBox:~$ sudo apt-get autoremove
[sudo] password for lori:
```

Figure 13.7

You can combine the two commands for removing a program and removing dependencies that are no longer being used into one, as shown below (again, two dashes before "auto-remove").

```
$ sudo apt-get purge --auto-remove gimp
```



```
lori@lori-VirtualBox:~$ sudo apt-get purge --auto-remove gimp
[sudo] password for lori:
```

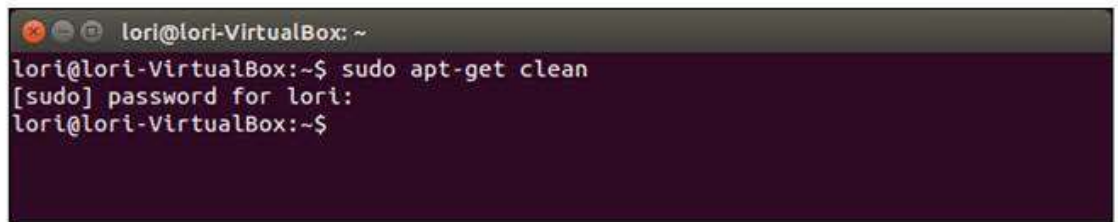
Figure 13.8

If you're short on space, you can use the "clean" command to remove downloaded archive files, as shown below.

```
$ sudo apt-get clean
```

This command removes the aptitude cache in "/var/cache/apt/archives". When you install a program, the package file is downloaded and stored in that

directory. You don't need to keep the files in that directory. However, the only drawback of deleting them, is that if you decide to install any of those programs again, the packages would have to be downloaded again.



```
lori@lori-VirtualBox: ~
lori@lori-VirtualBox:~$ sudo apt-get clean
[sudo] password for lori:
lori@lori-VirtualBox:~$
```

Figure 13.9

The “apt-get” is a handy tool that makes downloading, installing, and uninstalling programs quick and easy. For more information about using the “apt-get” command, type “apt-get” at the prompt and press “Enter”.

### 13.6 Self Learning Exercise

- Q.1 Uname command gives information about
- a) Users
  - b) Files
  - c) System
  - d) I/O device
- Q.2 Which utility writes files to and retrieves files from an archive
- a) cpio
  - b) tar
  - c) whoami
  - d) users
- Q.3 Kudzu is a hardware probing program (written by Red Hat Linux) which
- a) detects changes in the running system's hardware configuration, if any, and activates the newly detected hardware (or removal of hardware).
  - b) detects changes in operating system and activates the newly changes, if any.
  - c) detects changes in shell and activates the newly changes, if any.
  - d) detects changes in kernel and activates the newly changes, if any.

- Q.4 The “dpkg” command to see
- a) A list of all installed softwares on computer.
  - b) A list of all installed packages on computer.
  - c) A list of all installed commands on computer.
  - d) A list of all installed utilities on computer.

### **13.7 Summary**

Linux derives in many zests and it is modified by an amount of commands to perform in the way they wanted. System information change to from various commands like uname, hostname and disk partitions commands.

In Linux, user has facility to proceed for the backup and restore the files using associated commands.

Diverse packages install and uninstall in Linux using correlated commands.

Kudzu is a hardware configuration tool. It originates from Red Hat, now developed as part of the Fedora project, but can and is also used in other distributions. Kudzu probes for new hardware and attempts to configure it. It checks a database and checks if any new hardware has been added or old hardware removed. If any changes are found, it then gives the user the opportunity to configure this hardware.

### **13.8 Glossary**

Kudzu-detects new and changed hardware by comparing existing hardware against a database in /etc/sysconfig/hwconfig. Users are then given the opportunity to configure the hardware. Runs automatically at boot time

### **13.9 Answers to Self-Learning Exercise**

- Q.1 (d)  
Q.2 (a) and (b)  
Q.3 (a)  
Q. 4 (b)

## 13.10 Exercise

- Q.1 Which utility displays the name of the system you are working on
- a) youname
  - b) hostname
  - c) ghostname
  - d) username
- Q.2 Which command gives list of all existing disk partitions on your system?
- a) fdisk -a
  - b) fdisk -w
  - c) fdisk -l
  - d) fdisk -b
- Q.3 Which is true about cpio?
- a) The cpio (copy in/out) program is similar to tar but can read and write archive file in various formats, including the one used by tar.
  - b) The cpio (copy in/out) program is not similar to tar but can read and write archive file in various formats, including the one used by tar.
  - c) The cpio (copy in/out) program is similar to tar but can read and write archive file in only one format, including the one used by tar.
  - d) The cpio (copy in/out) program is similar to tar but can only read the archive file in various formats, including the one used by tar.
- Q.4 The restore utility can
- a) restore an entire filesystem, a directory hierarchy, or an individual file.
  - b) restore only file system
  - c) restore only individual file
  - d) restore only directory hierarchy
- Q.5 tar command with option -r
- a) create files in archive
  - b) delete files in archive
  - c) append files in archive
  - d) extract files in archive



Q.6 Which command remove a module from a running kernel

- a) removemode command
- b) rmmmod command
- c) rmmode command
- d) removemod command

Q.7 tar command with option -u

- a) create files in archive
- b) delete files in archive
- c) append files in archive
- d) update files in archive

### 13.11 Answers of Exercise

Q. 1 (b)

Q. 5 (c)

Q. 2 (c)

Q. 6 (b)

Q. 3 (a)

Q. 7 (d)

Q. 4 (a)

### References and Suggested Readings

1. A Practical Guide to Ubuntu Linux by Mark G. Sobell, Prentice Hall, 2007.
2. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
3. UNIX Shell Programming by Yashwant Kanetkar, BPB Publications, 2003.

# UNIT-14

## Introduction to X-Windows System

### Structure of the Unit

- 14.0 Objective
- 14.1 Introduction
- 14.2 Configure X-windows starting and using X desktop
- 14.3 KDE (K Desktop Environment)
- 14.4 GNOME graphical interfaces
- 14.5 Changing X windows settings
- 14.6 Self Learning Exercise
- 14.7 Summary
- 14.8 Glossary
- 14.9 Answers to Self Learning Exercise
- 14.10 Exercise
- 14.11 Answers to Exercise

### 14.0 Objective

In this chapter we shall focus upon the following topics

- Configuration of X-windows
- X-desktop
- KDE (K Desktop Environment)
- GNOME graphical interfaces
- Changing X windows settings

## 14.1 Introduction

The X Window System (commonly referred to as X or X11) is a network-transparent graphical windowing system based on a client/server model. Primarily used on UNIX and Unix-like systems such as Linux, versions of X are also available for many other operating systems. Although it was developed in 1984, X is not only still viable but also is in fact the standard environment for UNIX windowing systems.

The X Window System is a graphical windowing system that was developed at MIT in 1984. X was developed as part of Project Athena, a cooperative effort by MIT, IBM, and Digital Equipment Corporation to develop a network of heterogeneous engineering terminals that could be used for teaching purposes. On that time, several graphic displays were used for working and each having its own method for drawing. That would be absurd, so they reasoned that a separate program should handle the display. This led to a very important splitting of the application into two components- *client and server*.

Client-server computing is the division of labor between two components connected in a network. An FTP application is split up into a server and client components- two separate programs by themselves. The database server acts as a repository of data, while the front-end application resides in the user's own workstation. The job of handling the PC's display is handled by client application itself.

X reverses this paradigm, and its architecture places the responsibility of handling the display on the server, while the application itself is run as a client. X programs are thus quite portable, being free of the complexities involving the handling the terminal. To run an X client on a different terminal, all one has to do is to write the server component of X for that terminal. X server program must be able to handle not only the application's output, but its input as well.

X clients display their output on separate windows on the display. The X server itself doesn't provide any window management function at all. The specific look and feel of X programs is determined not by the X- server, but by the GUI, which is separate product by itself, known as window manager. The window manager is built on the X client programs, and ensures that all clients have the same look and feel irrespective of the vendor they come from.



X was primarily designed to run over a network. X applications use the protocol, which sits atop the standard networking protocols that are used by machines—like TCP/IP. The transmission is totally reliable; X uses the TCP protocol.

## **14.2 Configure X-windows, starting and using X desktop**

There are different ways to configure X-windows on UNIX and Linux types. Different flavors of Linux like Redhat, Fedora etc, have different procedure to configure X-windows.

### **Configure X-windows on UNIX**

There are basically two ways of starting X. A common way is to use the *startx* command resident in */usr/bin*. The command *startx* the X server and loads a number of icons onto the desktop. These icons feature a number of common services found in all UNIX systems, like mail and system administration. Unless you are using these services, further work from here can commence by invoking the *xterm* client, the most often client of the X windows system.

### **Configure X-windows on Redhat Linux**

While the heart of Red Hat Enterprise Linux is the kernel, for many users, the face of the operating system is the graphical environment provided by the X Window System, also called X.

The graphical environment for Red Hat Enterprise Linux is supplied by the X.Org Foundation, an open source consortium created to manage development and strategy for the X Window System and related technologies. X.Org is a large scale, rapidly developing project with hundreds of developers around the world. It features a wide degree of support for a variety of hardware devices and architectures, and can run on a variety of different operating systems and platforms. This release for Red Hat Enterprise Linux specifically includes the X11R6.8 release of the X Window System.

The X Window System uses a client-server architecture. The X server (the Xorg binary) listens for connections from X client applications via a network or local loopback interface. The server communicates with the hardware, such as the video card, monitor, keyboard, and mouse. X client applications exist in the



user-space, creating a graphical user interface (GUI) for the user and passing user requests to the X server.

Red Hat Enterprise Linux 4.5.0 uses the X11R6.8 release as the base X Window System, which includes many cutting edge X.Org technology enhancements, such as 3D hardware acceleration support, the XRender extension for anti-aliased fonts, a modular driver-based design, and support for modern video hardware and input devices.

The files related to the X11R6.8 release reside primarily in two locations:

`/usr/X11R6/` Contains X server and some client applications, as well as X header files, libraries, modules, and documentation.

`/etc/X11/` Contains configuration files for X client and server applications. This includes configuration files for the X server itself, the fs font server, the X display managers, and many other base components.

It is important to note that the configuration file for the newer Fontconfig-based font architecture is `/etc/fonts/fonts.conf` (which obsoletes the `/etc/X11/XftConfig` file).

Because the X server performs advanced tasks on a wide array of hardware, it requires detailed configuration. The installation program installs and configures X automatically, unless the X11R6.8 release packages are not selected for installation. However, if the monitor or video card changes, X must be reconfigured. The best way to do this is to use the X Configuration Tool (`system-config-display`).

To start the X Configuration Tool while in an active X session, go to the Main Menu Button (on the Panel) => System Settings => Display. After using the X Configuration Tool during an X session, changes takes effect after logging out and logging back in.

### **Configure X-windows on Fedora**

Fedora 12 uses the X11R7.1 release as the base X Window System, which includes several video driver, EXA, and platform support enhancements over the previous release, among others. In addition, this release also includes several automatic configuration features for the X server.

X11R7.1 is the first release to take specific advantage of the modularization of the X Window System. This modularization, which splits X into logically distinct modules, makes it easier for open source developers to contribute code to the system.

In the X11R7.1 release, all libraries, headers, and binaries now live under `/usr/` instead of `/usr/X11R6`. The `/etc/X11/` directory contains configuration files for X client and server applications. This includes configuration files for the X server itself, the xfs font server, the X display managers, and many other base components.

The configuration file for the newer Fontconfig-based font architecture is still `/etc/fonts/fonts.conf`. For more on configuring and adding fonts, refer to Section 20.4, “Fonts”.

Because the X server performs advanced tasks on a wide array of hardware, it requires detailed information about the hardware it works on. The X server automatically detects some of this information; other details must be configured.

In Fedora's default graphical environment, the X Configuration Tool is available at System (on the panel) > Administration > Display.

Changes made with the X Configuration Tool take effect after logging out and logging back in.

### **Configure X-windows using X Desktop Manager**

If you wish, you may use the X Desktop Manager (“xdm”) to start up the X Window System automatically at system boot time. This allows your Linux system to always run under X (although you 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 you and your 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. You may also wish to check your `"/etc/inittab"` file, probably near the bottom, to ensure the following line is present:

```
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

If you have enabled xdm and wish to use a higher `"bpp"` value than the default of 8 (and your video card and monitor will support it), you 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.

