CHAPTER 1

Introduction & History

1.0 OBJECTIVES

Friends, After studying this chapter you will be able to -

- Describe the invention of linux
- Analyze that how linux is still in the development phase and who is working on the project of Linux development.

1.1 INTRODUCTION

Linux is an operating system for PC computers and workstations that now features a fully functional graphical user interface (GUI), just like Windows and the Mac (though more stable). Linux was developed in the early 1990s by Linus Torvalds, along with other programmers around the world. As an operating system, Linux performs many of the same functions as UNIX, Mac, Windows, and Windows NT. However, Linux is distinguished by its power and flexibility. Most PC operating systems, such as Windows, began their development within the confines of small, restricted personal computers, which have only recently become more versatile machines. Such operating systems are constantly being upgraded to keep up with the ever-changing capabilities of PC hardware. Linux, on the other hand, was developed in a different context. Linux is a PC version of the UNIX operating system that has been used for decades on mainframes and minicomputers, and is currently the system of choice for workstations. Linux brings the speed, efficiency, and flexibility of UNIX to your PC, taking advantage of all the capabilities that personal computers can now provide. Along with its UNIX capabilities come powerful networking features, including support for Internet, intranets, Windows, and Apple Talk networking. As a standard, Linux is distributed with fast, efficient, and stable Internet servers, such as the Web, FTP, and Gopher servers, along with domain name, proxy, news, mail, and indexing servers. In other words, Linux has everything you need to set up, support, and maintain a fully functional network. Now, with both Gnome and K Desktop, Linux also provides GUI interfaces with that same level of flexibility and power. Unlike Windows and the Mac, you can choose the interface you want and then customize it further, adding panels, applets, virtual desktops, and menus, all with full drag-and-drop capabilities and Internet-aware tools. On your desktop, a file manager window can access any Internet site, enabling you to display Web pages and download files with a few simple mouse operations. To print a file, simply drag it to a Printer icon.

Linux does all this at a great price. Linux is free, including the network servers and GUI desktops. Unlike the official UNIX operating system, Linux is distributed freely under a GNU (General Public License) as specified by the Free Software Foundation, making it available to anyone who wants to use it. GNU stands for “Gnu’s Not UNIX” and is a project initiated and managed by the Free Software Foundation to provide free software to users, programmers, and developers. Linux is copyrighted, and it is not public domain. However, a GNU public license has much the same effect as being in the public domain.

The GNU public license is designed to ensure Linux remains free and, at the same time, standardized. Only one official Linux exists. Linux is technically the operating system kernel, the core operations. In addition Linux is commonly bundled with an extensive set of software available under the GNU public license, including environments, programming languages, Internet tools, and text editors. People sometimes have the mistaken impression that Linux is somehow less than a professional operating system because it is free. Linux is, in fact, a PC and workstation version of UNIX. Many consider it far more stable and much more powerful than Windows. This power and stability have made Linux an operating system of choice as a network server.

1.2 HISTORY OF LINUX AND UNIX

Since Linux is a version of UNIX, its history naturally begins with UNIX. The story begins in the late 1960s when a concerted effort to develop new operating system techniques occurred.

In 1968, a consortium of researchers from General Electric, AT&T Bell Laboratories, and the Massachusetts Institute of Technology carried out a special operating system research project called MULTICS (Multiplexed Information Computing System). MULTICS incorporated many new concepts in multitasking, file management, and user interaction. In 1969, Kend Thompson, Dennis Ritchie, and the researchers at AT&T Bell Laboratories developed the UNIX operating system, incorporating many of the features of the MULTICS research project. They tailored the system for the needs of a research environment, designing it to run on minicomputers. From its inception, UNIX was an affordable and efficient multiuser and multitasking operating system.

The UNIX system became popular at Bell Labs as more and more researchers started using the system. In 1973, Dennis Ritchie collaborated with Ken Thompson to rewrite the programming code for the UNIX system in the C programming language. Dennis Ritchie, a fellow researcher at Bell Labs, developed the C programming language as a flexible tool for program development. One of the advantages of C is it can directly access the hardware architecture of a computer with a generalized set of programming commands. Up until this time, an operating system had to be specially rewritten in a hardware-specific assembly language for each type of computer. The C programming language allowed Dennis Ritchie and Ken Thompson to write only one version of the UNIX operating system, which could then be compiled by C compilers on different computers. In effect, the UNIX operating system became transportable, able to run on a variety of different computers with little or no reprogramming.

UNIX gradually grew from one person's tailored design to a standard software product distributed by many different vendors, such as Novel and IBM. Initially, UNIX was treated as a research product. The first versions of UNIX were distributed free to the computer science departments of many noted universities. Throughout the 1970s, Bell Labs began issuing official versions of UNIX and licensing the systems to different users. One of these users was the Computer Science department of the University of California, Berkeley.

Berkeley added many new features to the system that later became standard. In 1975, Berkeley released its own version of UNIX, known by its distribution arm, Berkeley Software Distribution (BSD). This BSD version of UNIX became a major contender to the AT&T Bell Labs version. Other independently developed versions of UNIX sprouted up. In 1980, Microsoft developed a PC version of UNIX called Xenix. AT&T developed several versions of UNIX and, in 1983, it released the first commercial version, called System 3. This was later followed by System V, which became a supported commercial software product. You can find more information on UNIX in UNIX: The Complete Reference, written by the UNIX experts at AT&T labs, Kenneth Rosen, Doug Host, James Farber, and Richard Rosinski.

At the same time, the BSD version of UNIX was developing through several releases. In the late 1970s, BSD UNIX became the basis of a research project by the Department of Defense's Advanced Research Projects Agency (DARPA). As a result, in 1983, Berkeley released a powerful version of UNIX called BSD release 4.2. This release included sophisticated file management as well as networking features based on TCP/IP

network protocols-the same protocols now used for the Internet. BSD release 4.2 was widely distributed and adopted by many vendors, such as Sun Microsystems. The proliferation of different versions of UNIX led to a need for a UNIX standard. Software developers had no way of knowing on what versions of UNIX their programs would actually run. In the mid-1980s, two competing standards emerged, one based on the AT&T version of UNIX and the other based on the BSD version. In bookstores today, you can find many different books on UNIX for one or the other version. Some specify System V UNIX, while others focus on BSD UNIX.

AT&T moved UNIX to a new organization, called UNIX System Laboratories, which could focus on developing a standard system, integrating the different major versions of UNIX. In 1991, UNIX System Laboratories developed System V release 4, which incorporated almost all the features found in System V release 3, BSD release 4.3, Sun OS, and Xenix. In response to System V release 4, several other companies, such as IBM and Hewlett-Packard, established the Open Software Foundation (OSF) to create their own standard version of UNIX. Two commercial standard versions of UNIX existed then-the OSF version and System V release 4. In 1993, AT&T sold off its interest in UNIX to Novell. UNIX Systems Laboratories became part of Novell's UNIX Systems Group. Novell issued its own versions of UNIX based on System V release 4, called Unix Ware, designed to interact with Novell's Net-Ware system. UNIX Systems Laboratories is currently owned by the Santa Cruz Operation. With Solaris, Sun has introduced System V release 4 onto its Sun systems. Two competing GUI's for UNIX, called Motif and Open Look, have been superseded by a new desktop standard called the Common Desktop Environment (CDE), which has since been incorporated into Open Motif, an open source version of Motif also for use on Linux. Throughout much of its development, UNIX remained a large and demanding operating system requiring a workstation or minicomputer to be effective. Several versions of UNIX were designed primarily for the workstation environment. Sun-OS was developed for Sun workstations and AIX was designed for IBM workstations. As personal computers became more powerful, however, efforts were made to develop a PC version of UNIX. Xenix and System V/386 are commercial versions of UNIX designed for IBM-compatible PCs. AUX is a UNIX version that runs on the Macintosh. A testament to UNIX's inherent portability is that it can be found on almost any type of computer: workstations, minicomputers, and even supercomputers. This inherent portability made possible an effective PC version of UNIX. Linux was originally designed specifically for Intel-based personal computers. Linux started out as a personal project of a computer science student named Linus Torvalds at the University of Helsinki.

At that time, students were making use of a program called Minix, which highlighted different UNIX features. Minix was created by Professor Andrew Tannebaum and widely distributed over the Internet to students around the world. Linus's intention was to create an effective PC version of UNIX for Minix users. He called it Linux, and in 1991, Linus released version 0.11. Linux was widely distributed over the Internet and, in the following years, other programmers refined and added to it, incorporating most of the applications and features now found in standard UNIX systems. All the major window managers have been ported to Linux. Linux has all the Internet utilities, such as FTP file transfer support, Web browsers, and remote connections with PPP. It also has a full set of program development utilities, such as C++ compilers and debuggers. Given all its features, the Linux operating system remains small, stable, and fast. In its simplest format, Linux can run effectively on only 2MB of memory. Although Linux has developed in the free and open environment of the Internet, it adheres to official UNIX standards. Because of the proliferation of UNIX versions in the previous decades, the Institute of Electrical and Electronics Engineers (IEEE) developed an independent UNIX standard for the American National Standards Institute (ANSI). This new ANSI-standard UNIX is called the Portable Operating System Interface for Computer Environments (POSIX). The standard defines how a UNIX-like system needs to operate, specifying details such as system calls and interfaces. POSIX defines a universal standard to which all UNIX versions must adhere. Most popular versions of UNIX are now POSIX-compliant.

1.3 SUMMARY

Linux is free, including the network servers and GUI desktops. Unlike the official UNIX operating system, Linux is distributed freely under a GNU (General Public License)

as specified by the Free Software Foundation, making it available to anyone who wants to use it. GNU stands for “Gnu’s Not UNIX” and is a project initiated and managed by the Free Software Foundation to provide free software to users, programmers, and developers. Linux is copyrighted, and it is not public domain. However, a GNU public license has much the same effect as being in the public domain. The GNU public license is designed to ensure Linux remains free and, at the same time, standardized. Only one official Linux exists. Linux is technically the operating system kernel, the core operations. In addition Linux is commonly bundled with an extensive set of software available under the GNU public license, including environments, programming languages, Internet tools, and text editors.

1.1-1.3Check your progress

Fill in the blanks

1. .....was treated as a research product.
2. Most popular versions of UNIX are now .....compliant.
3. GNU stands for .....

1.4 CHECK YOUR PROGRESS - ANSWERS

1.1- 1.3

1. UNIX
2. POSIX
3. General Public License Not Unix

1.5 QUESTIONS FOR SELF-STUDY

Q.1 Give introduction of Linux.

Q.2 What is the history of Linux system.

1.6 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

NOTES

Chapter 2

Unix Basic Fundamentals

2.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- explain the directory Structure in Unix
- state different flavors of Unix
- analyze fundamentals of Unix

2.1 INTRODUCTION

What is Unix?

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available, for example, in a telnet session.

Unix was developed by Ken Thompson and Dennis Ritchie in AT & T Lab in 1969. Unix is basically Network Operating System generally used with Virtual Console (CLI). Everything is case sensitive in Unix including Usernames, commands, passwords and filenames.

Types of Unix

There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X.

Here in the School, we use Solaris on our servers and workstations, and Fedora Linux on the servers and desktop PCs.

The Unix operating system

The UNIX operating system is made up of three parts; the kernel, the shell and the programs.

The kernel

The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the file store and communications in response to system calls.

As an illustration of the way that the shell and the kernel work together, suppose a user types **rm myfile** (which has the effect of removing the file **myfile**). The shell searches the file store for the file containing the program **rm**, and then requests the kernel, through system calls, to execute the program **rm** on **myfile**. When the process **rm myfile** has

finished running, the shell then returns the UNIX prompt % to the user, indicating that it is waiting for further commands.