You 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 environment
```

### 14.3 KDE (K Desktop Environment)

KDE, the K Desktop Environment, is an open source graphical desktop environment designed to provide a convenient, consistent, and user-friendly working environment for everyone from beginners to power users. All of the features of X are available, including the traditional xterm command-line interface for those who prefer it, but the graphical interface both unifies and simplifies working with X. Figure 14-1 shows the KDE logo; as you work with KDE, you will become familiar with this logo as the entry point into the KDE menu system.



Figure 14.1

KDE provides a unified desktop that uses drag-and-drop technology. You drag a file icon to an application icon to start the application or to move it to a folder. Or you can click on a file icon to open it in the appropriate application. You can put folders or files on the desktop for easy access, as well as links to programs



that you frequently run—you can then start the program by clicking on the icon. Similarly, you can put a link to a device, such as your floppy drive, or a CD or DVD drive, on the desktop.

In addition, KDE provides a web browser, Konqueror, that also serves as the KDE filemanager, and provides the underlying technology for both the help facility and the graphical configuration tool known as the Control Center. A tool bar known as the panel sits on the screen, usually along the bottom, and provides a focal point for managing your desktop and running programs.

### The Desktop

A typical KDE desktop is shown in Figure 14-2. The desktop fills your screen and provides the area where you do your work. You can have multiple virtual desktops and move between them. By default, KDE provides four virtual desktops, but you can change that number at any time via the Control Center.



Figure 14.2

The desktop shown here has two open windows—a shell and a web browser. Along the left edge of the desktop are icons that provide a trash barrel for deleted files, a link to the user's home directory, and links to the CD and floppy drives. The bar along the bottom edge of the desktop is called the panel; it provides a launch pad for applications and ready access to your virtual desktops and running programs.

### Managing the Desktop

The default desktop is just a starting point. As you work with KDE, you'll want to rearrange the desktop, add and delete files and links, and generally customize it to suit your needs. KDE provides the desktop menu for this purpose. Right-



clicking anywhere on the desktop displays the desktop menu, shown in Figure 14.3.



**Figure 14.3**

When you select **Create New**, a submenu opens that lets you choose between creating a new directory, **HTML** file, text file, **CDROM** device, floppy device, application, or Internet address (**URL**). Whichever type you select, a dialog box opens for you to specify the object. For example, for a text file, you specify a filename; for a **URL**, you specify the **URL**; for a device, you indicate which device you are linking to; etc. An appropriate icon is placed on the desktop for the newly created object, and a file is stored in the `~/Desktop` directory under the name you gave the object.

You can undo the creation of the object by selecting the **Undo** option. Note that in the figure, this option is grayed out, indicating that it isn't available, because there was no previous action to undo. The **Paste** option allows you to paste an object from the clipboard to the desktop.

Selecting **Bookmarks** opens a submenu that lets you edit your Konqueror bookmarks or modify Netscape bookmarks. Selecting **Edit Bookmarks** from the submenu runs the Konqueror bookmarks editor, `keditbookmarks`. Once you are in the editor window, you can edit your existing bookmarks or import your Mozilla and/or Netscape bookmarks to save them as Konqueror bookmarks. When you are done with editing, you can export the updated bookmarks to

Mozilla and Netscape. If you select Netscape bookmarks from the editor submenu and click on a bookmark, KDE brings up Konqueror and goes to that page.

The Run Command option lets you run a single command without having to open a terminal window. Entering a URL instead of a command brings up a Konqueror window and goes to that URL. Running a command line this way doesn't show you any output, so you wouldn't want to use it, for example, to run an ls command. On the other hand, it would be useful for running chmod to change the permissions on a file. Using Run Command is equivalent to entering the keyboard shortcut Alt-F2. Run Command is also available on the K menu.

### **The KDE Panel**

The KDE panel, known as Kicker, is a toolbar used for launching applications. It consists of menus, icons that run programs when clicked, and small programs known as applets that run inside the panel.



**Figure 14.4**

Figure 14.4 shows a typical panel. It includes the following items, from left to right:

#### **K menu**

KDE's K logo marks the Application Starter, or K menu. Clicking on the K menu displays a hierarchical list of available applications. As you add new applications to your system, you can update the K menu by running `kappfinder` from a command line or selecting System->KAppfinder from the K menu.

#### **Show Desktop**

Clicking on the Show Desktop button iconifies all your open windows into the taskbar section of your panel. Your desktop is now shown, empty of open windows.

#### **Terminal**

The button that looks like a terminal with a shell in front of it opens a Konsole (KDEconsole) terminal emulation window.

#### **Control Center**

The button that looks like a terminal partially covered by a circuit board opens the KDE control center. The control center is the central point for configuring your desktop environment.

## **Help**

The KDE Help center is represented by a button that looks like a life-saving ring. Click on the icon to bring up the KDE help system.

## **Home**

Clicking on the button with a small house in front of a file folder opens the Konqueror file manager, with your home directory and its files displayed as icons.

## **Konqueror**

The next button represents the web browser aspect of Konqueror.

## **Text editor**

KDE includes several text editors, described briefly later in the section “Editors:kate, kedit, and kwrite”. The pen-and-paper icon is used for both kate and kwrite; in

this case, it represents a link to kate.

## **Desktop preview**

The desktop preview, or mini-pager appears next, consisting of a set of buttons that let you switch between virtual desktops. There is one button in the mini-pager for each virtual desktop. The highlighted button denotes the active desktop.

## **Taskbar**

The taskbar contains an entry for every running program. Clicking on one of these names calls up the program window, together with the desktop that it is in. This makes it easy to switch to any application without having to switch first to another desktop. If the window is currently iconified, it appears grayed out in the taskbar, and clicking on its entry restores the full window. If the window is not iconified, clicking on it moves the focus to that window (automatically switching desktops as necessary), and raises the window if it is currently hidden.

## **Lock/logout applet**

The lock and logout buttons to the right of the taskbar are actually part of a single applet that provides both screen locking and logout functions. Clicking one of these buttons is the same as selecting the equivalent option from the desktop menu, described earlier in the section “Managing Your Desktop”.

## **System tray**

Rounding up the panel is the system tray, where swallowed applications reside. Swallowed applications are mini-applications that run on the panel instead



of in a window on the desktop. Two examples shown here are the clock, which contains an embedded calendar, and klipper, KDE's clipboard tool.

### Hide panel

Clicking on the right-pointing arrow on the far right of the panel "rolls up" the whole panel to the right, to provide more desktop work area. This action leaves only an narrow vertical bar showing, with the arrow now pointing left. Clicking on this arrow brings the panel back into view again.

### Konqueror

One of KDE's most important and visible tools is Konqueror, the KDE web browser, advanced file manager, and universal viewer--all rolled into one

Figure 14-5 shows some of the features of Konqueror, including both its web browsing and file management capabilities. The figure also shows how you can split the Konqueror window into parts, each performing a different activity.



Figure 14.5

The largest window is being used for browsing; in this case, it is displaying the KDE home page ([www.kde.org](http://www.kde.org)). The two smaller windows show two views of Konqueror as file manager. The following sections describe the browser and file manager aspects of Konqueror in more detail, and then explain how to customize Konqueror.

Once Konqueror is running, it automatically switches between the browser and file manager modes as needed. But you can also open Konqueror in one mode or the other.

To open Konqueror in browser mode, do one of the following:



- Click on the Konqueror icon in the Panel.
- From the K menu, select Internet Konqueror.
- Enter a URL on the command line.

Or to open it in file manager mode, do one of the following:

- Click on the house icon in the Panel.
- Click on any directory icon on the desktop (including the trash barrel) to open Konqueror and display the contents of that directory.
- Enter the command `konqueror` on the command line.

### **The Window Manager**

Unlike GNOME, KDE has its own window manager, `kwin`, which is tightly integrated into KDE itself. Thus you won't find specific options or settings for the window manager in the Control Center—it's not generally treated as a separate entity. Managing Windows applies to the KDE window manager as well. In addition, this section contains a few tips for managing your windows. Windows can be pinned, so that they appear at the same location on every desktop. For example, you might pin the `kscd` CD player to turn down the volume when somebody walks into your office. To pin a window, click on the pin symbol on the left-hand side of the titlebar. To move a window to another desktop, access the Window menu by right-clicking on the window's title bar. Then choose the desired desktop from the To Desktop submenu. The window disappears from the current desktop. To access it again, simply go to the new desktop. Another method of moving a window is to pin down the window by clicking on the pin button. This puts the window on all the desktops. Then change to the desktop you want and pull out the pin. The window now appears only on the new desktop, and you can work with it immediately.

### **The Help Center**

The KDE Help Center provides a central point for accessing not only KDE help but also the man pages and Info pages for your Linux system. The Help Center's welcome page can be seen in Figure 14-6. On the left-hand side of the Help Center window are two tabs. In Figure 14-6, the Contentstab is displayed. Clicking on an entry with a page icon displays that entry; clicking on an entry with a book icon opens a hierarchical list of contents for that entry. For example, clicking on the Application manuals entry lists the hierarchy of

manuals for various KDE applications, organized into categories that match those of the K menu. Click on an entry to display the documentation for that KDE feature. The second tab is the Glossary, which provides a glossary of terms. You can access the glossary either alphabetically or by topic.

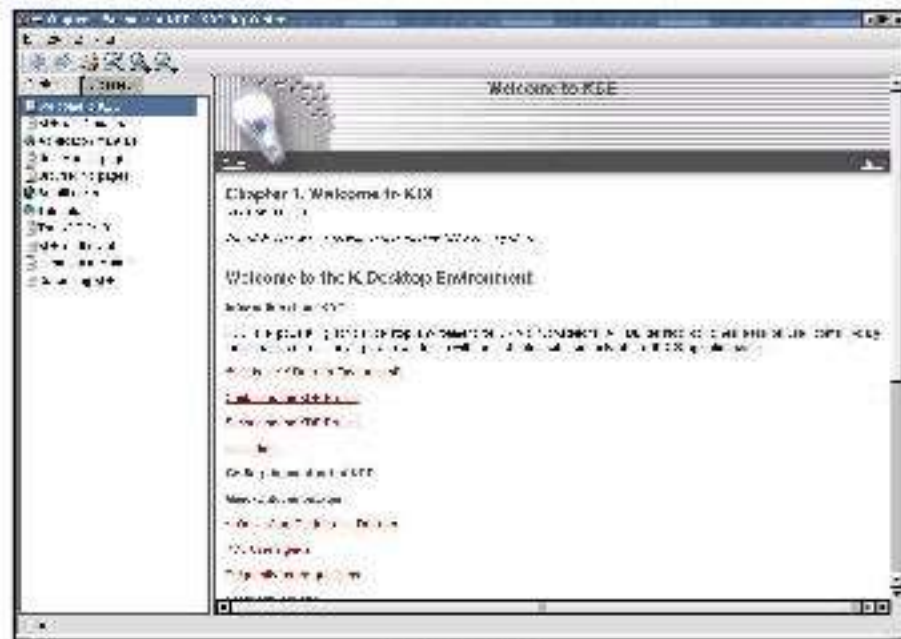


Figure 14.6

### The Control Center

The KDE Control Center, `kcontrol`, contains a number of configuration tools, called modules that allow you to configure and view information about your system. You can configure the desktop, window manager, input devices, and other important parts of your system here. The Control Center is split into two windows: the left window shows a hierarchical list of installed modules, and the right window displays the selected module. As is often the case, there is more than one way to customize KDE. In this section, we concentrate on the KDE Control Center. However, you can choose individual modules from the K menu's Preference option. In addition, many KDE applications have entry points into the appropriate configuration module, as we've seen in some of the earlier sections. `kcontrol` comfortably brings everything together in a central place.

To launch the KDE control center, either run the command `kcontrol` or select the ControlCenter option from the K menu. The Control Center appears, as shown in Figure 14-7.



Figure 14.7

The Control Center has three tabs on the left-hand side: **Index**, **Search**, and **Help**. Most of the time you will probably use the **Index**, shown in Figure 14-7, to access the module you want.

If you don't know what module you want, select **Search** and either enter a keyword or scroll through the alphabetical list of keywords and select the one you want. The results box at the bottom of the **Search** tab lists the module or modules containing configuration options for that keyword. Click on a module name and the main Control Center window opens to that module.

The **Help** tab provides information to help set configuration options in the currently displayed module. Select **Help** from the menu bar to get the documentation for the Control Center itself.

## 14.4 GNOME Graphical Interfaces

**GNOME** is the default desktop environment for Ubuntu Linux. It provides a simple, coherent user interface that is suitable for corporate use. **GNOME** uses **GTK** for drawing widgets. **GTK**, developed for the GNU Image Manipulation Program (**gimp**), is written in **C**, although bindings for **C++** and other languages are available. **GNOME** does not take much advantage of its component

architecture. Instead, it continues to support the traditional UNIX philosophy of relying on many small programs, each of which is good at doing a specific task. GNOME comes with numerous utilities that can make your work with the desktop easier and more productive.

### Deskbar Applet

Clicking the Deskbar applet (Figure 14-8) or pressing ALT-F3 opens the Deskbar Applet window (also in Figure 14-8). As you type in the text box labeled Search, this tool searches for the string you are entering. In Figure 14-8, the user has entered the string desk. Below the list box labeled History, the Deskbar Applet window displays matches it has found for desk. At the top of the list are actions that match or whose descriptions match the string.

For example, clicking Launch: Take Screenshot displays a window that includes a radio button labeled Grab the whole desktop. Below the actions are places: When you click Desktop, the Desktop Applet opens the Nautilus File Browser displaying the desktop. Finally, the window displays a list of Web searches. Click one of these to open Firefox and perform the search.

You can also use the Search for Files window to search for files.

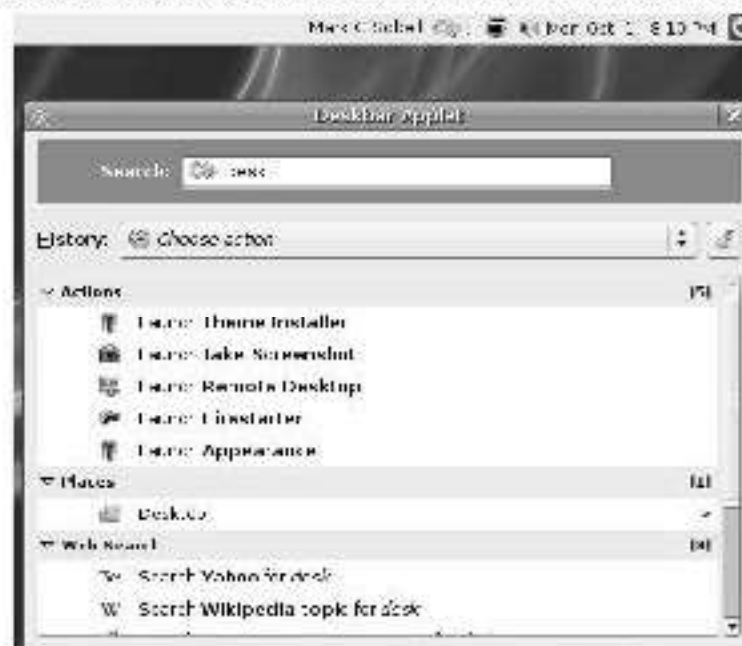


Figure 14.8

The Deskbar Applet displays matches based on extensions. Right-click the Deskbar applet and select Preferences to open the Deskbar Preferences window. In the Searches tab of this window, you can select the extensions you want the



applet to use and change the order in which it presents information generated by the extensions. For example, remove the tick from the check box labeled Programs to cause the Deskbar Applet window not to display programs in the Action section of its window.

Experiment with enabling and disabling extensions and changing their order. The Extensions with Errors tab lists extensions the Deskbar applet cannot use. When you highlight one of the lines in the Extensions with Errors frame, the Desktop Preferences window displays the reason the extension cannot be used.

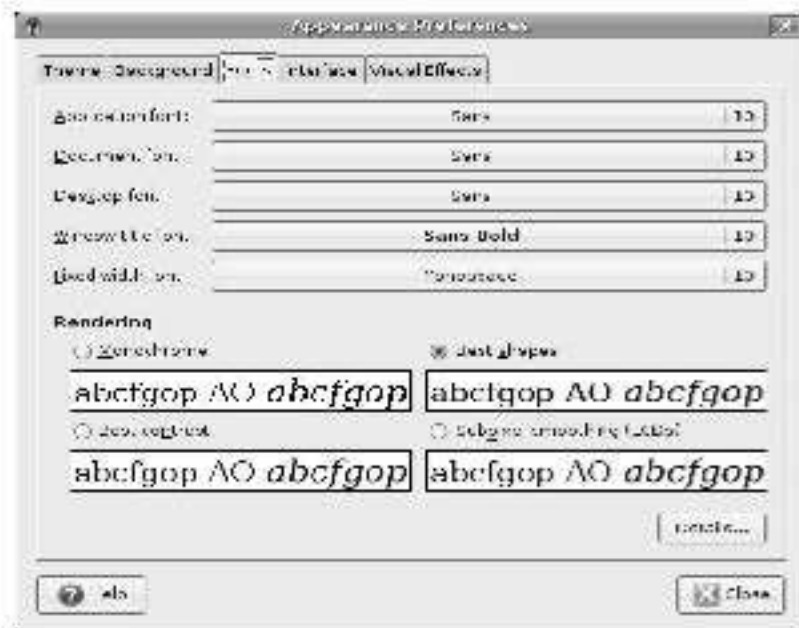


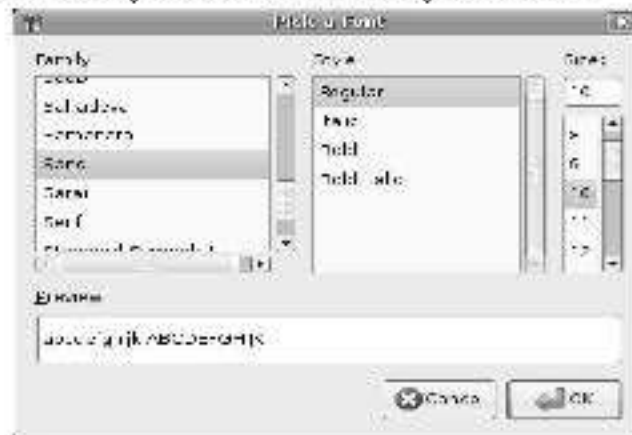
Figure 14.9

### Font Preferences

The Fonts tab of the Appearance Preferences window (Figure 14-9) enables you to change the font that GNOME uses for applications, documents, the desktop, window titles, and terminal emulators (fixed width). To display this window, select Main menu: System → Preferences → Appearance or enter `gnome-appearance-properties` on a command line. Click the Fonts tab. Click one of the five font bars in the upper part of the window to display the Pick a Font window. Examine the four sample boxes in the lower part of the window and select the one in which the letters look the best. Subpixel smoothing is usually best for LCD monitors. Click Details to refine the font rendering further, again picking the box in each frame in which the letters look the best.

### Pick a Font Window

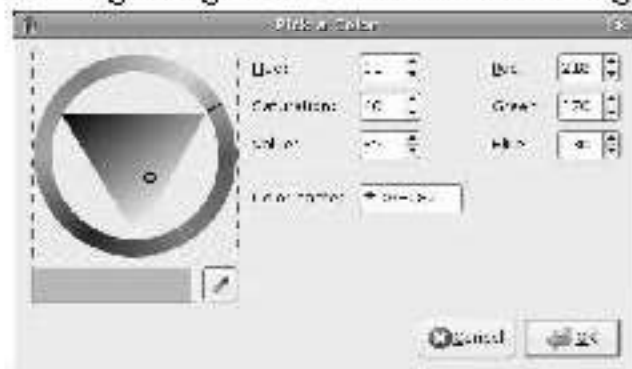
The Pick a Font window (Figure 14-10) appears when you need to choose a font. From this window you can select a fontfamily, a style, and a size. A preview of your choice appears in the Preview box inthe lower part of the window. Click OK when you are satisfied with your choice.



**Figure 14.10**

### Pick a Color Window

The Pick a Color window (Figure 14-11) appears when you need to specify a color, such as when you specify a solid color for the desktop background or a panel. To specify a color for a panel, right-click the panel to display its context menu, click the Background tab, click the radio button labeled Solid color, and click within the box labeled Color. GNOME displays the Pick a Color window. When the Pick a Color window opens, the bar below the color circle displays the current color. Click the desired color on the color ring, and click/drag the lightness of that color in the triangle.



**Figure 14.11**

As you change the color, the right end of the bar below the color circle previews the color you are selecting, while the left end continues to display the current color. You can also use the eyedropper to pick up a color from the workspace.

Click the eyedropper, and then click the resulting eyedropper mousepointer on the color you want to select. The color you choose appears in the bar. Click OK when you are satisfied with the color you have specified.

### Run Application Window

The Run Application window (Figure 14-12) enables you to run a program as though you had initiated it from a command line. To display the Run Application window, press Alt-F2. Enter a command in the text box. As soon as GNOME can uniquely identify the command you are entering, it completes the command and may display an object that identifies the application. Keep typing if the displayed command is not the one you want to run. Otherwise, press RETURN to run the command or TAB to accept the command in the text box. You can then continue entering information in the window. Click Run with file to specify a file to use as an argument to the command in the text box. Put a tick in the check box labeled Run in terminal to run a textual application, such as vim, in a terminal emulator window.



Figure 14.12

### Searching for Files

The Search for Files window (Figure 14-13) can help you find files whose locations or names you do not know or have forgotten. You can also use the Deskbar applet to search for files. Open this window by selecting Main menu: Places → Search for Files or enter `gnome-search-tool` on a command line from a terminal emulator or Run Application window (ALT-F2). To search by filename or partial filename, enter the (partial) filename in the combo box labeled Name contains and then select the folder you want to search in from the drop-down list labeled Look in folder. When GNOME searches in a folder, it searches subfolders to any level (it searches the directory hierarchy).

To search all directories in all mounted filesystems, select **File System** from the drop-down list labeled **Look in folder**. Select **Other** to search a folder not included in the drop-down list; GNOME opens the **Browse** window. Once you have entered the search criteria, click **Find**. GNOME displays the list of files matching the criteria in the list box labeled **Search results**. Double-click a file in this list box to open it. You can refine the search by entering more search criteria. Click the triangle to the left of **Select more options** to expand the window and display more search criteria. GNOME initially displays two search criteria and a line for adding criteria as shown in Figure 14-13. With this part of the window expanded, GNOME incorporates all visible search criteria when you click **Find**.



Figure 14.13

## 14.5 Changing X-Windows Settings

The X Window System was inspired by the ideas and features found in earlier proprietary window systems but are written to be portable and flexible. X is designed to run on a workstation, typically attached to a LAN. The designers built X with the network in mind. If you can communicate with a remote computer over a network, running an X application on that computer and sending the results to a local display is straightforward.

Although the X protocol has remained stable for a long time, additions to it in the form of extensions are quite common. One of the most interesting—albeit one that has not yet made its way into production—is the Media Application Server, which aims to provide the same level of network transparency for sound and video that X does for simple windowing applications.

### Starting X from a Character-Based Display



Once you have logged in on a virtual console, you can start an X WindowSystem server by using `startx`. For information on creating a `/etc/inittab` file that causes Linux to boot into recovery (single-user) mode, where it displays a textual interface. When you run `startx`, the X server displays an X screen, using the first available virtual console. The following command causes `startx` to run in the background so you can switch back to this virtual console and give other commands:

```
$ startx&
```

### **Remote Computing and Local Displays**

Typically the X server and the X client run on the same machine. To identify remote X server (display) an X application (client) is to use, you can either set a global shell variable or use a command line option. Before you can connect to a remote X server, you must turn off two security features: You must run `xhost` on the server to give the client permission to connect to the X server and you must turn off the `X-nolistentcp` option on the server.

#### **The X -nolistentcp Option**

As Ubuntu is installed, the X server starts with the `-nolistentcp` option, which protects the X server by preventing TCP connections to the X server. To connect to a remote X server, you must turn this option off on the server. To turn it off, select

Main menu: System → Administration → Login Window, Security tab, and remove

the tick from the check box labeled Deny TCP connections to Xserver.

#### **xhost Grants Access to a Display**

As Ubuntu is installed, `xhost` protects each user's X server. A user who wants to grant access to his X server needs to run `xhost`. Assume Max is logged in on the system named `tiny` and wants to allow a user on `dog` to use his display (X server). Max runs the following command:

```
max@tiny:~$ xhost +dog
```

dog being added to access control list

```
max@tiny:~$ xhost
```

access control enabled, only authorized clients can connect

```
INET:dog
```

Without any arguments, `xhost` describes its state. In the preceding example, `INET`

indicates an IPv4 connection. If Max wants to allow all systems to access his display, he can give the following command:

```
$ xhost +
```

access control disabled, clients can connect from any host

If you frequently work with other users via a network, you may find it convenient to add an `xhost` line to your `.bash_profile` file, but see the adjacent tip regarding security and `xhost`. Be selective in granting access to your X display with `xhost`, however; if another system has access to your display, you may find your work frequently interrupted.

### **The DISPLAY Variable**

The most common method of identifying a display is to use the `DISPLAY` shell environment variable to hold the X server ID string. This locally unique identification string is automatically set up when the X server starts. The `DISPLAY` variable holds the screen number of a display:

```
$ echo $DISPLAY
```

```
:0.0
```

The format of the complete (globally unique) ID string for a display is

```
[hostname]: display-number[.screen-number]
```

where `hostname` is the name of the system running the X server, `display-number` is the number of the logical (physical) display (0 unless multiple monitors or graphical terminals are attached to the system, or if you are running X over ssh), and `screen-number` is the logical number of the (virtual) terminal (0 unless you are running multiple instances of X). When you are working with a single physical screen, you can shorten the identification string. For example, you can use `tiny:0.0` or `tiny:0` to identify the only physical display on the system named `tiny`. When the X server and the X clients are running on the same

system, you can shorten this identification string even further to `:0.0` or `:0`. An `ssh` connection shows `DISPLAY` as `localhost:10.0`.

### **Running Multiple X Servers**

You can run multiple X servers on a single system. The most common reason for running a second X server is to use a second display that allocates a different number of bits to each screen pixel (uses a different color depth). The possible values are 8, 16, 24, and 32 bits per pixel. Most X servers available for Linux default to 24 or 32 bits per pixel, permitting the use of millions of colors simultaneously. Starting an X server with 8 bits per pixel permits the use of any combination of 256 colors at the same time. The maximum number of bits per pixel allowed depends on the computer graphics hardware and X server. With fewer bits per pixel, the system has to transfer less data, possibly making it more responsive. In addition, many games work with only 256 colors.

When you start multiple X servers, each must have a different ID string. The following command starts a second X server:

```
$ startx — :1
```

The `—` option marks the end of the `startx` options and arguments. The `startx` script uses the arguments to the left of this option and passes arguments to the right of this option to the X server. When you give the preceding command in a graphical environment, such as from a terminal emulator, you must work with root privileges; you will initiate a privileged X session. The following command starts an X server running at 16 bits per pixel:

```
$ startx -- -depth 16 &
```

### **Stopping the X Server**

How you terminate a window manager depends on which window manager you are running and how it is configured. If X stops responding, switch to a virtual terminal, login from another terminal or a remote system, or use `ssh` to access the system. Then kill the process running X. You can also press `CONTROL-ALT-BACKSPACE` to quit the X server. This method may not shut down the X session cleanly; use it only as a last resort.

### **Remapping Mouse Buttons**

Each description of a mouse click refers to the button by its position (left, middle, or right, with left implied when no button is specified) because the position of a mouse button is more intuitive than an arbitrary name or number. X numbers buttons starting at the left and continuing with the mousewheel. The buttons on a three-button mouse are numbered 1 (left), 2 (middle), and 3 (right). A mouse wheel, if present, is numbered 4 (rolling it up) and 5 (rolling it down). Clicking the wheel is equivalent to clicking the middle mouse button. The buttons on a two-button mouse are 1 (left) and 2 (right). If you are right-handed, you can conveniently press the left mouse button with your index finger; X programs take advantage of this fact by relying on button 1 for the most common operations. If you are left-handed, your index finger rests most conveniently on button 2 or 3 (the right button on a two- or three-button mouse).

“Mouse Preferences” describes how to use a GUI to change a mouse between right-handed and left-handed. You can also change how X interprets the mouse buttons using `xmodmap`. If you are left-handed and using a three-button mouse with a wheel, the following command causes X to interpret the right button as button 1 and the left button as button 3:

```
$ xmodmap -e 'pointer = 3 2 1 4 5'
```

Omit the 4 and 5 if the mouse does not have a wheel. The following command works for a two-button mouse without a wheel:

```
$ xmodmap -e 'pointer = 2 1'
```

If `xmodmap` displays a message complaining about the number of buttons, use the `xmodmap -pp` option to display the number of buttons X has defined for the mouse:

```
$ xmodmap -pp
```

There are 9 pointer buttons defined.

| Physical Button | Button Code |
|-----------------|-------------|
| 1               | 1           |
| 2               | 2           |
| 3               | 3           |
| 4               | 4           |



|   |   |
|---|---|
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |

Then expand the previous command, adding numbers to complete the list. If the `-pp` option shows nine buttons, give the following command:

```
$ xmodmap -e 'pointer = 3 2 1 4 5 6 7 8 9'
```

## 14.6 Self Learning Exercise

- Q.1 X-windows system is
- a) Networking tool
  - b) Graphical User Interface
  - c) Command line
  - d) Shell
- Q.2 KDE is a
- a) Kernel
  - b) Shell
  - c) Graphical User Interface tool
  - d) Process.
- Q.3 GNOME
- a) Thread
  - b) Job
  - c) Graphical User Interface tool
  - d) File manager

## 14.7 Summary

The X Window System (X11, or shortened to simply X) is a windowing system for bitmap displays, common on UNIX-like computer operating systems.

X provides the basic framework for a GUI environment: drawing and moving windows on the display device and interacting with a mouse and keyboard. X does not mandate the user interface – this is handled by individual programs.

As such, the visual styling of X-based environments varies greatly; different programs may present radically different interfaces.

KDE stands for K Desktop Environment. It is a desktop environment for Linux based operation system. User can think KDE as a GUI for Linux OS. KDE has proved Linux users to make it use as easy as they use windows. KDE provides Linux users a graphical interface to choose their own customized desktop environment. You can choose your Graphical Interface among various available GUI interfaces that have their own look.

User can imagine Linux without KDE and GNOME just like DOS in windows. KDE and GNOME are much similar with Windows except they are related to Linux through x server rather than operation system.

GNOME is a desktop environment composed of free and open-source software that runs on Linux and most BSD derivatives. GNOME is developed by The GNOME Project, which is composed of both volunteers and paid contributors, the largest corporate contributor being Red Hat. It is an project that aims to develop software frameworks for the development of software, to program end-user applications based on these frameworks, and to coordinate efforts for internationalization and localization and accessibility of that software.

## **14.8 Glossary**

KDE—K Desktop Environment

GNOME--- GNU Network Object Model Environment

## **14.9 Answers to Self-Learning Exercise**

Q.1 (b)

Q.2 (c)

Q.3 (c)

## **14.10 Exercise**

Q.1 KDE has own its window manager called

- a) Twin
- b) Kwin

- c) Hello
  - d) Twin Twin
- Q.2 The KDE Control Center contains a number of configuration tools, called modules that allow you to configure and view information about your system called
- a) kcontrol
  - b) pcontrol
  - c) xcontrol
  - d) ycontrol
- Q.3 Which tab of the Appearance Preferences window enables you to change the font that GNOME uses for applications, documents, the desktop, window titles, and terminal emulators (fixed width)?
- a) Image
  - b) Color
  - c) Font
  - d) Width
- Q.4 Which shell environment variable is used to hold the X server ID string?
- a) COLOR
  - b) DISPLAY
  - c) VIEW
  - d) DESIGN
- Q.5 Which describes how to use a GUI to change a mouse between right-handed and left-handed?
- a) Keyboard Preferences
  - b) Mouse Preferences
  - c) Printer Preferences
  - d) Monitor Preferences

### **14.11 Answers of Exercise**

- Q.1 (b)
- Q.2 (a)
- Q.3 (c)

Q.4 (b)

Q.5 (b)

### **References and Suggested Readings**

1. A Practical Guide to Ubuntu Linux by Mark G. Sobell, Prentice Hall, 2007.
2. UNIX Concepts and Applications by Sumitabha Das, Tata McGraw Hill, 2008.
3. UNIX Shell Programming by Yashwant Kanetkar, BPB Publications, 2003.



## UNIT-15

# Linux Networking

### Structure of the Unit

- 15.0 Objective
- 15.1 Introduction
- 15.2 Basic Networking Configuration
- 15.3 Network File System
- 15.4 Network Sniffing
- 15.5 Installation and configuration of Proxy server (Squid)
- 15.6 Self Learning Exercise
- 15.7 Summary
- 15.8 Glossary
- 15.9 Answers to Self Learning Exercise
- 15.10 Exercise

### 15.0 Objective

This is a Linux networking section. Students have learned how to install Linux, practising a few basic Linux commands which enabled them to manage Linux system in the previous Linux basics and Linux administration chapters. Here in the Linux networking section, students will be exposed to what Linux is all about, serving network.

### 15.1 Introduction

This chapter explains how to set up a Local Area Network (LAN) that includes various servers, which lets Microsoft Windows and UNIX systems access shared files and printers hosted by Linux user. The chapter explains how to setup a simple LAN and describes how to install, configure, and administer file servers and clients. Integrating Linux system with an existing LAN is no more

complicated than setting up own LAN. It explains how to connect to an existing network and how to use Linux backup and recovery utilities so that client systems can create and use backups stored on the server.

One of the great strengths of Linux is its powerful and robust networking capabilities. The Linux networking setup is open to inspection and completely configurable and totally transparent.

Basic networking principles don't differ much between Windows and Linux, and indeed the principles aren't unfamiliar. This chapter starts with an overview of networking, and then looks in more detail at Linux networking on a Local Area Network (LAN).

## **15.2 Basic Network Configuration**

Computers are connected in a network to exchange information or resources with each other. Two or more computer connected through network media called computer network. There are number of network devices or media involved to form computer network. Computer loaded with Linux Operating System can also be a part of network whether it is small or large network by its multitasking and multiuser natures. Maintaining of system and network up and running is a task of System / Network Administrator's job. In this section we are going to review frequently used network configuration and various Linux commands.

### **15.2.1 *ifconfig* Command**

**ifconfig** (**interface configurator**) command is used to initialize an interface, assign **IP address** to interface and **enable** or **disable** interface on demand. With this command we can view **IP Address** and **Hardware / MAC address** assign to interface and also **MTU (Maximum Transmission Unit)** size.

```

tecmin@tecmin ~ $ ifconfig
eth0 Link encap:Ethernet HWaddr 28:d2:44:eb:bd:98
 inet addr:192.168.0.104 Bcast:192.168.0.255 Mask:255.255.255.0
 inet6 addr: fe80::2ad2:44ff:feeb:bd98/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:342087 errors:0 dropped:0 overruns:0 frame:0
 TX packets:233764 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:406375041 (406.3 MB) TX bytes:25096967 (25.0 MB)

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:5146 errors:0 dropped:0 overruns:0 frame:0
 TX packets:5146 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:0
 RX bytes:469809 (469.8 KB) TX bytes:469809 (469.8 KB)

wlan0 Link encap:Ethernet HWaddr 38:b1:db:7c:78:c7
 UP BROADCAST MULTICAST MTU:1500 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

tecmin@tecmin ~ $

```

*ifconfig* with interface (**eth0**) command only shows specific interface details like **IP Address**, **MAC Address** etc. with **-a** options will display all available interface details if it is disable also.

Assigning an **IP Address** and **Gateway** to interface on the fly. The setting will be removed in case of system reboot.

```
ifconfig eth0 192.168.50.5 netmask 255.255.255.0
```

## 15.2.2 Enable or Disable Specific Interface

To **enable** or **disable** specific Interface, we use command as follows.

Enable eth0

```
ifup eth0
```

Disable eth0

```
ifdown eth0
```

## 15.2.3 Setting MTU (Maximum Transmission Unit) size

By default **MTU** size is **1500**. We can set required **MTU** size with below command. Replace **XXXX** with size.

```
ifconfig eth0 mtu XXXX
```

### 15.2.4 Setting Interface in Promiscuous mode

Network interface only received packets belong to that particular NIC. If we put interface in **promiscuous** mode it will receive all the packets. This is very useful to capture packets and analyze later. For this it may require super-user access.

```
ifconfig eth0 -promisc
```

### 15.2.5 PING Command

**PING (Packet Internet Groper)** command is the best way to test connectivity between **two nodes**. Whether it is **Local Area Network (LAN)** or **Wide Area Network (WAN)**. Ping uses **ICMP (Internet Control Message Protocol)** to communicate to other devices. We can ping host name of **ip address** using below command.

```
ping 4.2.2.2
PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.
64 bytes from 4.2.2.2: icmp_seq=1 ttl=44 time=203 ms
64 bytes from 4.2.2.2: icmp_seq=2 ttl=44 time=201 ms
64 bytes from 4.2.2.2: icmp_seq=3 ttl=44 time=201 ms
```

In **Linux** ping command keeps executing until you interrupt. Ping with **-c** option exit after **N** number of request (success or error respond).

### 15.2.6 *traceroute* Command

**The *traceroute*** is a network troubleshooting utility which shows number of hops taken to reach destination also determine packets traveling path. Below we are tracing route to global **DNS server IP Address** and able to reach destination also shows path of that packet is traveling.

### 15.2.7 *netstat* Command



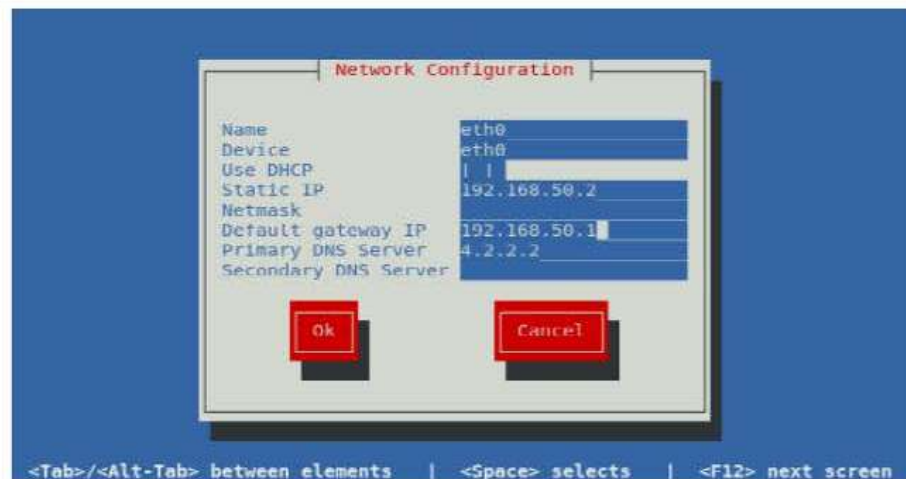
Netstat (Network Statistic) command display connection info, routing table information etc. To display routing table information use option as **-r**.

```
[root@linux-server root]# netstat -r
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
192.168.246.0 * 255.255.255.0 U 40 0 0 eth0
127.0.0.0 * 255.0.0.0 U 40 0 0 lo
default 192.168.246.1 0.0.0.0 UG 40 0 0 eth0
[root@linux-server root]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.246.0 * 255.255.255.0 U 0 0 0 eth0
127.0.0.0 * 255.0.0.0 U 0 0 0 lo
default 192.168.246.1 0.0.0.0 UG 0 0 0 eth0
[root@linux-server root]#
```

### 15.2.8 GUI tool system-config-network

Type **system-config-network** in command prompt to configure network setting and you will get nice **Graphical User Interface (GUI)** which may also use to configure **IP Address, Gateway, DNS** etc. as shown in below image.

```
system-config-network
```



### 15.2.9 Configuring a Gateway

Configuring a machine to switch packets between two Ethernets is pretty straightforward. Assume we're back at vlager, which is equipped with two Ethernet cards, each connected to one of the two networks. We have to do is configure both interfaces separately, giving them their respective IP addresses and matching routes, and that's it. It is quite useful to add information on the two interfaces to the hosts file as shown in the following example, so we have handy names for them, too:

```
172.16.1.1 vlager.vbrew.comvlager vlager-if1
```

```
172.16.2.1 vlager-if2
```

The sequence of commands to set up the two interfaces is then:

```
ifconfig eth0 vlager-if1
```

```
route add brew-net
```

```
ifconfig eth1 vlager-if2
```

```
route add wine-net
```

If this sequence doesn't work, make sure your kernel has been compiled with support for IP forwarding enabled. One good way to do this is to ensure that the first number on the second line of `/proc/net/snmp` is set to 1.

### 15.2.10 Network Commands

| File                          | Function                                          |
|-------------------------------|---------------------------------------------------|
| <code>/bin/netstat</code>     | Show network status.                              |
| <code>/bin/hostname</code>    | Delivers name of current host.                    |
| <code>/bin/ping</code>        | Sends ICMP ECHO_REQUEST packets to network hosts. |
| <code>/etc/hosts</code>       | list of hosts on network.                         |
| <code>/etc/hosts.equiv</code> | list of trusted hosts.                            |
| <code>/etc/http/conf/*</code> | HTTP configuration files                          |
| <code>/etc/inetd.conf</code>  | Inetd configuration file.                         |
| <code>/etc/netconfig</code>   | Configure networking products.                    |
| <code>/etc/protocols</code>   | List of Internet protocols.                       |
| <code>/etc/services</code>    | List of network services provided.                |
| <code>/usr/bin/rdate</code>   | Notify time server that date has changed.         |
| <code>/usr/bin/rdist</code>   | Remote file distribution program.                 |

|                      |                                         |
|----------------------|-----------------------------------------|
| /usr/bin/rlogin      | Remote login program.                   |
| /usr/bin/route       | Manually manipulate routing tables.     |
| /usr/bin/rwho        | Who is logged in on local network.      |
| /usr/bin/talk        | Talk to another user.                   |
| /usr/bin/telnet      | Telnet remote login program             |
| /usr/sbin/ftpsht     | Shutdown ftp services                   |
| /usr/sbin/httpd      | HTTP Daemon                             |
| /usr/sbin/inetd      | Internet 'super-server'                 |
| /usr/sbin/in.rlogind | Remote login (rlogin) daemon            |
| /usr/sbin/in.telnetq | Telnet Daemon                           |
| /usr/sbin/traceroute | Trace packet routes to remote machines. |

## 15.3 Network File System (NFS)

NFS (Network File System) is basically developed for sharing of files and folders between Linux/Unix systems by Sun Microsystems in 1980. It allows to mount local file systems over a network and remote hosts to interact with them as they are mounted locally on the same system. With the help of NFS, we can set up file sharing between Unix to Linux system and Linux to Unix system.

### 15.3.1 NFS Services

It's a System V-launched service. The NFS server package includes three facilities, included in the portmap and nfs-utils packages.

- portmap : It maps calls made from other machines to the correct RPC service (not required with NFSv4).

- `nfs`: It translates remote file sharing requests into requests on the local file system.
- `rpc.mountd`: This service is responsible for mounting and unmounting of file systems.

### 15.3.2 Important Files for NFS Configuration

- `/etc/exports`: Its a main configuration file of NFS, all exported files and directories are defined in this file at the NFS Server end.
- `/etc/fstab`: To mount a NFS directory on your system across the reboots, we need to make an entry in `/etc/fstab`.
- `/etc/sysconfig/nfs`: Configuration file of NFS to control on which port `rpc` and other services are listening.

### 15.3.3 Setup and Configure NFS Mounts on Linux Server

To setup NFS mounts, we'll be needing at least two Linux/Unix machines. Here we are using two servers. These are:

- NFS Server: `nfsserver.example.com` with IP-`192.168.0.100`
- NFS Client : `nfscient.example.com` with IP-`192.168.0.101`

We need to install NFS packages on our NFS Server as well as on NFS Client machine. We can install it via “yum” (Red Hat Linux) and “apt-get” (Debian and Ubuntu) package installers.