The shell
The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt (% on our systems).

The adept user can customize his/her own shell, and users can use different shells on the same machine. Staff and students in the school have the **tcsh shell** by default.

The tcsh shell has certain features to help the user inputting commands.
Filename Completion - By typing part of the name of a command, filename or directory and pressing the [**Tab**] key, the tcsh shell will complete the rest of the name automatically. If the shell finds more than one name beginning with those letters you have typed, it will beep, prompting you to type a few more letters before pressing the tab key again.

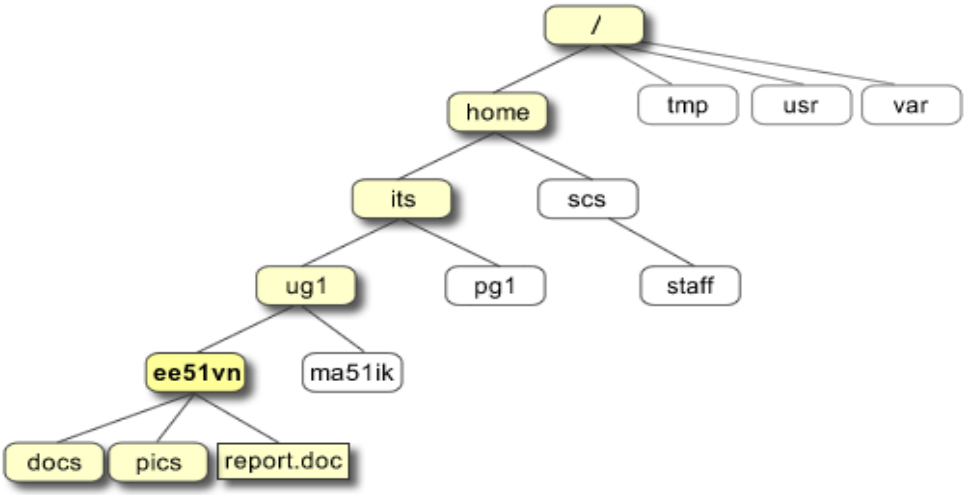
History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

Files and processes
Everything in UNIX is either a file or a process. A process is an executing program identified by a unique PID (process identifier). A file is a collection of data. They are created by users using text editors, running compilers etc.

Examples of files:
- A document (report, essay etc.)
- The text of a program written in some high-level programming language
- Instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
- A directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.

2.2 DIRECTORY STRUCTURE

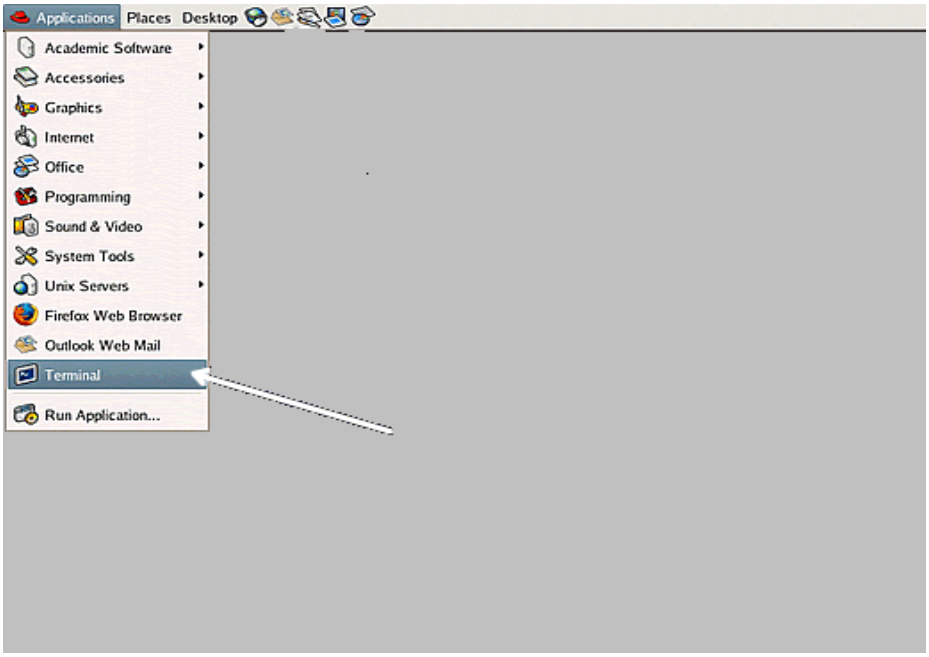
The Directory Structure
All the files are grouped together in the directory structure. The file-system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called **root** (written as a slash /)



In the diagram above, we see that the home directory of the undergraduate student **"ee51vn"** contains two sub-directories (**docs** and **pics**) and a file called **report.doc**. The full path to the file **report.doc** is **"/home/its/ug1/ee51vn/report.doc"**

Starting an UNIX terminal

To open an UNIX terminal window, click on the "Terminal" icon from Applications/ Accessories menu



An UNIX Terminal window will then appear with a % prompt, waiting for you to start entering commands

2.1 & 2.2 Check your progress

A. Fill in the blanks

1] The UNIX operating system is made up of three parts; the .....,the .....and the .....

2] The .....of UNIX is the hub of the operating system
3] The .....acts as an interface between the user and the kernel.
4] The .....is a command line interpreter (CLI).
5] Everything in UNIX is either a .....or a .....

B. State True or False

1] Unix is a Desktop operating system.

2] Unix supports GUI i.e. Graphical User Interface only.

3] The top of the hierarchy is traditionally called **root**. (Written as a slash /)

4] Unix including Usernames, commands, passwords and filenames.

2.3 UNIX FLAVORS

Flavors of UNIX Definition

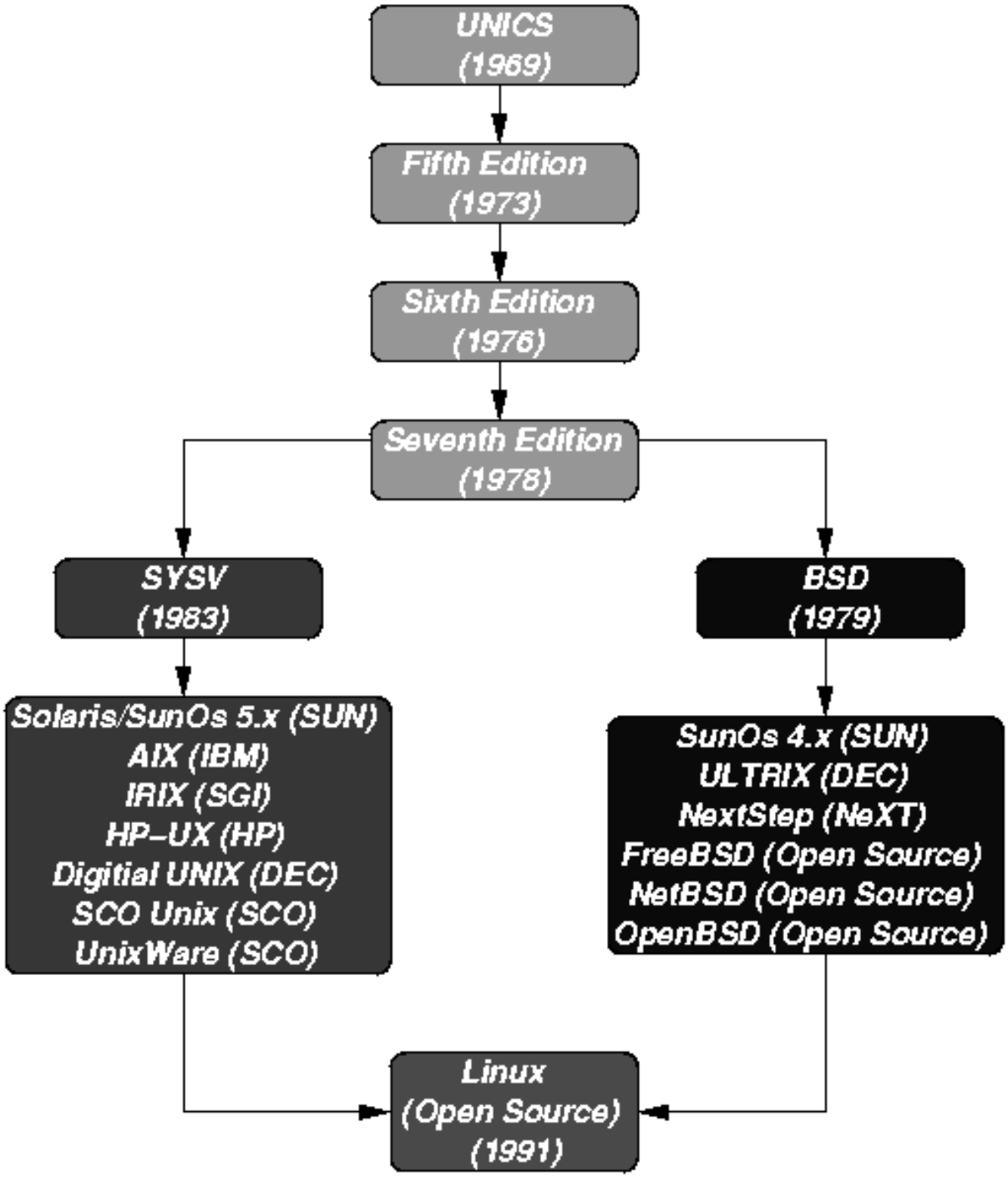
The widely used term *flavors of UNIX* refers to the many Unix-like operating systems that have been developed based on the original UNIX that was written in 1969 by Ken Thompson and Dennis Ritchie at Bell Labs.
Fragmentation of UNIX occurred almost from the beginning. It was the result of both commercial pressures and differences in opinion among developers as to the way in which operating systems should behave.
Among the ways in which the various *flavors* of UNIX differ are (1) fundamental design, (2) commands and features, (3) the hardware platform(s) (i.e., processors) for which they are intended and (4) whether they are proprietary software (i.e., commercial software) or free software (i.e., software that anyone can obtain at no cost and use for any desired purpose).
Many of the proprietary flavors have been designed to run only (or mainly) on proprietary hardware sold by the same company that has developed them. Examples include:
UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities.
What follows here is a simplified history of how UNIX has developed.

2.3 Check your progress

Fill in the blanks

1) Widely used term .....of Unix refers to the many unix -like operating system that have been developed based on original Unix o/s.

2) Unix has been a popular os for more than two decards because of its multiuser & ..... environment with networking capabilities.



Simplified UNIX Family Tree

2.4 UNIX FUNDAMENTALS

UNIX fundamentals

UNIX has been around for a long time (over 30 years). It predates the concept of the personal computer. As such, it was designed from the ground up to be a multi-user, shared, networked operating environment. UNIX has concepts such as **Users**, **Groups**, **Permissions** and **Network-Shared Resources** (such as files, printers, other computer systems, etc.) built-in to the core of its design. This makes UNIX a uniquely powerful and flexible operating system. Along with this power and flexibility comes some unique concepts that make UNIX what it is. These concepts are relatively simple and should be understood to take full advantage of the operating system.