```
[root@nfsserver ~]# yum install nfs-utilsnfs-utils-lib
[root@nfsserver ~]# yum install portmap (not required with NFSv4)
```

```
[root@nfsserver ~]# apt-get install nfs-utilsnfs-utils-lib
```

Now start the services on both machines.

```
[root@nfsserver ~]# /etc/init.d/portmap start
[root@nfsserver ~]# /etc/init.d/nfs start
[root@nfsserver ~]# chkconfig --level 35 portmap on
```



```
[root@nfsserver ~]# chkconfig --level 35 nfs on
```

After installing packages and starting services on both the machines, we need to configure both the machines for file sharing.

### 15.3.4 Setting Up the NFS Server

First we have to configure export directory. For sharing a directory with NFS, we need to make an entry in “/etc/exports” configuration file. Here I’ll be creating a new directory named “nfsshare” in “/” partition to share with client server, you can also share an already existing directory with NFS.

```
[root@nfsserver ~]# mkdir /nfsshare
```

Now we need to make an entry in “/etc/exports” and restart the services to make our directory shareable in the network.

```
[root@nfsserver ~]# vi /etc/exports/nfsshare
192.168.0.101(rw,sync,no_root_squash)
```

In the above example, there is a directory in / partition named “nfsshare” is being shared with client IP “192.168.0.101” with read and write (rw) privilege, you can also use hostname of the client in the place of IP in above example.

Some other options we can use in “/etc/exports” file for file sharing are as follows.

- ro: With the help of this option we can provide read only access to the shared files i.e client will only be able to read.
- rw: This option allows the client server to both read and write access within the shared directory.
- sync: Sync confirms requests to the shared directory only once the changes have been committed.

- `no_subtree_check`: This option prevents the subtree checking. When a shared directory is the subdirectory of a larger file system, nfs performs scans of every directory above it, in order to verify its permissions and details. Disabling the subtree check may increase the reliability of NFS, but reduce security.
- `no_root_squash`: This phrase allows root to connect to the designated directory.

### 15.3.5 Setting Up the NFS Client

After configuring the NFS server, we need to mount that shared directory or partition in the client server. Here, we have to mount shared directories on NFS client.

Now at the NFS client end, we need to mount that directory in our server to access it locally. To do so, first we need to find out that share available on the remote server or NFS Server.

```
[root@nfsclient ~]# showmount -e 192.168.0.100 Export list for
192.168.0.100: /nfsshare 192.168.0.101
```

Above command shows that a directory named “nfsshare” is available at “192.168.0.100” to share with your server.

### 15.3.6 Mount Shared NFS Directory

To mount that shared NFS directory we can use following mount command.

```
[root@nfsclient ~]# mount -t nfs 192.168.0.100:/nfsshare /mnt/nfsshare
```

The above command will mount that shared directory in “/mnt/nfsshare” on the client server. You can verify it by following command.

```
[root@nfsclient ~]# mount | grep nfs
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
192.168.0.100:/nfsshare on /mnt type nfs (rw,addr=192.168.0.100)
```

The above mount command mounted the nfs shared directory on to nfs client temporarily, to mount an NFS directory permanently on your system across the reboots, we need to make an entry in “/etc/fstab”.

We can test our NFS server setup by creating a test file on the server end and check its availability at nfs client side or vice-versa.

### **15.3.7 Important commands for NFS**

Some more important commands for NFS as follow:

- showmount -e : Shows the available shares on your local machine
- showmount -e <server-ip or hostname>: Lists the available shares at the remote server
- showmount -d : Lists all the sub directories
- exportfs -v : Displays a list of shares files and options on a server
- exportfs -a : Exports all shares listed in /etc/exports, or given name
- exportfs -u : Unexports all shares listed in /etc/exports, or given name
- exportfs -r : Refresh the server’s list after modifying /etc/exports

### **15.3.8 Benefits of NFS**

- NFS allows local access to remote files.
- It uses standard client/server architecture for file sharing between all Unix based machines.
- With NFS it is not necessary that both machines run on the same OS.
- With the help of NFS we can configure centralized storage solutions.
- Users get their data irrespective of physical location.
- No manual refresh needed for new files.
- Newer version of NFS also supports acl, pseudo root mounts.
- Can be secured with Firewalls and Kerberos.

## 15.4 Network Sniffing

A network administrator should be able to use a sniffer like Wireshark or tcpdump to trouble shoot network problems. A student should often use a sniffer to learn about networking. This section introduces network sniffing.

Sniffing a network can generate many thousands of packets in a very short time. This can be overwhelming. Try to mitigate by isolating your sniffer on the network. Preferably sniff an isolated virtual network interface over which you control all traffic. If we are at home to learn sniffing, then it could help to close all network programs on computer, and disconnect other computers and devices like smartphones and tablets to minimize the traffic.

### 15.4.1 Sniffing ping

We started the sniffer and captured all packets while doing these three ping commands (there is no need for root to do this):

```
root@debian7:~# ping -c2 ns1.paul.local
PING ns1.paul.local (10.104.33.30) 56(84) bytes of data.
64 bytes from 10.104.33.30: icmp_req=1 ttl=64 time=0.010 ms
64 bytes from 10.104.33.30: icmp_req=2 ttl=64 time=0.023 ms
--- ns1.paul.local ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.010/0.016/0.023/0.007 ms
root@debian7:~# ping -c3 linux-training.be
PING linux-training.be (188.93.155.87) 56(84) bytes of data.
64 bytes from antares.ginsys.net (188.93.155.87): icmp_req=1 ttl=56
time=15.6 ms
64 bytes from antares.ginsys.net (188.93.155.87): icmp_req=2 ttl=56
time=17.8 ms
64 bytes from antares.ginsys.net (188.93.155.87): icmp_req=3 ttl=56
time=14.7 ms
--- linux-training.be ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 14.756/16.110/17.881/1.309 ms
root@debian7:~# ping -c1 centos7.paul.local
PING centos7.paul.local (10.104.33.31) 56(84) bytes of data.
64 bytes from 10.104.33.31: icmp_req=1 ttl=64 time=0.590 ms
--- centos7.paul.local ping statistics ---
```

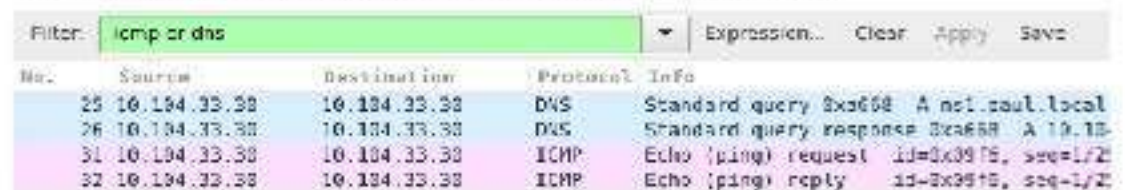


**1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 0.590/0.590/0.590/0.000 ms**

In total more than 200 packets were sniffed from the network. Things become clearer when we enter `icmp` in the filter field and press the apply button.

### 15.4.2 Sniffing ping and dns

Using the same capture as before, but now with a different filter. We want to see both `dns` and `icmp` traffic, so we enter both in the filter field. We put `dns` or `icmp` in the filter to achieve this. Putting `dns` and `icmp` would render nothing because there is no packet that matches both protocols.

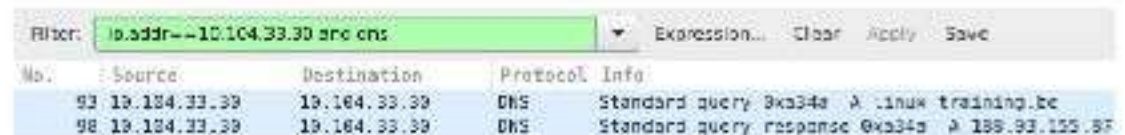


The screenshot shows a Wireshark packet capture with a filter of `icmp or dns`. The packet list contains four entries:

| No. | Source       | Destination  | Protocol | Info                                   |
|-----|--------------|--------------|----------|----------------------------------------|
| 25  | 10.104.33.30 | 10.104.33.30 | DNS      | Standard query 0xa668 A ns1.soul.local |
| 26  | 10.104.33.30 | 10.104.33.30 | DNS      | Standard query response 0xa668 A 10.10 |
| 31  | 10.104.33.30 | 10.104.33.30 | ICMP     | Echo (ping) request id=0x0976, seq=1/2 |
| 32  | 10.104.33.30 | 10.104.33.30 | ICMP     | Echo (ping) reply id=0x0976, seq=1/2   |

In the screenshot above you can see that packets 25 and 26 both have `10.104.33.30` as source and destination ip address. That is because the `dns` client is the same computer as the `dns` server. The same is true for packets 31 and 32, since the machine is actually pinging itself.

This is a screenshot that filters for `dns` packets that contain a certain ip address. The filter in use is `ip.addr==10.104.33.30 and dns`. The 'and' directive forces each displayed packet to match both conditions.



The screenshot shows a Wireshark packet capture with a filter of `ip.addr==10.104.33.30 and dns`. The packet list contains two entries:

| No. | Source       | Destination  | Protocol | Info                                           |
|-----|--------------|--------------|----------|------------------------------------------------|
| 93  | 19.164.33.39 | 19.164.33.39 | DNS      | Standard query 0xa34a A linux.training.be      |
| 98  | 19.164.33.39 | 19.164.33.39 | DNS      | Standard query response 0xa34a A 189.93.125.87 |

Packet 93 is the `dns` query for the `A` record of `linux-training.be`. Packet 98 is the response from the `dns` server. What do you think happened in the packets between 93 and 98? Try to answer this before reading on (it always helps to try to predict what you will see, and then checking your prediction).

### 15.4.3 tcpdump

Sniffing on the command line can be done with `tcpdump`. Here are some examples.

Using the `tcpdump host $ip` command displays all traffic with one host (`192.168.1.38` in this example).

Capturing only ssh (tcp port 22) traffic can be done with tcpdumpteport port Sport. This screenshot is cropped to 76 characters for readability in the pdf.

```
root@ubuntu910:~# tcpdump host 192.168.1.38
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

root@deb503:~# tcpdumpteport port 22
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:22:20.716313 IP deb503.local.37973 > rhel53.local.ssh: P
666050963:66605
14:22:20.719936 IP rhel53.local.ssh > deb503.local.37973: P 1:49(48) ack
48
14:22:20.720922 IP rhel53.local.ssh > deb503.local.37973: P 49:113(64) ack
14:22:20.721321 IP rhel53.local.ssh > deb503.local.37973: P 113:161(48)
ack
14:22:20.721820 IP deb503.local.37973 > rhel53.local.ssh: .ack 161 win 200
14:22:20.722492 IP rhel53.local.ssh > deb503.local.37973: P 161:225(64)
ack
14:22:20.760602 IP deb503.local.37973 > rhel53.local.ssh: .ack 225 win 200
14:22:23.108106 IP deb503.local.54424 > ubuntu910.local.ssh: P
467252637:46
14:22:23.116804 IP ubuntu910.local.ssh > deb503.local.54424: P 1:81(80)
ack
14:22:23.116844 IP deb503.local.54424 > ubuntu910.local.ssh: .ack 81 win
2
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

Same as above, but write the output to a file with the tcpdump -w \$filename command.

```
root@ubuntu910:~# tcpdump -w sshdump.tcpdumpteport port 22tcpdump:
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
^C
17 packets captured
```

17 packets received by filter

0 packets dropped by kernel

With `tcpdump -r $filename` the file created above can be displayed.

```
root@ubuntu910:~# tcpdump -r sshdump.tcpdump
```

#### **15.4.4 Network Sniffing: Hands on Practice**

Practice:-

1. Install wireshark on your computer (not inside a virtual machine).
2. Start a ping between your computer and another computer.
3. Start sniffing the network.
4. Display only the ping echo's in the top pane using a filter.
5. Now ping to a name (like `www.linux-training.be`) and try to sniff the DNS query and response. Which DNS server was used ? Was it a tcp or udp query and response ?

**Solution: -**

1. Install wireshark on your computer (not inside a virtual machine).  
Debian/Ubuntu: `aptitude installwireshark`  
Red Hat/Mandriva/Fedora: `yum install wireshark`
2. Start a ping between your computer and another computer.  
`ping $ip_address`
3. Start sniffing the network.  
`(sudo) wireshark`  
select an interface (probably `eth0`)
4. Display only the ping echo's in the top pane using a filter.  
type 'icmp' (without quotes) in the filter box, and then click 'apply'
5. Now ping to a name (like `www.linux-training.be`) and try to sniff the DNS query and response. Which DNS server was used ? Was it a tcp or udp query and response ? First start the sniffer. Enter 'dns' in the filter box and click apply.

```
root@ubuntu910:~# ping www.linux.com
```

```
PING www.linux.com (88.151.243.8) 56(84) bytes of data.
```



64 bytes from fosfor.openminds.be (88.151.243.8): icmp\_seq=1 ttl=58 time=14.9 ms

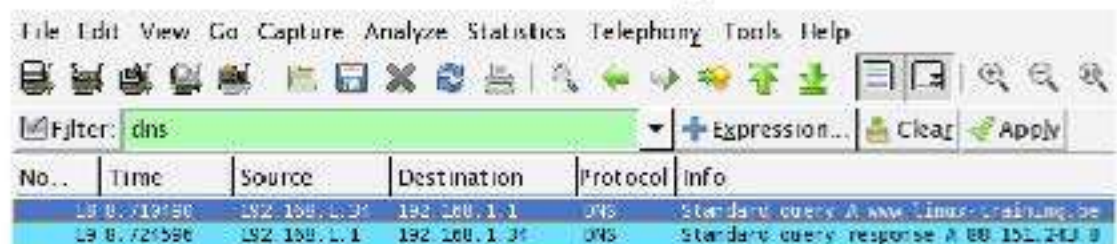
64 bytes from fosfor.openminds.be (88.151.243.8): icmp\_seq=2 ttl=58 time=16.0 ms

^C

2 packets transmitted, 2 received, 0% packet loss, time 1002ms

rtt min/avg/max/mdev = 14.984/15.539/16.095/0.569 ms

The wireshark screen should look something like this.