·**Users** - In order to make use of a UNIX system, you must first log in. This requires a **user account**, which consists of:

- **o Username:**
  This is your login name and is how you are identified to the system itself and to other users of the system.
- **o Password:**
  Along with your username, your password grants you access to the system. Don't forget or lose your password. If you write your password down, keep it in a safe place.
- **o Default group:**
  The default group that your username belongs to (see **Groups** below).
- **o Contact info:**
  So that system administrators and other users can contact you if necessary.
- **o Home directory:**
  A directory or "folder" assigned to your username. This grants you access to disk storage. This is where you will keep your files and data.
- **o Default shell:**
  The program which manages your login and command line sessions (covered in detail later)

·**Groups** - A UNIX group is a collection of users - i.e. a list of usernames. Groups provide a mechanism to assign **permissions** (see below) to a list of users all at once. For our purposes, group associations are typically based on which research group or area of study a user is affiliated with. Each user can belong to more than one group.

·**Permissions** - Everything in UNIX is "owned" by both a user and a group. The simplest example of this would be files (but this concept is not limited only to files). By manipulating permissions, the user who owns a file can define which other users and groups can read or modify that file. In this way, users can secure sensitive files from prying eyes and keep themselves or others from accidentially deleting important data.

·**Shared Resources** - UNIX is a networked operating environment at its core. As such, nearly everything that one can access on the local system can also be accessed via the network from remote systems. This includes, among other possibilities, editing and sharing files, running software, or using printers. Even the contents of a UNIX system's display can be manipulated remotely.

The actions that an individual user is able to perform remotely is defined by the permissions assigned to that user (or any group to which the user belongs) for each of these activities. Some of these things will be discussed in greater detail later on.

UNIX Processes

When a program is started on UNIX, it creates what is known as a "process" on the system. Every process is assigned a unique serial number called its **process id** or **PID** for short. Processes can be created by any user, but can only be destroyed by someone with the permissions to do so - usually the user that created the process or the system administrator. This ensures that the compute jobs you start on the system will not be disturbed by any other user of the system until they complete or you decide to stop them yourself.

Processes and process management becomes important on UNIX systems that are shared between a number of users. The concept of users and PIDs is the main tool by which the available system resources are shared fairly among everybody who needs access to them. Processes can be suspended or given lower priority in cases where one

or more users should step out of the way for someone else, but wish to do so without losing their work up to that point.
One further consideration on this topic is the fact that a running UNIX process can spawn "child" processes. For example, any program you run from inside a UNIX shell will be a child process of that shell. Conversely, the shell is the parent process of this child. Child processes have associated with them both their own process id (PID) as well as their parent's process id (PPID).
Normally this concept of parent and child processes is not something you need to be bothered with as a user. However, it can be useful to understand how UNIX organizes processes if you are trying to keep track of certain system resources (e.g. memory and CPU), if you are working with environment variables, or if you need to track down a rogue program or script.

2.4 Check Your Progress

State True or False

1] A Unix Group is collection of users

2] Every process is assigned a common serial number called PID

2.5 SUMMARY

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. The most popular varieties of UNIX are Sun Solaris, GNU/ Linux, and MacOS X. The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the file store and communications in response to system calls. The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The file-system is arranged in a hierarchical structure, like an inverted tree.
The top of the hierarchy is traditionally called **root.** UNIX has been around for a long time (over 30 years). It predates the concept of the personal computer. As such, it was designed from the ground up to be a multi-user, shared, networked operating environment.
UNIX has concepts such as Users, Groups, Permissions and Network-Shared Resources.

Source : *http://structbio.vanderbilt.edu (Link)*

2.6 CHECK YOUR PROGRESS-ANSWERS

2.1 & 2.2

A
1] (Kernel, Shell and programs)
2] (Kernel)
3] (Shell)
4] (Shell)
5] (File or process)

B
1] False
2] False
3] True
4] True

2.3 1] flavors

2] multi-tasking

2.4 1] True 2] False

2.7 QUESTIONS FOR SELF - STUDY

1) What is Unix?

2) List various flavours of unix.

3) Explain unix fundamental in detail.

2.8 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

CHAPTER 3

Concept & Architecture

3.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- explain the concept and the architecture of the Linux Operating System.
- state features of shell and types of shells in linux

3.1 INTRODUCTION

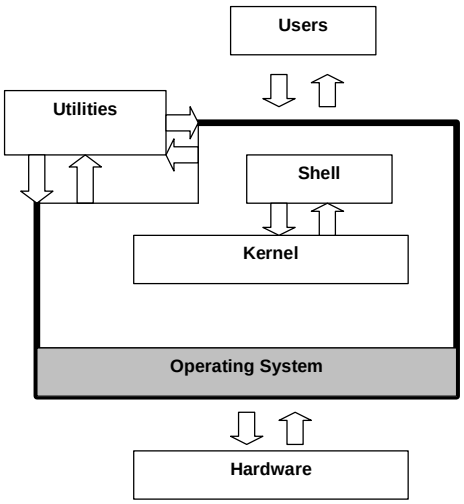
Operating System is a set of programs & functions that provides an access to the system resources including both hardware & software; it acts as a communicator between the user & the computer. These resources include the central processing unit, the file system, monitors or terminals, disk drives, printers & other mass storage devices.

Linux is a multilayered operating system, each layer in the system can be thought of as a country with its own indigenous language. The words operating system are frequently used to refer to the layers kernel, shells & the utilities or commands.

Technically, utilities are not the part of the o.s. Utilities that come with the o.s. are basic tools that have evolved into standard Unix commands. They make the o.s. more immediately useful to the user, but only the Kernel & Shell are truly the Operating System (o.s.).

Conceptually, the Linux o.s. is similar to an onion. It consists of many layers, one above the other. Each layer has its own functionality & interacts with other layers. The layers of the Linux Operating System & their communication with each other can be well explained with the help of following figure.

Figure: The Linux Architecture



3.2 THE LINUX ARCHITECTURE

3.2.1 Operating System Summarization

The hardware can converse with the Kernel. The Kernel is bilingual. It can talk to the hardware or to the shell. The shell is multilingual & can talk to any part of the system with the exception of the hardware.

When the commands are typed by user in the command line interface (CLI), or when utility/application program needs to access the system resources the shell interacts with the kernel & the kernel will interact with hardware.

3.2.2 Overview of Linux Kernel

The Linux Kernel is a free software Unix-like operating system kernel that was begun by the Linus Torvalds in 1991 & subsequently improved with the assistance of developers around the world.

Kernel is the primary part of the Operating System. This manages all the o.s. functions such as input/output management to access the peripheral devices like mouse, keyboard etc, memory management, tasks scheduling, & the devices accessed during device drives etc.

It is the first part of the o.s. to load into memory during the system start up, & remains there for the entire duration of the computer session because its services are required continuously. Thus it is important to be as small as possible to boot the system faster while providing all the essential services needed by the other parts of the o.s. & by the various application programs.

The Kernel itself does not interact directly with the user, but rather interacts with the shell & other programs with the hardware devices on the system, including the CPU, memory & disk drives.

3.2.3 Kernel Space & Users Space

Kernel space is where the 'Kernel' executes & provides its services. User space is that set of memory locations in which user processes run. A process is an executing instance of a program.

3.3 SHELLS IN LINUX

In the family of Linux Operating System the use of a program that serves as an interface between user & operating system is called the Shell. Its is a program that presents an interface to various operating system functions & services. The shell is so called because it is an outer layer of interface between the user & the innards of the operating systems. The shell is a kind of interpreter i.e. a shell is a program which takes user input (e.g. commands which we type) & translates them into instructions that the operating system can understand.

The Operating system shell has two categories :

1) Command Line Interface (CLI)
2) Graphical User Interface (GUI)

A CLI is a method of interacting with a computer by giving it lines of textual commands either from keyboard input or from a script.

A GUI is a method of interacting with a computer through a metaphor of direct manipulation of graphical images & widgets.

3.3.1 Features of Shell

The shell offers the following interactive features :

1) Command History :

The command history buffer stores the commands you enter & let you display them at any time. As a result, you can select a previous command, or parts of previous commands, & then re-execute them. This feature saves time because it lets you reuse long commands instead of retyping them.

2) Command Aliases :

The command aliases feature lets you abbreviate long command lines or rename commands. You do this by creating aliases for long command lines that you frequently use. In addition, aliases let you make up more descriptive names for o.s. commands.
For example, you can define an alias named cls for the clear command.
Syntax :
alias asliasname=command

3) File name completion :

The file name completion feature saves typing by allowing you to enter a portion of the file name, when you press the tab/esc key, the shell will complete the file name for you.

4) Command Line Editing :

This allows you to retrieve a previously entered command & edit it.

3.3.2 Types of Shells

There are many different types of shells. Under Linux bash (the Bourne Again shell) is more commonly used, bash is an extension of the Bourne shell & uses the Bourne shell syntax.
Different shells use slightly different syntax for some commands.

1) Sh or Bourne shell :

The original shell still used on Unix systems & in Unix related environments. This is a basic shell, a small program with basic features. This shell does not support any of the shell features such as history, command line completion & command line editing.

2) Bash or Bourne Again shell :

The standard GNU shell, is intuitive & flexible, this shell is most advisable for beginning users while being at the same time a powerful tool for the advanced & professional users. On Linux, bash is the standard shell for common users. This shell is a so called superset of the Bourne shell. This means that the bash shell is compatible with bourne shell : commands that work in sh, also work in bash. It is the default shell in Linux.

csh or C shell has a syntax that resembles that of the highly popular C programing language & thus preferred by the programmers.

tcsh or Turbo C shell is based on csh but also has programmable file-name completion, command line editing, a history mechanism & other features lacking in csh.

ksh or the Korn shell is a superset of sh. It includes many features of C shell as well, including a command history, which was inspired by the requests of Bell Labs users. It also features built-in arithmetic evaluation & advanced scripting capabilities similar to those found in powerful programming languages such as awk, sed & perl.

Note : The file /etc/shells gives a list of supported shells on a Linux system.
The users default shell is set in the /etc/passwd file.

3.3.3 Basic Features of Linux

1. Multi-user:
   Not only can you have many user accounts on a Linux System, you can also have multiple uses logged in & working on the same system at the same time.

3. Multitasking:
   In Linux,it is possible to have many programs running at the same time,which means that only can you have many programs going at once,but that the Linux operating system itself can have programs running in the background,

3. Hardware Support:
   You can configure support for almost every type of hardware that can be connected to a computer. There is support for floppy disk drives,CD-Roms,Sound Card, Tape Devices, Video Cards etc..

4. Networking Connectivity:
   To connect your Linux system to a network, Linux offers variety of LAN Cards, Modems and serial Devices.

5. Network Servers:
   A variety of software packages are available that enables you to use Linux as a Print Server, File Server, TP Server, Mail Server, Web Server, News Server or Workgroup Server.

3.3.4 Primary Linux Advantages

1) Free Operating System & Open Source:
The Linux Operating System is available at No Cost & anyone can download it from Internet without paying for it. The source code of the operating system is provided with it so that anyone can edit the source code for convenience.

2) Reliable:
As the Linux Operating System can be recovered easily the data loss doesn't happen, that is why it is reliable.

3) Secure:
There is very less possibility of hacking of Linux that is why it s more secure than any other Operating System.

4) Help is available on Internet:
There is always someone available on Internet from Linux Community to help Linux Users.

3.1, 3.2 & 3.3 Check your Progress

Q.1 Fill in the blanks

1. .....acts as a communicator between the user and the computer.
2. Linux project was begun in the year.....
3. .....is the primary part of the Operating System.
4. a program that serves as an interface between user & operating system is called the .....

Q.2 True or False

1. Linux is very costly.
2. Bourne shell has all the features like command line editing, file name completion.
3. A CLI is a method of interacting with a computer through a metaphor of direct manipulation of graphical images & widgets.