The screenshot shows the Wireshark interface with the filter set to 'dns'. The packet list pane shows two packets:

| No. | Time     | Source       | Destination  | Protocol | Info                                   |
|-----|----------|--------------|--------------|----------|----------------------------------------|
| 18  | 0.719486 | 192.168.1.31 | 192.168.1.1  | DNS      | Standard query A www.linux-training.be |
| 19  | 0.721586 | 192.168.1.1  | 192.168.1.31 | DNS      | Standard query response A 88.151.243.8 |

The details in wireshark will say the DNS query was inside audp packet.

## 15.5 Installation and configuration of Proxy server(Squid)

A proxy server is a server that caches the internet. Clients connect to the proxy server with a request for an internet server. The proxy server will connect to the internet server on behalf of the client. The proxy server will also cache the pages retrieved from the internet server. A proxy server may provide pages from his cache to a client, instead of connecting to the internet server to retrieve the (same) pages. A proxy server has two main advantages. It improves web surfing speed when returning cached data to clients, and it reduces the required bandwidth (cost) to the internet. Smaller organizations sometimes put the proxy server on the same physical computer that serves as a NAT to the internet. In larger organizations, the proxy server is one of many servers in the DMZ. When web traffic passes via a proxy server, it is common practice to configure the proxy with extra settings for access control. Access control in a proxy server can mean user account access, but also website(url), ip-address or dns restrictions.

We can find lists of open proxy servers on the internet that enable to surf anonymously. This works when the proxy server connects on your behalf to a website, without logging ip-address. But be careful, these (listed) open proxy servers could be created in order to eavesdrop upon their users.



## 15.5.1 squid

This section introduces the squid proxy server (<http://www.squid-cache.org>). We will first configure squid as a normal proxy server.

This screenshot shows how to install squid on Debian with aptitude. Use yum in case of Red Hat/CentOS.

```
root@debian7:~# aptitude install squid
The following NEW packages will be installed:
squid squid-common{a} squid-langpack{a}
0 packages upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,513 kB of archives. After unpacking 4,540 kB will be used.
Do you want to continue? [Y/n/?]
...output truncated...
Setting up squid-langpack (20120616-1) ...
Setting up squid-common (2.7.STABLE9-4.1) ...
Setting up squid (2.7.STABLE9-4.1) ...
Creating squid spool directory structure
2014/08/01 15:19:31 | Creating Swap Directories
Restarting Squid HTTP proxy: squid.
```

squid's main configuration file is `/etc/squid/squid.conf`. The file explains every parameter in great detail.

```
root@debian7:~# wc -l /etc/squid/squid.conf
4948 /etc/squid/squid.conf
```

By default the squid proxy server will listen to port 3128.

```
root@debian7:~# grep ^http_port /etc/squid/squid.conf
http_port 3128
root@debian7:~#
```

We can manage squid with the standard service command as shown in this screenshot.

```
root@debian7:~# service squid start
Starting Squid HTTP proxy: squid.
root@debian7:~# service squid restart
Restarting Squid HTTP proxy: squid.
root@debian7:~# service squid status
```



Disabling the proxy with `service squid stop` should result in an error message similar to this screenshot.



### 15.5.2 Upside down images

A proxy server sits inbetween your browser and the internet. So besides caching of internet data (the original function of a proxy server) and besides firewall like restrictions based on www content, a proxy server is in the perfect position to alter the WebPages that you visit. We could for instance change the advertising on a webpage (or remove certain advertisers), or like we do in this example; change all images so they are upside down. The server needs command line tools to manipulate images and a perl script that uses these tools (and wget to download the images locally and serve them with apache2). In this example we use imagemagick (which provides tools like convert and mogrify).

```
root@debian7:~# aptitude install imagemagick wget perl apache2
...output truncated...
root@debian7:~# dpkg -S $(readlink -f $(which mogrify))
imagemagick: /usr/bin/mogrify.im6
root@debian7:~#
```

The standard log file location for squid is `/var/log/squid`.

```
[root@RHEL4 ~]# grep "/var/log" /etc/squid/squid.conf
cache_access_log /var/log/squid/access.log
cache_log /var/log/squid/cache.log
cache_store_log /var/log/squid/store.log
```

The default squid setup only allows localhost access. To enable access for a private networkrange, look for the "INSERT YOUR OWN RULE(S) HERE..." sentence in squid.conf and add two lines similar to the screenshot below.

```
INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR
CLIENTS
aclcompany_networksrc 192.168.1.0/24
http_access allow company_network
```

When we test the squid, we must sure that the server running squid has access to the internet.

Then configure a browser on a client to use the proxy server, or we could set the

HTTP\_PROXY (sometimes http\_proxy) variable to point command line programs to theproxy.

Testing a client machine can then be done with wget (wget -q is used to simplify the screen shot).

```
[root@RHEL4 ~]# wget -q http://linux-training.be/index.html
[root@RHEL4 ~]# ls -l index.html
-rw-r--r-- 1 root root 2269 Sep 18 13:18 index.html
[root@RHEL4 ~]#

[root@fedora ~]# export HTTP_PROXY=http://192.168.1.39:8080
[root@ubuntu ~]# export http_proxy=http://192.168.1.39:8080

[root@RHEL5 ~]# > /etc/resolv.conf
[root@RHEL5 ~]# wget -q http://www.linux-training.be/index.html
[root@RHEL5 ~]# ls -l index.html
-rw-r--r-- 1 root root 2269 Sep 18 2008 index.html
[root@RHEL5 ~]#
```



## 15.6 Self Learning Exercise

1. MAC addresses are also known as.
  - A. Hardware address
  - B. Physical address
  - C. both
  - D. IP address
2. UDP is an unreliable protocol.
  - A. True
  - B. False
3. Which daemon is used for the Apache server?
  - A. Apached
  - B. Httpd
  - C. HTML
  - D. Shhttp
  - E. None of the
- Q.4 What command do you use to add routes to a Linux router?
  - A. Addroute
  - B. Route
  - C. Netstat
  - D. Net
  - E. None of the above
- Q.5 Which of the following command is used to mount NFS filesystems?
  - A.Nfsmount
  - B.Knfsd
  - C.Mount
  - D.All of the above
  - E.None of the above

## 15.7 Summary

- Linux networking devices are created directly in the Linux kernel when a kernel module supporting a type of networking is loaded
- Many types of networking are supported in Linux, though the most widely used for standard LANs is Ethernet
- The `ifconfig` command sets up a networking interface in the Linux kernel or displays the current setup for all configured interfaces
- The `route` command establishes entries in the kernel IP routing table or displays the current routing table entries
- A number of networking scripts are used to streamline the configuration of Linux networking, making it more flexible and robust
- Networking configuration parameters are stored in files within the `/etc/sysconfig/network-scripts` directory
- IP aliasing occurs when multiple IP addresses are assigned to the same physical network interface
- Network File System allows to mount local file systems over a network and remote hosts to interact with them as they are mounted locally on the same system. With the help of NFS, we can set up file sharing between Unix to Linux system and Linux to Unix system.
- The `telnet` utility lets you connect to a remote host as if you were sitting at that host
- Squid :A proxy in linux is a host which relays web access requests from clients  
used when clients do not access the web directly
- Proxy server is used for security, logging, accounting and performance.

## 15.8 Glossary

- **Connection:** In networking, a connection refers to pieces of related information that are transferred through a network. This generally infers that a connection is built before the data transfer (by following the procedures laid out in a protocol) and then is deconstructed at the end of the data transfer.
- **Packet:** A packet is, generally speaking, the most basic unit that is transferred over a network. When communicating over a network, packets are the envelopes that carry your data (in pieces) from one end point to the other.
- **Network Interface:** A network interface can refer to any kind of software interface to networking hardware. For instance, if you have two network cards in your computer, you can control and configure each network interface associated with them individually.
- **MTU(Maximum Transmission Unit)-** A maximum transmission unit (MTU) is the largest size packet or frame, specified in octets (eight-bit bytes), that can be sent in a packet- or frame-based network such as the Internet.
- **ICMP(Internet Control Message Protocol)-** is an error-reporting protocol network devices like routers use to generate error messages to the source IP address when network problems prevent delivery of IP packets.
- **DNS(Domain Name Service)-** The DNS database contains records that map user-friendly alphanumeric names for network resources to the IP address used by those resources for communication.
- **NAT(Network Address Translation)-** is the process where a network device, usually a firewall, assigns a public address to a computer (or group of computers) inside a private network.

## 15.9 Answer to self learning Exercises

Q.1 (C)

Q.2 (A)

Q.3 (B)

Q.4 (B)

Q.5 (C)

### **15.10 Exercise**

Q.1 What is web server?

Q.2 What is the prerequisite for hosting multiple web site on a machine?

Q.3 Identify the contents of the files that are needed to host a web site.

Q.4 What is Network File System?

Q.5 What do you mean by proxy server? What are the advantages and disadvantages of proxy server?

Q.6 What do you mean by network sniffing?

### **References and Suggested Readings**

1. <http://www.basicconfig.com/linuxservers.html>
2. <http://www.tecmint.com/install-openssh-server-in-linux/>
3. Richard Peterson, "Linux: The Complete Reference", Osborne McGrawHill.



# **UNIT-16**

## **Introduction to Server**

### **Structure of the Unit**

- 16.0 Objective
- 16.1 Introduction
- 16.2. DNS Server (BIND)
- 16.3 Mail Server
- 16.4 Web Server (Apache)
- 16.5 File Server (Samba)
- 16.6 DHCP Server
- 16.7 Installation and Configuration of a SSH server and client
- 16.8 Installation and configuration of FTP server and client
- 16.9 Self Learning Exercise
- 16.10 Summary
- 16.11 Glossary
- 16.12 Answers to Self Learning Exercise
- 16.13 Exercise

### **16.0 Objective**

In this chapter we shall focus upon the following topics:

- DNS Server (BIND)
- Mail Server
- Web Server
- File Server (Samba)
- DHCP Server
- Installation and Configuration of a SSH server and client
- Installation and configuration of FTP server and client

### **16.1 Introduction**

A single Linux system can provide several different kinds of services, ranging from security to administration and including more obvious Internet services like websites and FTP sites, email, and printing. Security tools such as SSH and Kerberos run as services, along with administrative network tools such as DHCP and LDAP. The network connection interface is itself a service that you can restart at will. Each service operates as a continually running daemon looking for requests for its particular services. In the case of a web service, the requests will come from remote users. You can turn services on or off by starting or shutting down their daemons. The process of starting up or shutting down a service is handled by service scripts. This Chapter will help students understand and configure different servers like DNS Server (BIND), Mail Server, Web Server, File Server (Samba), DHCP Server. Along with Installation and Configuration of a SSH server and ftp server and client.

## 16.2 DNS Server (BIND)

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

In a DNS server such as BIND (Berkeley Internet Name Domain), all information is stored in basic data elements called resource records (RR). The resource record is usually a fully qualified domain name (FQDN) of a host, and is broken down into multiple sections organized into a tree-like hierarchy. This hierarchy consists of a main trunk, primary branches, secondary branches, and so on.

There are two nameserver configuration types:

**Authoritative:** It answers to resource records that are part of their zones only. This category includes both primary (master) and secondary (slave) nameservers.

**Recursive:** offer resolution services, but they are not authoritative for any zone. Answers for all resolutions are cached in a memory for a fixed period of time, which is specified by the retrieved resource record.

BIND consists of a set of DNS-related programs. It contains a nameserver called `named`, an administration utility called `rndc`, and a debugging tool called `dig`.

### 16.2.1. Configuring the `named` Service

When the `named` service is started, it reads the configuration from the files as described in following Table 1: The `named` service configuration files.

Table-1: **The `named` service configuration files**

| Path                         | Description                                                                                     |
|------------------------------|-------------------------------------------------------------------------------------------------|
| <code>/etc/named.conf</code> | The main configuration file.                                                                    |
| <code>/etc/named/</code>     | An auxiliary directory for configuration files that is included in the main configuration file. |

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

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

The following types of statements are commonly used in `/etc/named.conf`:

**Acl**

The `acl` (Access Control List) statement allows you to define groups of hosts, so that they can be permitted or denied access to the nameserver. It takes the following form:

```
acl acl-name {
 match-element;
 ...
};
```

The `acl-name` statement name is the name of the access control list, and the `match-element` option is usually an individual IP address (such as 10.0.1.1) or a CIDR (Classless Inter-Domain Routing) network notation (for example, 10.0.1.0/24). For a list of already defined keywords, see below “Predefined access control lists”.