3.4 SUMMARY

In these chapters, we learnt about the basic concepts of a Linux Operating System, Architecture of the Operating System, History of Linux, Advantages & Features of the Linux. Also we have seen what is a shell and its features. The available shells in Linux. And how those shells are important for executing number of commands.

3.5 CHECK YOUR PROGRESS- ANSWERS

3.1 ,3.2 ,3.3

1. Operating System	2. 1991	3. Kernel	4.Shell
1. False	2. true	3. False	

3.6 QUESTIONS FOR SELF-STUDY

Q.1 Define Operating System and Explain the Architecture.

Q.2 Define a Shell. Explain Features of a Shell & Shell Types.

Q.3 Explain Primary advantages and Features of Linux.

3.7 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

CHAPTER 4

Installation

4.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- discuss installation of the Linux.
- describe installation modes in Linux.
- analyse post-installation tasks.
- state how to configure network and swap space.

4.1 INTRODUCTION

This chapter will give a brief knowledge of how to install a Linux operating System and the number of methods used to install the Linux Operating System.

4.2 MEDIA

Linux installation can be done using a variety of different media. Each installation method has different pros and cons depending on the environment you have. Here are some examples:

- **Boot disk:** The boot disk or boot floppy is generally not an installation technique by itself. You will use a Linux boot disk in order to launch setup using one of the other media types. These disks are usually provided as floppy images on the cdrom itself along with the proper software to copy them on floppies.

- **CD-Rom:** This is the most common type of installation. To do this, you need to have a system that allows for cdrom booting. You also need a Linux distribution on cd. To start setup, you simply need to insert the cdrom and start the computer. The setup should start automatically. If your system does not allow for Cdrom start up, you can launch the system using a Linux boot setup disk.

- Other methods including Http, FTP, NFS and SMB are generally used as an enterprise solution to deploy servers or workstations. All of these methods are network based and are not necessarily common.

4.2 Check your Progress

Fill in the blanks

1) .....installation can be done using a variefy by of different media.

2) .....is the most common type of installation .

4.3 INSTALLATION MODES

Originally, Linux installation was a painful process which could only be done by a small elite group of users. Now, some distributions are even easier to install than other commercial operating systems.

Once you have launched setup using one media or another, you will be faced with the option to use either a “simple” mode or an “advanced - expert” mode. What this really refers to whether you are going to use a “graphical user interface” mode or a “text” mode. The GUI mode is a more straight forward process, it is a wizard like experience featuring point and click menus. On the other side, the text mode will often give you the opportunity to make a more personalized installation. The downside of a text installation is its harsh nature.

Figure: Installing Linux in text mode

Whichever mode you are going to use, keep in mind that the best instructions are always the ones that come with your specific distribution. Common elements to every distribution generally include:

4.3.1 Setting up the Language

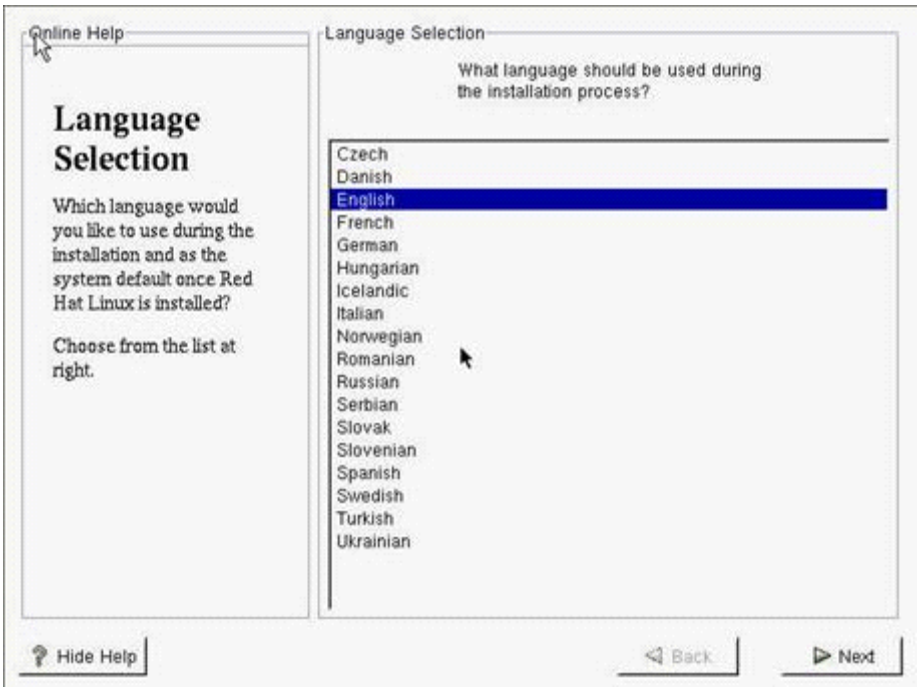


Figure - Choosing your language

4.3.2 Setting up the Keyboard and Mouse

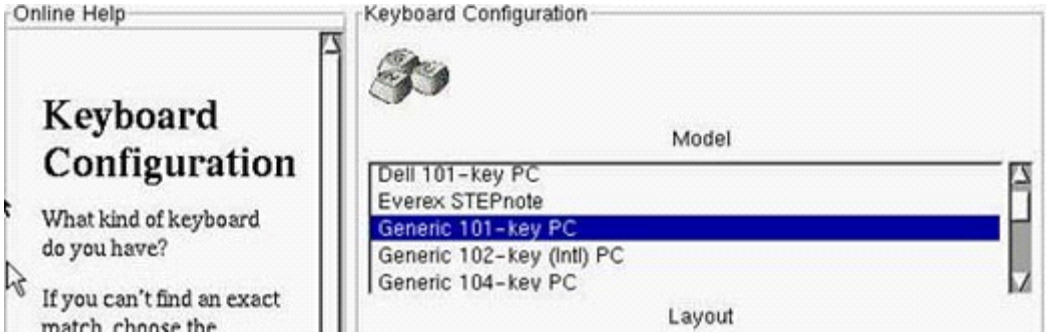


Figure - Configuring the keyboard

You will then get to choose which kind of system you want to build. Depending on your choices, the rest of setup will differ. A workstation setup is generally straightforward and automatic. On some distributions, a workstation installation will generate automatic partitioning and will be easier than a server or custom installation.

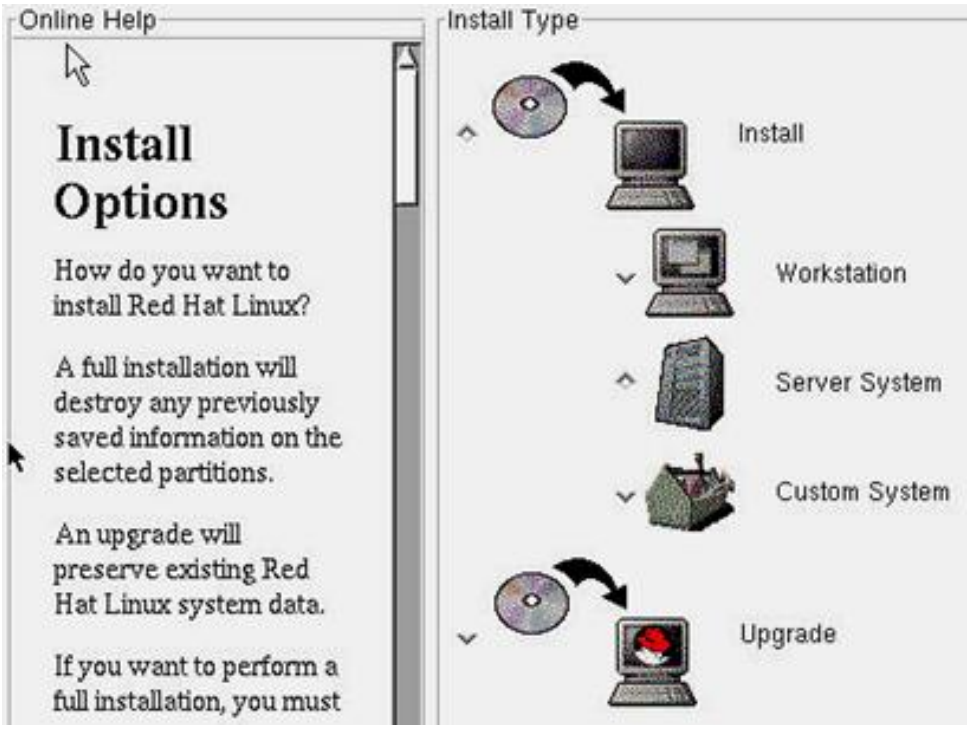


Figure - Choosing what kind of installation should be done

Then, you will get the chance to choose what partitioning scheme you are to use. Automatic partition is the easiest way to go but not the preferred way of doing it. If you remember section 2 (planning the implementation), you might want to customize your partitions for your specific needs.