Table: Predefined access control lists

| Keyword   | Description                                                                   |
|-----------|-------------------------------------------------------------------------------|
| any       | Matches every IP address                                                      |
| localhost | Matches IP address that is in use by the local system                         |
| localnets | Matches any IP address on any network to which the local system is connected. |
| None      | Does not match any IP address                                                 |

```
acl black-hats {
 10.0.2.0/24;
 192.168.0.0/24;
 1234:5678::9abc/24;
};
acl red-hats {
 10.0.1.0/24;
};
options {
 blackhole { black-hats; };
 allow-query { red-hats; };
 allow-query-cache { red-hats; };
};
```

### include

The `include` statement allows you to include files in the `/etc/named.conf`, so that potentially sensitive data can be placed in a separate file with restricted permissions. It takes the following form:



```
include "file-name"
```

The *file-name* statement name is an absolute path to a file.

```
include "/etc/named.rfc1912.zones";
```

**options**

The options statement allows you to define global server configuration options as well as to set defaults for other statements. It can be used to specify the location of the named working directory, the types of queries allowed, and much more. It takes the following form:

```
options {
 option;
 ...
};
```

For a list of frequently used *option* directives, shown in below Table

Table :Commonly used options

| Option            | Description                                                                                                                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| allow-query       | Specifies which hosts are allowed to query the nameserver for authoritative resource records. It accepts an access control list, a collection of IP addresses, or networks in the CIDR notation. All hosts are allowed by default. |
| allow-query-cache | Specifies which hosts are allowed to query the nameserver for non-authoritative data such as recursive queries. Only localhost and localnets are allowed by default.                                                               |
| blackhole         | Specifies which hosts are <i>not</i> allowed to query the nameserver. This option should be used when particular host or network floods the server with requests. The default option is none.                                      |
| directory         | Specifies a working directory for the named service. The default option is /var/named/.                                                                                                                                            |

|            |                                                                                                                                                                                                                                                                                                                                                                           |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| forwarders | Specifies a list of valid IP addresses for nameservers to which the requests should be forwarded for resolution.                                                                                                                                                                                                                                                          |
| forward    | Specifies the behaviour of the forwarders directive. It accepts the following options:<br>first — The server will query the nameservers listed in the forwardersdirective before attempting to resolve the name on its own.<br>only — When unable to query the nameservers listed in the forwardersdirective, the server will not attempt to resolve the name on its own. |
| listen-on  | Specifies the IPv4 network interface on which to listen for queries. On a DNS server that also acts as a gateway, you can use this option to answer queries originating from a single network only. All IPv4 interfaces are used by default.                                                                                                                              |

### 16.2.2 Common Resource Records

The following resource records are commonly used in zone files:

#### A

The *Address* record specifies an IP address to be assigned to a name. It takes the following form:

```
hostname IN A IP-address
```

If the *hostname* value is omitted, the record will point to the last specified *hostname*.

Following are the requests for server1.example.com are pointed to 10.0.1.3 or 10.0.1.5.

```
server1 IN A 10.0.1.3
IN A 10.0.1.5
```

#### CNAME

The *Canonical Name* record maps one name to another. Because of this, this type of record is sometimes referred to as an *alias record*. It takes the following form:

```
alias-name IN CNAME real-name
```

- CNAME records are most commonly used to point to services that use a common naming scheme, such as www for Web servers. However, there are multiple restrictions for their usage:
- CNAME records should not point to other CNAME records. This is mainly to avoid possible infinite loops.
- CNAME records should not contain other resource record types (such as A, NS, MX, etc.). The only exception are DNSSEC related records (that is, RRSIG, NSEC, etc.) when the zone is signed.

Other resource records that point to the fully qualified domain name (FQDN) of a host (that is, NS, MX, PTR) should not point to a CNAME record.

The A record binds a host name to an IP address, while the CNAME record points the commonly used www host name to it.

```
server1 IN A 10.0.1.5
www IN CNAME server1
```

## MX

The *Mail Exchange* record specifies where the mail sent to a particular namespace controlled by this zone should go. It takes the following form:

```
IN MX preference-value email-server-name
```

The *email-server-name* is a fully qualified domain name (FQDN). The *preference-value* allows numerical ranking of the email servers for a namespace, giving preference to some email systems over others. The MX resource record with the lowest *preference-value* is preferred over the others. However, multiple email servers can possess the same value to distribute email traffic evenly among them.

The first mail.example.com email server is preferred to the mail2.example.com email server when receiving email destined for the example.com domain.

```
example.com. IN MX 10 mail.example.com.
```

```
IN MX 20 mail2.example.com.
```

## NS

The *Nameserver* record announces authoritative nameservers for a particular zone. It takes the following form:

```
IN NS nameserver-name
```

The *nameserver-name* should be a fully qualified domain name (FQDN). Note that when two nameservers are listed as authoritative for the domain, it is not important whether these nameservers are secondary nameservers, or if one of them is a primary server. They are both still considered authoritative.

```
IN NS dns1.example.com.
```

```
IN NS dns2.example.com.
```

## PTR

The *Pointer* record points to another part of the namespace. It takes the following form:

```
last-IP-digit IN PTR FQDN-of-system
```

The *last-IP-digit* directive is the last number in an IP address, and the *FQDN-of-system* is a fully qualified domain name (FQDN).

PTR records are primarily used for reverse name resolution, as they point IP addresses back to a particular name.

## SOA

The *Start of Authority* record announces important authoritative information about a namespace to the nameserver. Located after the directives, it is the first resource record in a zone file. It takes the following form:

```
@ IN SOA primary-name-server hostmaster-email (
serial-number
time-to-refresh
time-to-retry
time-to-expire
```



*minimum-TTL* )

The directives are as follows:

- The @ symbol places the \$ORIGIN directive (or the zone's name if the \$ORIGIN directive is not set) as the namespace being defined by this SOA resource record.
- The *primary-name-server* directive is the host name of the primary nameserver that is authoritative for this domain.
- The *hostmaster-email* directive is the email of the person to contact about the namespace.
- The *serial-number* directive is a numerical value incremented every time the zone file is altered to indicate it is time for the named service to reload the zone.
- The *time-to-refresh* directive is the numerical value secondary nameservers use to determine how long to wait before asking the primary nameserver if any changes have been made to the zone.
- The *time-to-retry* directive is a numerical value used by secondary nameservers to determine the length of time to wait before issuing a refresh request in the event that the primary nameserver is not answering. If the primary server has not replied to a refresh request before the amount of time specified in the *time-to-expire* directive elapses, the secondary servers stop responding as an authority for requests concerning that namespace.
- In BIND 4 and 8, the *minimum-TTL* directive is the amount of time other nameservers cache the zone's information. In BIND 9, it defines how long negative answers are cached for. Caching of negative answers can be set to a maximum of 3 hours (that is, 3H).

## 16.3 Mail Server

Linux offers many advanced applications to serve and access email. This section describes modern email protocols in use today, and some of the programs designed to send and receive email.

### 16.3.1 Email Protocols

Today, email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

### **16.3.2 Mail Transport Protocols**

Mail delivery from a client application to the server, and from an originating server to the destination server, is handled by the *Simple Mail Transfer Protocol (SMTP)*. The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Red Hat Enterprise Linux, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail. One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or *spam* possible. Imposing relay restrictions limits random users on the Internet from sending email through your SMTP server, to other servers on the internet. Servers that do not impose such restrictions are called *open relay* servers.

### **16.3.4 Mail Access Protocols**

There are two primary protocols used by email client applications to retrieve email from mail servers: the Post Office Protocol (POP) and the Internet Message Access Protocol (IMAP).

POP: The default POP server under Red Hat Enterprise Linux is Dovecot and is provided by the dovecot package.

When using a POP server, email messages are downloaded by email client applications. By default, most POP email clients are automatically configured

to delete the message on the email server after it has been successfully transferred, however this setting usually can be changed.

POP is fully compatible with important Internet messaging standards, such as Multipurpose Internet Mail Extensions (MIME), which allow for email attachments. POP works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections, POP requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments. The most current version of the standard POP protocol is POP3. There are, however, a variety of lesser-used POP protocol variants:

- APOP — POP3 with MD5 authentication. An encoded hash of the user's password is sent from the email client to the server rather than sending an unencrypted password.
- KPOP — POP3 with Kerberos authentication.
- RPOP — POP3 with RPOP authentication. This uses a per-user ID, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so RPOP is no more secure than standard POP.

### **16.3.5 IMAP**

The default IMAP server under Red Hat Enterprise Linux is Dovecot and is provided by the dovecot package.

When using an IMAP mail server, email messages remain on the server where users can read or delete them. IMAP also allows client applications to create, rename, or delete mail directories on the server to organize and store email.

IMAP is particularly useful for users who access their email using multiple machines. The protocol is also convenient for users connecting to the mail server via a slow connection, because only the email header information is downloaded for messages until opened, saving bandwidth. The user also has the ability to delete messages without viewing or downloading them.

For convenience, IMAP client applications are capable of caching copies of messages locally, so the user can browse previously read messages when not directly connected to the IMAP server.

IMAP, like POP, is fully compatible with important Internet messaging standards, such as MIME, which allow for email attachments.

For added security, it is possible to use SSL encryption for client authentication and data transfer sessions. This can be enabled by using the `imaps` service, or by using the `stunnel` program.

Other free, as well as commercial, IMAP clients and servers are available, many of which extend the IMAP protocol and provide additional functionality.

### 16.3.5 Dovecot

The `imap-login` and `pop3-login` processes which implement the IMAP and POP3 protocols are spawned by the master `dovecot` daemon included in the `dovecot` package. The use of IMAP and POP is configured through the `/etc/dovecot/dovecot.conf` configuration file; by default `dovecot` runs IMAP and POP3 together with their secure versions using SSL. To configure `dovecot` to use POP, complete the following steps:

1. Edit the `/etc/dovecot/dovecot.conf` configuration file to make sure the `protocols` variable is uncommented (remove the hash sign (`#`) at the beginning of the line) and contains the `pop3` argument. For example:

```
protocols = imap pop3 lmtp
```

When the `protocols` variable is left commented out, `dovecot` will use the default values as described above.

2. Make the change operational for the current session by running the following command:

```
~]# service dovecot restart
```

3. Make the change operational after the next reboot by running the command:

```
~]# chkconfig dovecot on
```

Unlike SMTP, both IMAP and POP3 require connecting clients to authenticate using a user name and password. By default, passwords for both protocols are passed over the network unencrypted.

To configure SSL on `dovecot`:

- Edit the `/etc/dovecot/conf.d/10-ssl.conf` configuration to make sure the `ssl_cipher_list` variable is uncommented, and append `!SSLv3`:



```
ssl_cipher_list = ALL:!LOW:!SSLv2:!EXP:!aNULL:!SSLv3
```

These values ensure that dovecot avoids SSL versions 2 and also 3, which are both known to be insecure. This is due to the vulnerability described in POODLE: SSLv3 vulnerability (CVE-2014-3566).

## 16.4 Web Server

HTTP (Hypertext Transfer Protocol) server, or a *web server*, is a network service that serves content to a client over the web. This typically means web pages, but any other documents can be served as well.

### 16.4.1 The Apache HTTP Server

This section focuses on the **Apache HTTP Server 2.2**, a robust, full-featured open source web server developed by the Apache Software Foundation, that is included in Red Hat Enterprise Linux 6. It describes the basic configuration of the `httpd` service, and covers advanced topics such as adding server modules, setting up virtual hosts, or configuring the secure HTTP server. There are important differences between the Apache HTTP Server 2.2 and version 2.0, and if you are upgrading from a previous release of Red Hat Enterprise Linux, you will need to update the `httpd` service configuration accordingly. This section reviews some of the newly added features, outlines important changes, and guides you through the update of older configuration files.

The Apache HTTP Server version 2.2 introduces the following enhancements:

- Improved caching modules, that is, `mod_cache` and `mod_disk_cache`.
- Support for proxy load balancing, that is, the `mod_proxy_balancer` module.
- Support for large files on 32-bit architectures, allowing the web server to handle files greater than 2GB.
- A new structure for authentication and authorization support, replacing the authentication modules provided in previous versions.

#### Running the `httpd` Service

This section describes how to start, stop, restart, and check the current status of the Apache HTTP Server. To be able to use the `httpd` service, make sure you have the `httpd` installed. You can do so by using the following command:

```
~]# yum install httpd
```

### 16.4.2 Starting the Service

To run the httpd service, type the following at a shell prompt as root:

```
~]# service httpd start
Starting httpd: [OK]
```

If you want the service to start automatically at the boot time, use the following command:

```
~]#chkconfighttpd on
```

This will enable the service for runlevel 2, 3, 4, and 5..

### 16.4.3 Stopping the Service

To stop the running httpd service, type the following at a shell prompt as root:

```
~]# service httpd stop
Stopping httpd: [OK]
```

To prevent the service from starting automatically at the boot time, type:

```
~]#chkconfighttpd off
```

This will disable the service for all runlevels. Alternatively, you can use the **Service Configuration** utility also under “Enabling and Disabling a Service” option.

### 16.4.5 Restarting the Service

There are three different ways to restart a running httpd service:

1. To restart the service completely, enter the following command as root:

```
2. ~]# service httpd restart
3. Stopping httpd: [OK]
```

```
Starting httpd: [OK]
```

This stops the running httpd service and immediately starts it again. Use this command after installing or removing a dynamically loaded module such as PHP.

4. To only reload the configuration, as root, type:

```
~]# service httpd reload
```

This causes the running httpd service to reload its configuration file. Any requests being currently processed will be interrupted, which may cause a client browser to display an error message or render a partial page.

5. To reload the configuration without affecting active requests, enter the following command as root:

```
~]# service httpd graceful
```

This causes the running httpd service to reload its configuration file. Any requests being currently processed will use the old configuration.

Alternatively, you can use the Service Configuration utility also under “Starting, Restarting, and Stopping a Service” option.

#### 16.4.6 Verifying the Service Status

To verify that the httpd service is running, type the following at a shell prompt:

```
~]# service httpd status
httpd (pid 19014) is running...
```

When the httpd service is started, by default, it reads the configuration from locations that are listed in Table below, “The httpd service configuration files”.

**Table : The httpd service configuration files**

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

Although the default configuration should be suitable for most situations, it is a good idea to become at least familiar with some of the more important configuration options. Note that for any changes to take effect, the web server has to be restarted first. To check the configuration for possible errors, type the following at a shell prompt:

```
~]# service httpdconfigtest
Syntax OK
```

## 16.5 File Server(Samba)

To make the recovery from mistakes easier, it is recommended that you make a copy of the original file before editing it.

Most Linux systems are the part of networks that also run Windows systems. Using Linux **Samba servers**, your Linux and Windows systems can share directories and printers. This is most use full situation where your clients are window native and you want to use the linux security features.

samba rpm is required to configure samba server. check them if not found then install

```
[root@Server ~]# rpm -qa samba*
samba-3.0.25b-0.e15.4
samba-common-3.0.25b-0.e15.4
samba-client-3.0.25b-0.e15.4
[root@Server ~]# _
```

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

#setup Select System service from list

```
[*]portmap
[*]xinetd
[*]smb
```

Now restart xinetd and portmap and smb service

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

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



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

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

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

Create a normal user named vinita

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

Now create /data directory and grant it full permission

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

open /etc/samba/smb.conf main samba configuration files

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

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

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

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

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

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

save file with :wq and exit

Now add vinita user to samba user

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

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

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

if you already have on this service then restart it with `service smb restart` commands.

## 16.6 DHCP Server

Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns TCP/IP information to client machines. Each DHCP client connects to the centrally located DHCP server, which returns the network configuration (including the IP address, gateway, and DNS servers) of that client.

### 16.6.1 Why Use DHCP?

DHCP is useful for automatic configuration of client network interfaces. When configuring the client system, you can choose DHCP instead of specifying an IP address, netmask, gateway, or DNS servers. The client retrieves this information from the DHCP server. DHCP is also useful if you want to change the IP addresses of a large number of systems. Instead of reconfiguring all the systems, you can just edit one configuration file on the server for the new set of IP addresses. If the DNS servers for an organization changes, the changes happen on the DHCP server, not on the DHCP clients. When you restart the network or reboot the clients, the changes go into effect.

If an organization has a functional DHCP server correctly connected to a network, laptops and other mobile computer users can move these devices from office to office.

### 16.6.2 Configuring a DHCPv4 Server

The `dhcp` package contains an Internet Systems Consortium (ISC) DHCP server. First, install the package as the superuser:

```
~]# yum install dhcp
```

Installing the `dhcp` package creates a file, `/etc/dhcp/dhcpd.conf`, which is merely an empty configuration file:

```
~]# cat /etc/dhcp/dhcpd.conf
DHCP Server Configuration file.
see /usr/share/doc/dhcp*/dhcpd.conf.sample
```

The sample configuration file can be found at `/usr/share/doc/dhcp-<version>/dhcpd.conf.sample`. You should use this file to help you configure `/etc/dhcp/dhcpd.conf`, which is explained in detail below.

DHCP also uses the file `/var/lib/dhcpd/dhcpd.leases` to store the client lease database.

### 16.6.3 Configuration File

The first step in configuring a DHCP server is to create the configuration file that stores the network information for the clients. Use this file to declare options and global options for client systems.

The configuration file can contain extra tabs or blank lines for easier formatting. Keywords are case-insensitive and lines beginning with a hash sign (`#`) are considered comments.

There are two types of statements in the configuration file:

- **Parameters** — State how to perform a task, whether to perform a task, or what network configuration options to send to the client.
- **Declarations** — Describe the topology of the network, describe the clients, provide addresses for the clients, or apply a group of parameters to a group of declarations.

The parameters that start with the keyword `option` are referred to as options. These options control DHCP options; whereas, parameters configure values that are not optional or control how the DHCP server behaves.

## 16.7 Installation and Configuration of a SSH server and client

The `openssh-server` RPM package is required to configure a Red Hat Enterprise Linux system as an OpenSSH server. If it is not already installed, install it with `rpm` commands as described in our previous article. After it is

installed, start the service as root with the command `service sshd start`. The system is now an **SSH server** and can accept connections. To configure the server to automatically start the service at boot time, execute the command `chkconfig sshd on` as root. To stop the server, execute the command `service sshd stop`. To verify that the server is running, use the command `service sshd status`.

Configure ssh server

In this example we will configure a ssh server and will invoke connection from client side.

For this example we are using two systems one linux server one linux clients.

To complete these per quest of ssh server Follow [this link](#)

Network configuration in Linux

- A linux server with ip address 192.168.0.254 and hostname Server
- A linux client with ip address 192.168.0.1 and hostname Client1
- Updated /etc/hosts file on both linux system
- Running portmap and xinetd services
- Firewall should be off on server

Three rpm are required to configure ssh server. `openssh-server`, `portmap`, `xinetd` check them if not found then install

```
[root@Server ~]# rpm -qa openssh-server
openssh-server-4.3p2-24.el5
[root@Server ~]# rpm -qa portmap
portmap-4.8-65.2.2.1
[root@Server ~]# rpm -qa xinetd
xinetd-2.3.14-10.el5
[root@Server ~]# _
```

Now check `sshd`, `portmap`, `xinetd` service in system service it should be on

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

n Linux client

ping from ssh server and run ssh command and give root password



```
[root@Client1 ~]# ssh 192.168.0.254
The authenticity of host '192.168.0.254 (192.168.0.254)' can't be established.
RSA key fingerprint is 66:83:74:ed:06:95:15:6c:44:6d:aa:43:ef:87:9e:cf.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.254' (RSA) to the list of known hosts.
root@192.168.0.254's password:
Last login: Sun Feb 14 22:57:07 2010 from Client1
[root@Server ~]# _
```

By default **ssh** command will enable root session. If you want to login from normal user then specify his name with **-l** options.

```
[root@Client1 ~]# ssh 192.168.0.254 -l vinita
vinita@192.168.0.254's password:
Last login: Sun Feb 14 22:57:34 2010 from Client2
[vinita@Server ~]# _
```

Now restart **xinetd** and **portmap** and **sshd** service

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

[root@Server ~]# service sshd restart
Stopping sshd: [OK]
Starting sshd: [OK]
[root@Server ~]# chkconfig sshd on
[root@Server ~]# _
```

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

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

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

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

Create a normal user named vinita

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

With `ssh` you can run any command on server without login (user password require)

```
[root@Client1 ~]# ssh root@192.168.0.254 ls /root
root@192.168.0.254's password:
anaconda-ks.cfg
Desktop
dump
install.log
install.log.syslog
test.sh
[root@Client1 ~]# _
```

## 16.8 Installation and Configuration of a FTP server and client

**Ftp server** is used to transfer files between server and clients. All major operating system supports ftp. ftp is the most used protocol over internet to transfer files. Like most Internet operations, FTP works on a client/ server model. FTP client programs can enable users to transfer files to and from a remote system running an FTP server program.

This article is written for RHEL. Any Linux system can operate as an FTP server. It has to run only the server software—an FTP daemon with the appropriate configuration. Transfers are made between user accounts on client and server systems. A user on the remote system has to log in to an account on a server and can then transfer files to and from that account's directories only. A special kind of user account, named **ftp**, allows any user to log in to it with the username “**anonymous**.” This account has its own set of directories and files that are considered public, available to anyone on the network who wants to download them.

The numerous FTP sites on the Internet are FTP servers supporting FTP user accounts with anonymous login. Any Linux system can be configured to support anonymous FTP access, turning them into network FTP sites. Such sites can work on an intranet or on the Internet.

### Configuring the ftp Server

The **vsftpd** RPM package is required to configure a Red Hat Enterprise Linux system as an ftp server. If it is not already installed, install it with **rpm** commands as described in our pervious article. After it is installed, start the service as root with the command **service vsftpd start** . The system is now an **ftp server** and can accept connections. To configure the server to

automatically start the service at boot time, execute the command **chkconfig vsftpd on** as root. To stop the server, execute the command **service vsftpd stop**. To verify that the server is running, use the command **service vsftpd status**.

Configure vsftpd server

In this example we will configure a **vsftpd** server and will transfer files from client side.

For this example we are using three systems one linux server one linux clients  
Network configuration in Linux

- A linux server with ip address 192.168.0.254 and hostname Server
- A linux client with ip address 192.168.0.1 and hostname Client1
- Updated /etc/hosts file on both linux system
- Running portmap and xinetd services
- Firewall should be off on server

Three rpm are required to configure ssh server. **vsftpd**, **portmap**, **xinetd** check them if not found then install

```
[root@Server ~]# rpm -qa vsftpd
vsftpd-2.0.5-10.el5
[root@Server ~]# rpm -qa portmap
portmap-4.0-65.2.2.1
[root@Server ~]# rpm -qa xinetd
xinetd-2.3.14-10.el5
[root@Server ~]# _
```

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

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

Now restart **xinetd** and **portmap** and **vsftpd** service

```

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

[root@Server ~]# service vsftpd restart
Shutting down vsftpd: [OK]
Starting vsftpd for vsftpd: [OK]
[root@Server ~]# chkconfig vsftpd on
[root@Server ~]# _

```

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

```

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

```

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

```

[root@Server ~]# service portmap status
portmap (pid 3438) is running...
[root@Server ~]# service xinetd status
xinetd (pid 3462) is running...
[root@Server ~]# _

```

Create a normal user named **vinita**

```

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

```

Login for this user on other terminal and create a **test** file

```

[vinita@Server ~]$ cat > test
This is test file created on Linux ftp server
[vinita@Server ~]$ _

```



On Linux client **ping** from **ftp** server and run **ftp** command and give **username** and **password**.

after login you can download files from the specified directories

Most commonly commands used on ftp prompt are

**put** To upload files on server

```
[root@Client1 ~]# ftp 192.168.0.254
Connected to 192.168.0.254.
220 (vsFTPd 2.0.5)
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.0.254:root): vinita
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get test
local: test remote: test
227 Entering Passive Mode (192,168,0,254,240,40)
150 Opening BINARY mode data connection for test (46 bytes).
226 File send OK.
46 bytes received in 0.0026 seconds (17 Kbytes/s)
ftp> quit
221 Goodbye.
[root@Client1 ~]# ls
anaconda-ks.cfg anaconda-ks.cfg.backup install.log install.log.syslog test
[root@Client1 ~]# cat test
This is test file created on Linux ftp server
[root@Client1 ~]# _
```

**get** To download files from server

**mput** To upload all files

**mget** To download all files

**?** To see all available command on ftp prompts

**cd** To change remote directory

**lcd** To change local directory.

```
ftp> ?
Commands may be abbreviated. Commands are:
! delete literal prompt
? debug ls put
append dir ndelete pwd
ascii disconnect ndir quit
bell get nget quote
binary glob nkdir recu
bye hash nls remotehelp
cd help nput rename
close lcd open rmdir
ftp> _
```

## 16.9 Self Learning Exercise

- Q.1 What service is used to translate domain names to IP addresses?
- a) NFS
  - b) SMB
  - c) NIS
  - d) DNS
- Q2. What protocol(s) is(are) allowed a user to retrieve her/his mail from the mail server to her/his mail reader?
- a) POP3
  - b) FTP
  - c) MAP
  - d) All of the above
- Q3. Which of the following server is used with the BIND package?
- a) httpd
  - b) shttp
  - c) dns
  - d) named
- Q4. Which of the following is the main Apache configuration file?
- a) /etc/apachconf
  - b) /etc/httpd/config.ini
  - c) /etc/httpd/conf/httpd.conf
  - d) /etc/srm.conf
- Q5. Which of the following command is used to access an SMB share on a Linux system?
- a) A.NFS
  - b) B.SMD
  - c) C.smbclient
  - d) D.smbserver
- Q.6. Secure shell (SSH) network protocol is used for
- a) secure data communication

- b) remote command-line login
- c) remote command execution
- d) all of the mentioned

Q.7. FTP is built on \_\_\_\_\_ architecture

- a) Client-server
- b) P2P
- c) Both of the mentioned
- d) None of the mentioned

## 16.10 Summary

- A DNS server, or name server, is used to resolve an IP address to a hostname or vice versa. DNS is usually implemented using one or more centralized servers that are authoritative for certain domains. When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the name requested.
- Mail Server :Email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client. SMTP, POP, IMAP protocols used for mailing tasks.
- DHCP Server: DHCP is useful for automatic configuration of client network interfaces. When configuring the client system, you can choose DHCP instead of specifying an IP address, netmask, gateway, or DNS servers. The client retrieves this information from the DHCP server. DHCP is also useful if you want to change the IP addresses of a large number of systems. Instead of reconfiguring all the systems, you can just edit one configuration file on the server for the new set of IP addresses.
- ftp server is used to transfer files between server and clients. All major operating system supports ftp. ftp is the most used protocol over internet to transfer files. Like most Internet operations, FTP works on a client/ server model. FTP client programs can enable users to transfer files to and from a remote system running an FTP server program. Any Linux system can operate as an FTP server.



- **Web Servers** :Linux distributions provide several web servers for use on your system. The primary web server is Apache, which has almost become the standard web server for Linux distributions. It is a very powerful, stable, and fairly easy-to-configure system. Other web servers are also available, such as Tux, which is smaller, but very fast and efficient at handling web data that does not change. Linux distributions provide default configurations for the web servers, making them usable as soon as they are installed.
- **Telnet and FTP** are well-known protocol but they send data in plain text format, which can be captured by someone using another system on the same network, including the Internet. On the other hand, all data transferred using OpenSSH tools is encrypted, making it inherently more secure. The OpenSSH suite of tools includes ssh for securely logging in to a remote system and executing remote commands, scp for encrypting files while transferring them to a remote system, and sftp for secure FTP transfers.

## 16.11 Glossary

**DNS: Domain Name System** :The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network.

**BIND: Berkeley Internet Name Domain**: BIND is open source software that enables you to publish your Domain Name System (DNS) information on the Internet, and to resolve DNS queries for your users.

**HTTP: Hyper Text Transfer Protocol**: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems.

**FTP: File Transfer Protocol**: The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files from a server to a client using the Client–server model on a computer network.

**SMTP: Simple Mail Transfer Protocol**: s a TCP/IP protocol used in sending and receiving e-mail.

**POP: Post Office Protocol**: a protocol used to retrieve e-mail from a mail server.

**DHCP: Dynamic Host Configuration Protocol**: is a standardized network protocol used on Internet Protocol (IP) networks. The DHCP is controlled by a



DHCP server that dynamically distributes network configuration parameters, such as IP addresses, for interfaces and services.

### **16.12 Answers to Self-Learning Exercise**

- |          |         |
|----------|---------|
| Q1. (d)  | Q2. (a) |
| Q3. (c)  | Q4. (c) |
| Q5. (c)  | Q.6.(d) |
| Q.7. (a) |         |

### **16.13 Exercise**

- Q.1. Explain different email Protocols with their functions.
- Q.2. Explain use of Domain Name server in World Wide Web.
- Q.3. Explain difference between HTTP and FTP.
- Q.4. Explain utilization DHCP servers.
- Q.5 .Explain the utilization of SAMBHA Server.

### **References and Suggested Readings**

1. <http://www.basicconfig.com/linuxservers.html>
2. <http://www.tecmint.com/install-openssh-server-in-linux/>
3. Richard Peterson, “Linux: The Complete Reference”, Osborne McGrawHill.