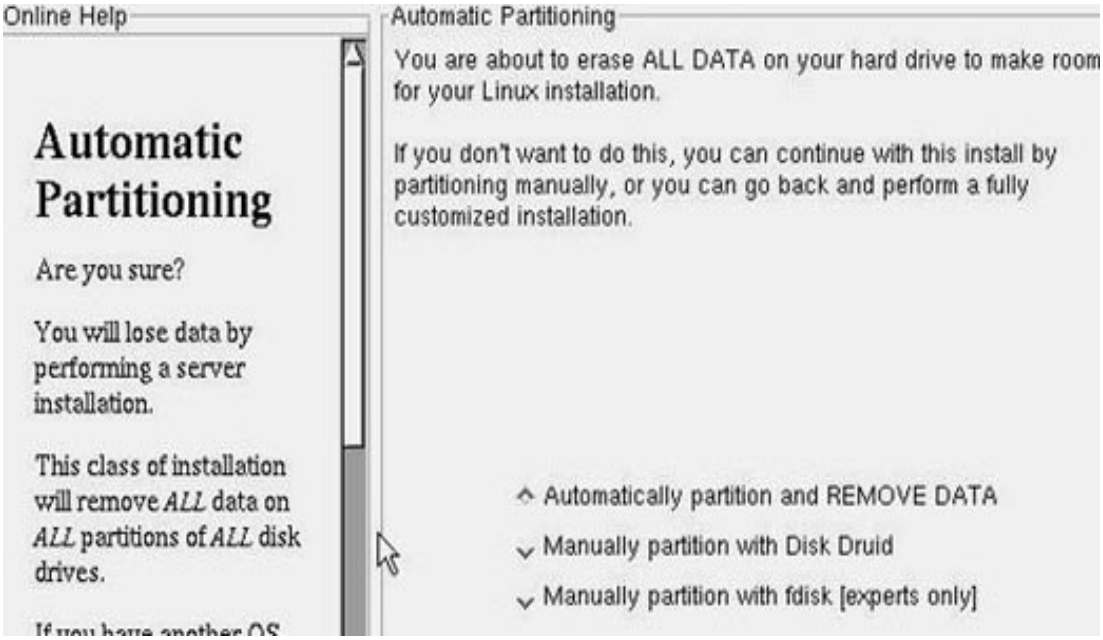


Figure - Choosing the partitioning method

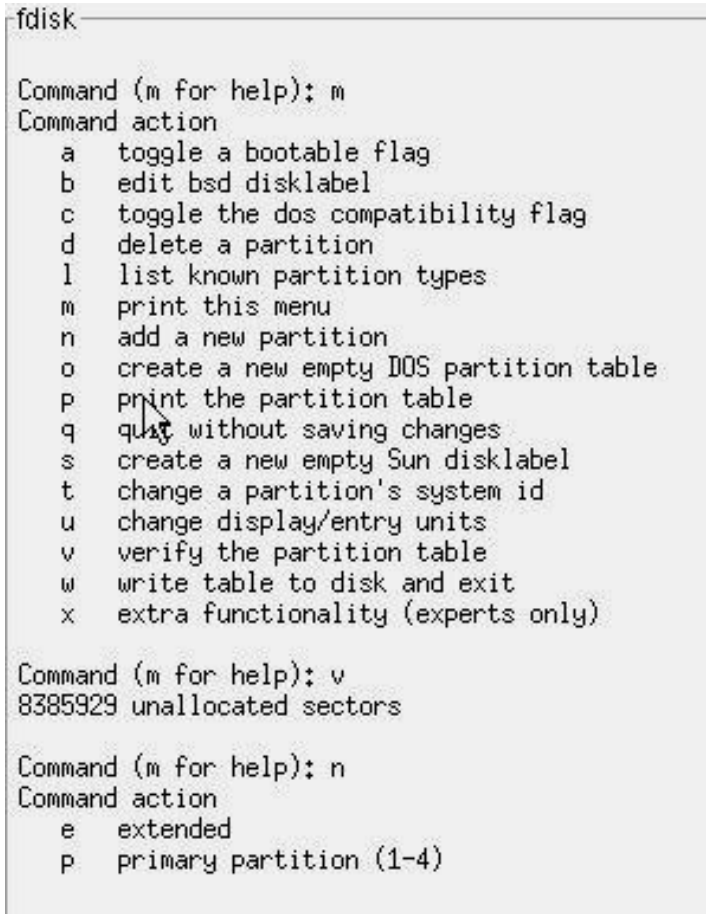


Figure - Using Fdisk to make partitions

The next step is to configure network settings. The ethx on the top is the Ethernet adapter. If your network has a DHCP server, you may want to let the setup to be automatically configured.

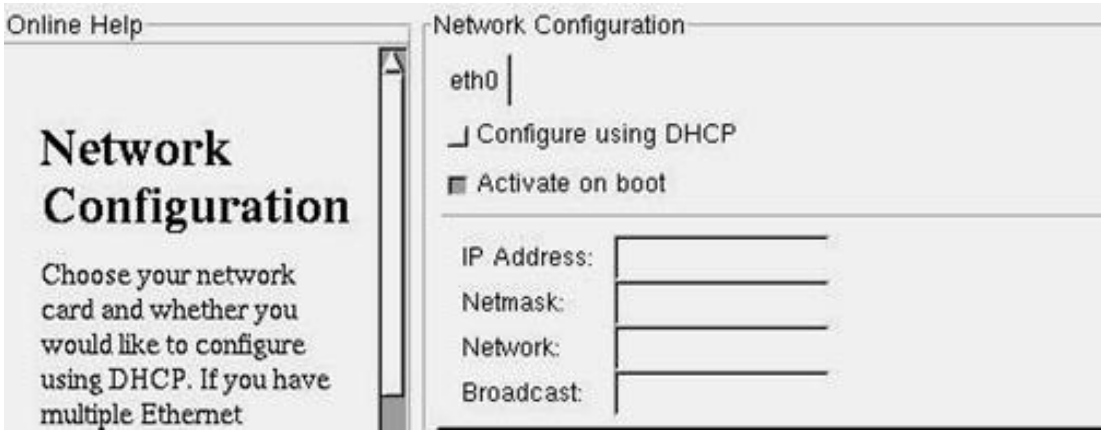


Figure - Configuring Network settings in GUI mode

During setup, you will be prompted to give the root account a password. I suggest you give a strong password as this is the most important account on the system, the one with all the privileges. It is also recommended to create at least one user account

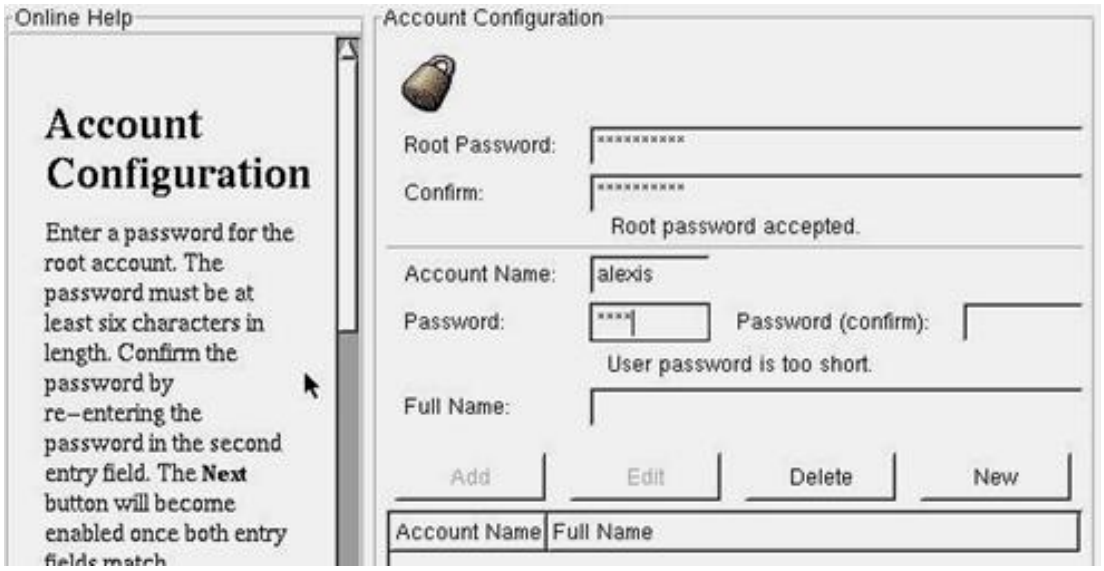


Figure - Creating a user account

If you went through the server or custom setup, you will need to configure the packages you want in order to personalize your installation.

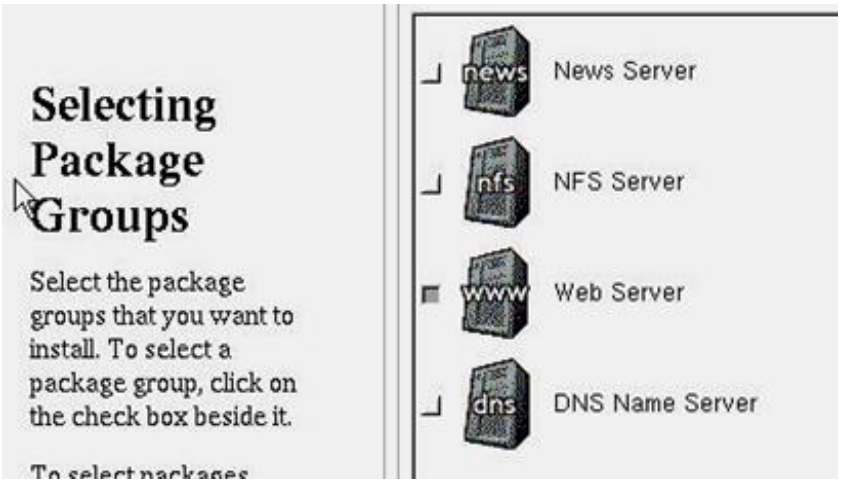


Figure - Configuring the packages for a Web Server

Depending on your installation, you may have to configure the Xfree86 engine. To do this you will have to choose a monitor and configure its vertical and horizontal refresh rate. Choosing a brand name screen will generally ease this step as most manufacturers will be listed.

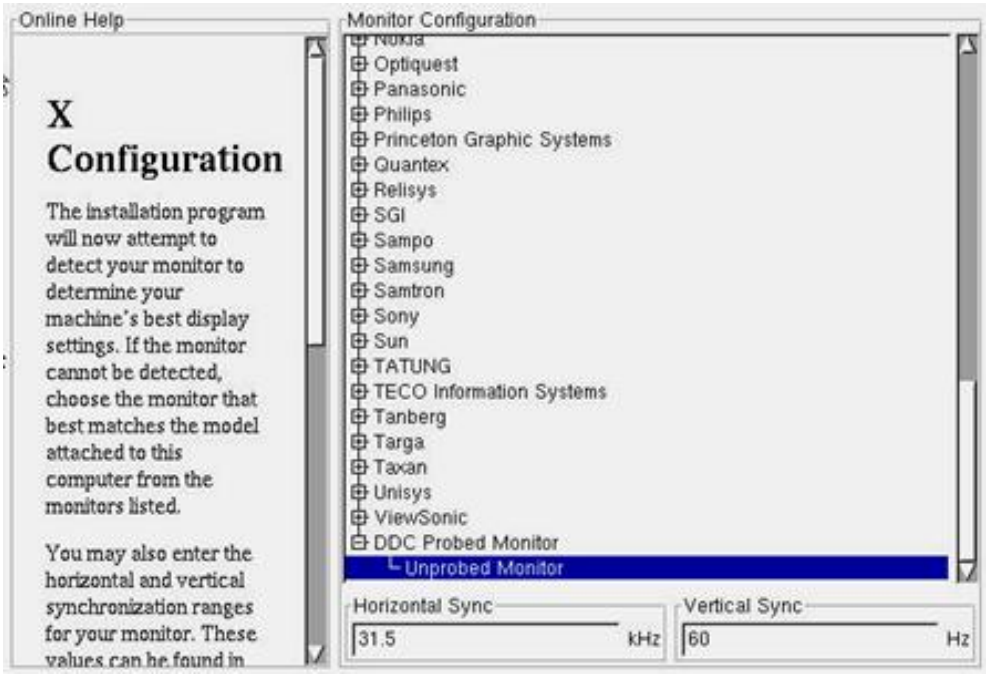


Figure - Configuring a custom monitor with its respected refresh rates

If you chose to install your machine as a workstation, you will most likely need to choose a desktop environment such as KDE or GNOME.

Figure - Choosing your desktop environment

4.3.3 Graphical Interface Startup

In a lot of distributions nowadays, you might be asked during setup to directly boot into the graphical interface. It is strongly recommended not to do so for security and stability reasons.

4.3.4 Post-Installation Tasks

Once the interactive portion of setup is done, the packages will be installed and the kernel will be compiled. Speaking of kernel compilation, it is important that you understand that the Linux Kernel can be compiled at any point after the installation and the reasons for that.
Although the kernel shipped with your distribution is probably very good and stable, you have to understand that it is built to work with most hardware and systems available on the market thus making it full of code that you will probably never use. Therefore recompiling your kernel will enable you to optimize it by picking only what needs to be in it. Other reasons to recompile a kernel will generally include: upgrading your system, doing hardware changes, adding or removing features, etc.
After setup is done, you might also want to take a look at the installation logs to make sure everything went fine. Most distributions will have the following logs:

Location	Description
/var/log	Location of most application logs
/proc/	Hardware information
/etc/rc.d/	Most system initialization, startup and shutdown logs
/etc/syslog.conf	This file contains the name and location of your system log files

4.3.5 Installing More Applications

The way that you install additional applications depends on their format. A.gz application format can be installed using the gunzip. gz command. A .tar application can be installed using the tar –xvf .tar..tar command. These two commands will uncompress the files required for installation. You are likely to go through compilation before the applications work. An .rpm file can be installed using the rpm command.

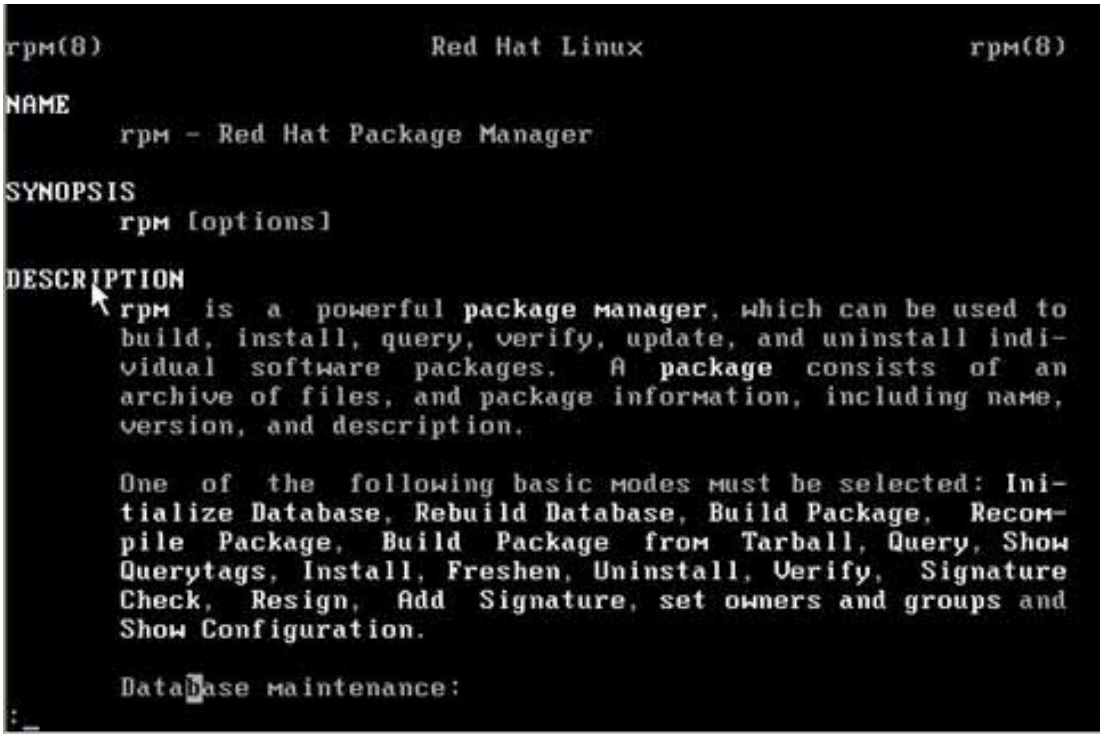


Figure - Man rpm output

The rpm command has a wide variety of parameters and options. Make sure you know how they work before taking the test!

4.3.6 Configuration

Now that your Linux installation is done and verified, let's take a look at further customization. Configuring your Xwindows No matter what desktop environment you chose, it is most likely that it will use the Xwindows architecture. This is why you should know how to reconfigure your Xwindows using automated utilities such as Xconfigurator and XF86Setup.

Figure - Xconfigurator under RedHat

4.3.7 Configuring Networking

Networking, remote access and network clients can be configured using the Linux conf utility. (Simply enter linux conf at the command shell). Specific distributions have optional commands available like Red Hat's Netconfig.

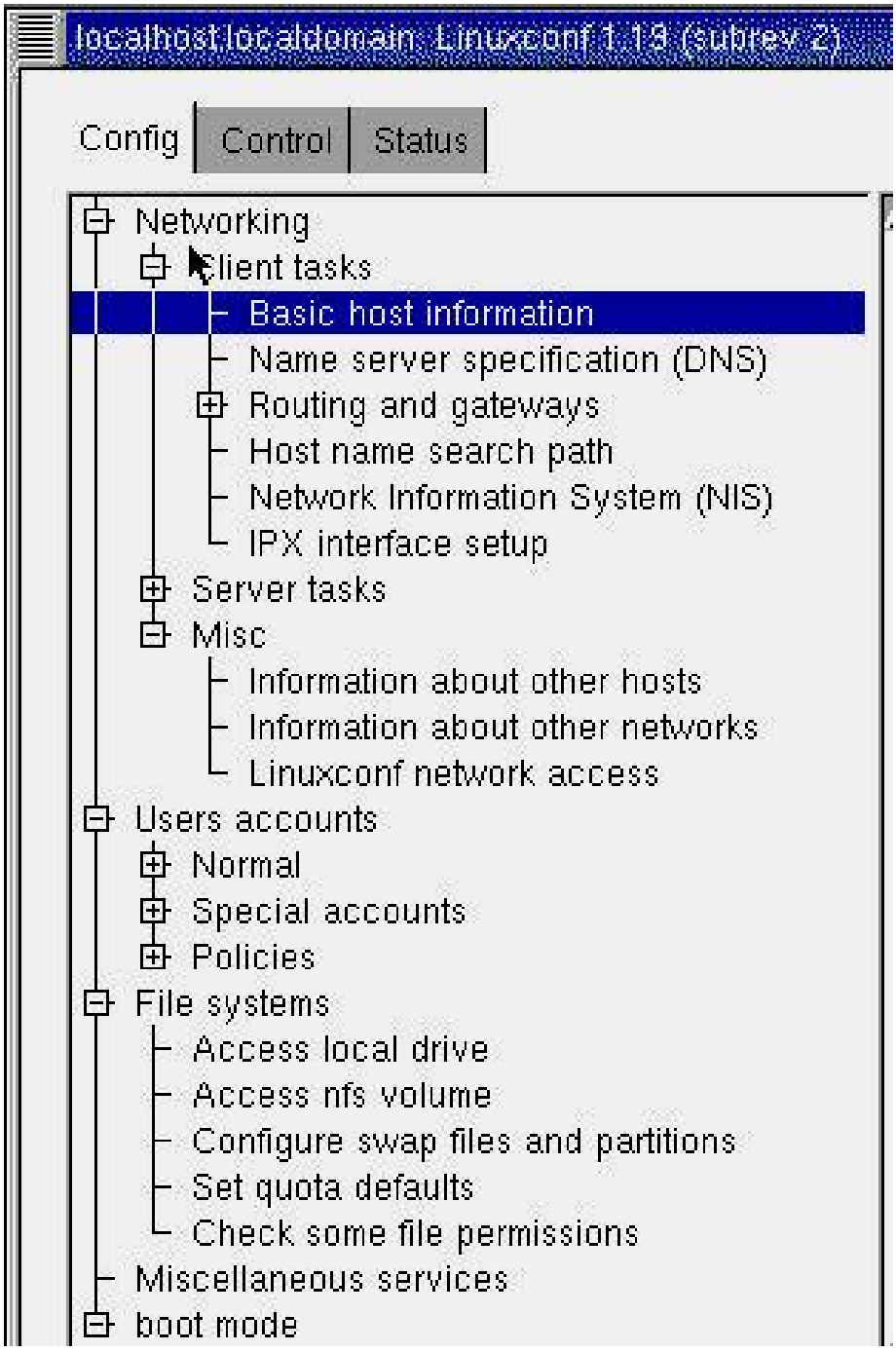


Figure - Using Linux conf in GUI mode

Using Linux conf, you can do most basic configurations, not only networking, including network server related tasks. For example, you can use Linux conf to do basic NFS configurations like in figure 4.3.

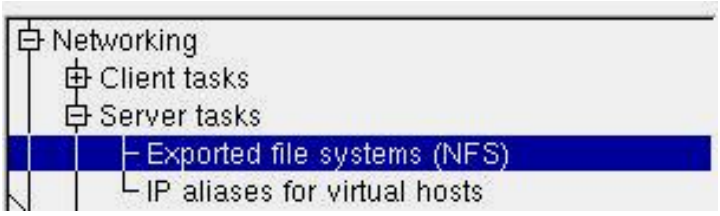


Figure - Configuring NFS with Linuxconf

Depending on your distribution and the version of Linuxconf you have, you should be aware that limited configuration of the following can be accomplished: X, Samba, NIS, NFS, Apache, SMTP, POP, SNMP, FTP, etc. This includes access rights for each of those services.

4.3.8 Configuring the Boot Sector

Linuxconf will also let you modify the way your system boots by changing LILO (the Linux loader)

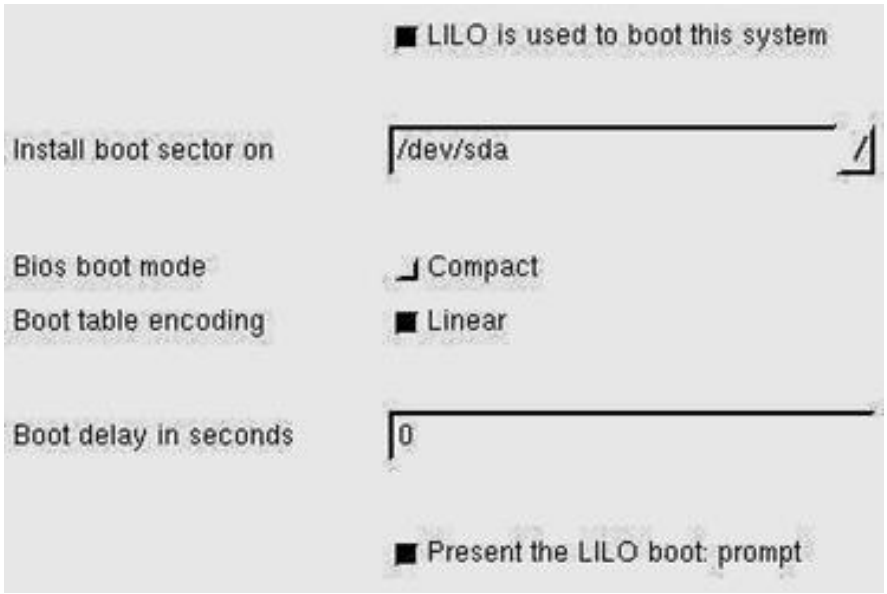


Figure - Changing Lilo with Linuxconf

4.3.9 Configuring Swap Space

In order to work properly, your Linux machine needs some hard disk space to work with. When your system gets heavily loaded, it may become necessary to increase this space. When you add memory, you will need to increase your swap space too. The recommended size of your swap space is the double of the amount of ram memory you have. In order to keep things clean, Linux dedicates a partition for this disk space. It is called the Swap partition. To view your Swap partition, use the cfdisk command.

Figure - Viewing Swap space with cfdisk

You can then use cfdisk to delete and create a bigger swap partition. Once this is done, activate it using mkswap.

4.3 Check your Progress

True or false

1) Once the interactive portion of setup is done the package will be installed and kernel will be compiled.

2) Networking, remote access and network clients can be configured using linux conf utility.

4.4 SUMMARY

In the installation chapter, we learnt about the installation media and the types of installation medias. The installation of Linux in graphical as well as in text method. Configuring the Linux system, configuring networking on a Linux system, as well as configuring file system and swap space management.

Source : *www.mcmcse.com(Link)*

4.5 CHECK YOUR PROGRESS-ANSWERS

4.2 1) Linux 2) CD ROM
4.3 1) True 2) True

4.6 QUESTIONS FOR SELF-STUDY

Q.1 List and explain the installation medias for Linux Operating System.
Q.2 Explain Configuration of Network and Swap Space.

4.7 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

CHAPTER 5

Linux File System Architecture

5.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- describe File System Architecture of the Linux Operating System.
- state basic & essential commands of linux.

5.1 INTRODUCTION

The Linux file system looks like as Inverted Tree Structure. It starts with the root directory, denoted by /, at the top & work from through sub-directories underneath it.

5.2 LINUX FILE SYSTEM ARCHITECTURE

/root

This is the home directory of the system administrator.

/home

Each & every user have their own home directory. For General user, , home directory will be /home/username. For root user, home directory is /root.

/bin

This directory contains several useful commands that are of use to both the system admin as well as non-privileged users.

/boot

This directory contains everything required for the boot process except for the configuration files not needed at the boot time.

/etc

This is the nerve center of the system. It contains all the system related configuration files here or in its sub-directories.

/dev

This is the location of the special files or the device files.

/lib

This directory contains several modules & those shared library images needed to boot the system & run commands in the root file system.

/mnt
This is a generic mount point under which you mount your file system.

/tmp
This directory contains mostly files that are required temporarily.

/opt
This directory is reserved all the software & add-on packages that are not the part of default installation.

/usr
Usually contains by far the largest share of the data on a system. Hence this is one of the most important directory in the system as it contains all the binaries, their documentation, libraries, header files etc. User programs like telnet, ftp, etc are also placed here.

5.3 LINUX BASIC & ESSENTIAL COMMANDS

#pwd
Prints current working directory.

#date
Shows current date along with time.

#cal
Shows calendar for the current month

#cal month year
Shows the calendar for the required year & month.
e.g. #cal 1 2001

#touch
This command is used to create an empty file.
Syntax: touch filename
e.g. touch file1 will create an emty file named file1.

#cat
Displays the contents of the file.
Syntax: cat filename
e.g. cat file1

#cat > filename
Replaces existing text with new text.
e.g. #cat file1

#cat >> filename
Appends the new text at the end of the old one.

#ls [options] [filename or directory name]
#ls –l filename displays extra information about the file.
#ls –R filename displays extra information including sub-directory
#ls –ld filename displays directories & symbolic information.
#ls –a filename displays hidden files.

5.3.1 Copying Files & Directories

#cp
Is used to copy files & directories.
Syntax:
#cp [option] source-file destination-file

#cp –i : copy interactively
#cp –r : copy recursively
#cp –p preserves permissions, ownership, timestamps
#cp –a : combination of –r & -p.

5.3.2 Moving Files & Directories

#mv
This command is used to move or rename the files or directories

Syntax:

#mv [option] sourcefile destinationfile

#mkdir
This command is used to create a directory.
Syntax:
#mkdir directory name

#rmdir
This command is used to remove empty directory only.

Syntax:

#rmdir directory name

#rm directory name
This command is used to remove the directories.
Syntax:

#rm –r dir-name : used to remove directories recursively
#rm –i dir-name : used to remove directories interactively
#rm –f dir-name : used to remove directories forcefully.

#more filename
This command is used to display whole contents of a file at a time.
e.g. #more /etc/passwd

#less filename
This command is used to display one page at a time.
e.g. #less /etc/passwd

#head filename
This command is used to display first 10 lines of a file.
e.g. 1) head /etc/passwd
2) head -n 3 /etc/passwd

#tail filename
This command is used to display last 10 lines of a file.
e.g. 1) tail /etc/passwd
2) tail -n 5 /etc/passwd

#chown filename username
The command is used to change the owner of a file.
e.g. #chown user1 file1 :- will change the ownership of the file1 such that the new owner of the file will be user1.

#wc(word count)
Syntax: wc [option] filename
wc -l filename – displays the number of lines.

wc -w filename – displays the number of words.
wc -c files name – displays the number of bytes.

#diff file1 file2
Compares the 2 files for differences.

#cmp file1 file2
will work same as the diff command

#history
Displays the commands previously used.

#history 8
Displays the last 8 commands used.

#!!
Run previous command.

#! 100
Run 100th command shown in the history.

#sort filename
Sort or arranges a file in alphabetical order.

#sort [option] filename
e.g. #sort -r filename – will sort the file in reverse order of alphabets.
#sort -n filename – will perform a numeric sort.
#sort -u filename - will remove duplicate lines.

#shutdown [OPTION]... TIME
This command is used to shutdown the system.
e.g. 1) shutdown -h 5 :- the system will be shut down after 5min
2) shutdown -r 5 :- the system will rebooted after 5min

#init 0
The system will be shutdown immediately.

#init 6
The system will be rebooted immediately.

#poweroff
The system will be halted.

#who
Displays the users currently logged in to the system.

who m i
Displays who you are from all logged in users.

5.2 & 5.3 Check your Progress

Fill in the blanks

1. .....is the home directory of the system administrator..
2. .....directory contains the useful commands.
3. .....is the nerve center of the System.
4. The command to rename the files or directories is .....
5. .....command is used to remove empty directory only.
6. .....command is used to display whole contents of a file at a time.

5.4 SUMMARY

In this chapter, we learnt about directory structure in Linux and we studied some Linux essential commands to start with such as commands to create a file, a directory,

commands to remove files & directories, also the commands to rename the files & directories. Also we learnt about the history command and how to use the history command as well.

5.5 CHECK YOUR PROGRESS- ANSWERS

5.2 & 5.3

1. /root
2. /bin
3. /etc
4. mv
5. rmdir
6. more

5.6 QUESTIONS FOR SELF-STUDY

Q.1 Explain the directory structure in Linux.

Q.2 List and explain the command to copy & move files.

Q.3 Explain the history command & the options to use the history command.

Q.4 List all the commands with their detailed options to move files & directories

5.7 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

CHAPTER 6

The VIM Editor

6.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- explain the use of VIM editor in linux
- state common "Redirection operators " in linux

6.1 INTRODUCTION

VIM is the standard text editor in Linux. There are some advantages of VIM editor, such as: Speed, Simplicity, Availability. The a command will place you in insert made, allowing you to append the cursor. The letter a will not appear in your document, but all other characters that you type will appear, until you exit insert mode. To exit insert mode, hit the <ESC> key. <ESC> takes you from insert mode back to command mode. Use p or P to put (paste) copied or deleted data. Linux provides three I/O channels to processes: Standard input, Standard output, Standard error.

6.2 THE VIM EDITOR

VIM is the standard text editor in Linux.
There are some advantages of VIM editor, such as:
1) Speed : Do more with fewer key-strokes.

2) Simplicity : No dependence o mouse/GUI
3) Availability: Included with most Unix like OSes

However, there are some disadvantages that you will discover, such as :

1) Difficulty: Steeper learning curve than simpler editors.
2) Key Binding emphasize speed over intuitiveness.

VIM is used to edit, change & save the contents of a file.

There are three main modes with VIM:

1) Command Mode : The default mode. Allows keys to be used for Cursor Movement, copying, Cutting & Pasting Text.

2) Insert Mode: Allows key strokes to be inserted into the document i.e. Modify Text

3) Exit Mode: Allows keys to be used to save, save as & quit the document.

6.2.1 Cursor Movement

h	left
w	word ahead
j	down
b	word back
k	up
l	right
(	sentence back
)	sentence forward
{	paragraph above
}	paragraph bellow.

The I, j, k and l movements move the cursor left, down, up and right respectively.

6.2.2 Entering Insert Mode

a	append after the cursor
I	insert before the cursor
o	open a line bellow
A	append to send end of the file,
I	insert the beginning of the line.
O	open a line above

The a command will place you in insert made, allowing you to append the cursor. The letter a will not appear in your document, but all other characters that you type will appear, until you exit insert mode. To exit insert mode, hit the <ESC> key.

The I command is similar to the a command, but you will insert your data before the cursor. Again, exit insert mode by hitting the <ESC> key.

The o command opens a line bellow the current line, placing you in insert mode.

6.2.3 Leaving Insert Mode : <ESC>

<ESC> takes you from insert mode back to command mode.
Hint: when in trouble, press <ESC> and then press <ESC> again.

6.2.4 Change, Delete and Yank

	Change	Delete	Yank(copy)
Line	cc	dd	yy
Letter	cl	dl	yl
Word	cw	dw	yw
Sentence ahead	c)	d)	y)
Sentence behind	c(	d(	y(
Paragraph above	c{	d{	y{
Paragraph below	c}	d}	y}

The cc command will change a line: it will delete the current line and place you into insert mode to enter the replacement, which could be another line, several lines, or just a few characters. As before, press <ESC> to return to command mode.
The cl command will delete the current character and place you in insert mode; the cw command will delete the current word and place you into insert mode. The paragraph and sentence commands will operate similarly.
The dd deletes the current line, leaving you in command mode. The remaining deletion commands operate in a similar fashion.
The yy command may take some explanation. These days, the common name for this action is “copy”; that is, place some text in a buffer without modifying the original data.

6.2.5 Put (Paste)

Use p or P to put (paste) copied or deleted data.

For line oriented data:
p puts the data bellow the current line
P puts the data above the current line.

For character oriented data:
p puts the data after the cursor
P puts the data before the cursor.

The command to put data from a buffer into the document is p and P. How these work depend on the nature of the data in the buffer. If the data is the oriented (you yanked a line or a paragraph), the p command will open a new line below the current line and place the data there; the P command will open a new line above the current line and place the data there.

6.2.6 Unloading Changes

u	undo most recent change
U	undo all changes to the current line since the cursor landed on the line.
<ctrl-r>	redo last “undone” change.

To undo all successive changes to current line, use the U command.
To redo a change undone by a u command, use the <ctrl-r> command.

6.2.7 Searching for Text

/text	search downwards for “text”
?text	search upwards for text
n	continue search in the same direction.
N	continue search in the opposite direction.
/dog	search downwards in the file for the string “dog”
?mouse	search upwards in the file for string “mouse”

6.2.8 Command-Mode Tricks

dtc delete from cursor to the letter c (does not span lines)
5dd delete five lines (a number can precede any of the two character change,
delete yang or put commands.
x delete a character
rc replace a character with c
R replace character-for-character until <esc> .

5dd delete five lines. Any number can precede any of the change, delete,yank, put commands. Thus 3y} will yank three paragraphs below the current line and 2p will put the three paragraphs back into the document twice, for a total of six extra paragraphs.
R replace one character for every character typed until the <ESC> character is entered.

6.2.9 Saving and Exiting: ex mode

	Save changes	**Abandon changes**
Exit	:wq	:q or :q!
Do not exit	:w	:e!

Forcing changes:

	Save changes
Exit	:wq!
Do not exit	:w!

The traditional method for exiting a vi or vim session is with ex command:
:wq
This will write out the file and quit from the editing session. To write out changes but remain in the editor, run the ex command:
:w
Similarly, to quit an editing session in which you have made no changes.
:q
This command will fail to quit if you have changes that you not committed. To abandon changes, run:
:q!
Abandon changes, staying in the editor, through :e! and force a write through :w! . Finally, a force write and quit with :w! . This last command will forcibly write the file, but only quit if the forcible write was successful.

6.3 STANDARD INPUT AND OUTPUT

Linux provides three I/O channels to processes
* Standard input – keyboard is default
* Standard output – terminal window is default
* Standard error – terminal window is default.

One of the most important features of Linux (and UNIX) is the streaming nature of data known as standard input, standard output and standard error. In general, this allows the

addition, the output from one command can be fed directly to the input of other command.

Standard output, by default, is the screen or terminal window. Most commands send their output to standard output without explicitly being told to do so. When standard output is sent to a destination other than the screen, such as a file, one is redirecting standard output.

A third data stream is standard error. This is a secondary output stream that carries warnings, usage messages, error message and other “out-of-band” information, in an output stream distinct from standard output. This error stream is normally sent to the screen, but may be redirected elsewhere.

These streams are abbreviated as stdin (called file descriptor number 0), stdout (file descriptor number 1), stderr (file descriptor number 2). The fact that there are two output channels allows separation of error messages from normal output. For example, error messages are could be saved in a file with the normal output going to the monitor.

6.3.1 Redirecting Input and Output

Standard Input, Output and Error can be reconnected to alternate locations.

The standard output of commands, witch ordinarily displays on the terminal, can be
redirected into a file or piped into another command.

Standard error, which also ordinarily displays on terminal, can be redirected into a file.
Although it is also possible to pipe standard error into a file using some fairly complex Syntax, this is generally not done.

Standard input, ordinarily coming from the keyboard, can be redirected from a file.
More commonly, the standard output of one command can be piped into the standard input of another command.

6.3.2 Common Redirection Operators

> command > file	Directs standard output of command to file.
>> command >> file	Appends standard output of command to file.
< command < file	command receives its input from file.
2> command 2 > file	Error messages from command are directed to file.

6.3.3 Piping

I command1 I command2
Pipes the standard output of command1 into the Standard input of command2.

6.3.4 Redirecting Output

* In order to study redirecting standard output and error, we will use the find command.
  find / etc -name passwd
* This command will search for all files named passwd in / etc and its subdirectories.
* By default both the standard output and standard error are displayed on the screen.

Example output

```
find / etc - name passwd
/etc/passwd

find: /etc/default: permission denied
```

/etc/pam.d/passwd
Ex.
[bob@station2 tmpl $ find / etc -name passwd > findresult
/ etc / passwd
/ etc / pam.d / passwd

6.3.5 Overwriting or Appending

Redirect output and creative / overwrite the output file

[bob@station2 tmpl $ find / etc -name passwd > output
find: / etc / default : permission denied

Redirect output and append the output file

[bob@station2 tmpl $ find / etc -name passwd >> output
find: / etc / default : permission denied
find: / etc / cups / certs: permission denied

View the output file

[bob@station2 tmpl $ cat output
/ etc / passwd
/ etc / pam.d / passwd
/ etc / passwd
/ etc / pam.d / passwd

Redirect output and overwrite the output file

[bob@station2 tmpl $ find / etc -name passwd > output

Find: /etc / default : permission denied

View the output file

[bob@station2 tmpl $ cat output
/ etc / pam.d / passwd

6.3.6 Redirecting Standard Error

Redirect stderr to the finderrors file

[bob@station2 tmpl $ find / etc -name passwd 2> finderrors
/ etc / passwd
/ etc / pam.d / passwd

View the finderrors file
[bob@station2 tmpl $ cat finderrors
find : / etc / default : permissions denied
find : / etc / cups / certs : permission denied

Redirect some more stderr to the finderrrors file

find / tmp – name passwd 2 >> finderros
View the finderrors file
[bob@station2 tmpl $ cat finderrors
find : / etc default : permission denied
find : / tmp / orbit-root : permission denied
Redirect stderr to the errs file and redirect stdout to the results file

[bob@station2 tmpl $ find / etc -name passwd 2> errs > results
View the errs file

[bob@station2 tmpl $ cat errs
find : / etc / default : permission denied

View the results file

[bob@station2 tmpl $ cat results
/ etc / passwd
/ etc pam.d / passwd

Redirect bothstderr and stdout to the all output file

[bob@station2 tmpl $ find / etc -name passwd > alloutput 2> &1

View the alloutput file

[bob@station2 tmpl $ cat alloutput
/ etc / passwd
find : / etc / default : permission denied
/ etc / pam.d / passwd

* Before using this command create file textfile.txt with all data with capital letters.

Example output
[bob@station2 tmpl $ cat textfile.txt

TEXT TYPED IN ALL CAPITALS IS OFTEN DIFFICULT TO READ
TRANSLATING SUCH TEXT INTO LOWER CASE CAN IMPROVE
ITS LEGIBILITY.

[bob@station2 tmpl $ tr 'A-Z' 'a – z' < textfile.txt
Text typed in all capitals is often difficult to read
Translating such text into lower case improve
Its legibility.

(Do not use quotes)

6.2 & 6.3 Check Your Progress

Fill in the blanks

1. .....append after the cursor.
2. .....opens a line bellow.
3. .....takes you from insert mode back to command mode.
4. .....is used to delete the current line.
5. The command to put data from a buffer into the document is ..... and .....
6. .....undo most recent change.
7. .....undo all changes to the current line since the cursor landed on the line.
8. .....search downwards for "text".

9. .....continue search in the opposite direction.

10. The traditional method for exiting a vi or vim session is with ex command is .....

11. .....keyboard is default.

12. .....terminal window is default.

13. .....terminal window is default.

6.4 SUMMARY

In the above chapter, we learnt how to use the VIM editor, the advantages of using the VIM editor. The chapter gives a detailed explanation & study of the VIM editor, all the shortcut keys to copy, delete & paste the characters or lines of characters without the use of mouse. This is the specialty of the VIM editor.

Then we studied about one of the most important features of Linux (and UNIX) is the streaming nature of data known as standard input, standard output and standard error.

6.5 CHECK YOUR PROGRESS- ANSWERS

6.2 & 6.3

1. a
2. o
3. <ESC>
4. dd
5. p, P
6. u
7. U
8. /text
9. N
10. :wq
11. Standard input
12. Standard output
13. Standard error

6.6 QUESTIONS FOR SELF-STUDY

Q1. List & explain the advantages, disadvantages & modes of the VIM editor.

Q2. Explain how to change, delete & yank the text in detail.

Q3. Explain how to paste, unload changes & search for text in detail.

Q4. List some common redirection operators & explain redirecting input & output.

Q5. Explain redirecting standard error.

6.7 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

CHAPTER 7

User & Group Administration

7.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- describe group administration in linux.
- expalin user administration in linux.
- state some important commands for user administration and group administraion

7.1 INTRODUCTION

The command-line utility useradd provides a simple method for adding new users to the system. When user accounts are created, a private group is also created with the same name. The user may belong to other groups and switch to one with the newgrp command and/or change the file's group ownership with chgrp. This starts the group IDs at 101 and will increase them up to GID 499.gpasswd can be used to define group members in /etc/group, group administration, & to create or change group password in /etc/gshadow, if desired. The su Command is used to switch to another account from the command line.

7.2 USER ADMINISTRATION

7.2.1 Adding a New User Account

- Most common method is useradd:
  #useradd [options] username

- Set account password using passwd

The command-line utility useradd provides a simple method for adding new users to the system:

For example,

#useradd joshua

The above command will add a new account to the machine called Joshua as well as set up that user's home directory, and create a private group for the user also called joshua. The next step would be assign joshua a password which you can do by simply typing the following command :

#passwd joshua

When you need to add several users, you can use the newusers command. Create a file, formatted like /etc/passwd, that contains the usernames and plain text passwords of the users.The command: newusers yourfile will create those users. One apparent drawback of using this method is that the users home directories do not get populated with the files from /etc/skel/.

cat /etc/passwd
alice:password:500:500:Alice:/home/allice:/bin/bash
bob:pa55wOrd:501:Bob:/home/bob:/bin/bash

newusers myusers

If you are using MD5 passwords(the default),ensure that newusers will generate MD5 passwords,else the users will not be able to login!

/etc/login.defs needs an entry like this:

MD5_CRYPT_ENAB yes

7.2.2 User Private Groups

When user accounts are created, a private group is also created with the same name
Users are assigned to this private group
User's new files affiliated with this group

The user may belong to other groups and switch to one with the newgrp command and/ or change the file's group ownership with chgrp:

 # chgrp groupname filename

7.2.3 Modifying/Deleting User Accounts

- To change fields in user's /etc/passwd entry you can :
  - Edit the file by hand
  - Use usermod [options] username

- To remove a user either:
  Manually remove the user from /etc/passwd,/etc/shadow,etc/group,/etc/ gshadow,/var/spool/mail, etc.

  Use userdel [- r] username

7.3 GROUP ADMINISTRATION

7.3.1 Creating & Modifying Groups

- Entries added to /etc/group and /etc /gshadow

- groupadd
- groupmod
- groupdel

Explanation:

- New groups may be created by hand-editing the file /etc/group or by using groupadd. The basic syntax for groupadd is very simple:
  groupadd groupname
- groupdel is used in a similar fashion to remove groups:groupdel groupname

- groupmod can be used, among other things to change the name of a group:groupmod – n newname oldname

For example,if several users are members of the employee group and a root user issues the following command,the group will be changed to staff and all the same members will remain:

#groupmod –n staff employee

The syntax of /etc / group is as follows, one group per line:

gurus:x:501:Joshua,dax,bryan,chris,heather,jon

The first field is the name of the group.The second field is the group password, or an "x" when using shadow passwords.The third field is the unique group ID.The fourth field is a comma-separated list of group members.

In order to avoid using a GID within the range typically assigned to users and their private groups,use the –r option:

#groupadd –r groupname

This starts the group IDs at 101 and will increase them up to GID 499..
gpasswd can be used to define group members in /etc/group, group administration, & to create or change group password in /etc/gshadow, if desired.

7.3.2 Some Important Commands for User Administration

#usermod –l login-name username

#usermod –u uid username

#usermod g gid username

#usermod –c comment username -> this is often the users full name

#usermod -L username -> lock the user account

#usermod –U username -> unlock the user account

7.4 SWITCHING ACCOUNTS

#su

The su command is used to switch to another account from the command line. This command is most often used by system administrators to temporarily become the root user without logging out of their non-privileged account. It is very important to only use the root account when absolutely necessary, because of the awesome power of the root account. Day to day use of the system should never be conducted with the root account.

Syntax:

- su [-] [user]
- su [-] [user] – c command
- Allows the user to temporarily become another user Default user is root
- The "-" option makes the new shell a login shell.

7.1- 7.4 Check your Progress

Q.1 Fill in the blanks

1. .....is the command used to add a new user account.
2. .....is the command used to set the password for the user.
3. .....is the command used to delete a user account.
4. .....is the command used to add a new group to the system.
5. .....is the command used to delete the group from the system.
6. .....is the command used to add a new group with the group ID's in the range of 101 to 499.
7. .....is the command used to lock the user account.
8. .....is the command used to switch to another account from the command line.

7.5 SUMMARY

In the chapter above, studied about the user administration & the group administration in detail so that you can be the administrator of your Linux system & do the changes in the system as per the requirement. Also we studied how to switch from one user login to another without logging out from the first logged in user account.

7.6 CHECK YOUR PROGRESS- ANSWERS

7.1 - 7.4

1. useradd	2. passwd
3. userdel	4. groupadd
5. groupdel	6. groupadd –r
7. usermod –L	8. su

7.7 QUESTIONS FOR SELF-STUDY

Q.1 Explain user administration in detail.
Q.2 Explain group administration in detail.
Q.3 Explain how to switch from the user accounts.

7.8 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

NOTES

CHAPTER 8

Files, Directories & Special Permissions

8.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- explain files and directories permissions in Linux
- state some special permissions like sticky bit SUID and SGID.

8.1 FILES & DIRECTORIES PERMISSION IN LINUX

8.1.1 Permission Types

Four symbols are used when displaying permissions:

- r: permission to read file or list a directory's contents.
- w: permission to write a file or create and remove files from a directory.
- x: permission to execute a program or change into a directory and do a long listing of the directory.
- -: no permission (in place of the r,w, or,x)

8.1.2 File Permissions in Linux

Each of the standard permissions types can be restrict access for the file's user, access for the file's group and access for everyone else.

Permissions on files for any particular user category are as follows:

read permissions means the contents of the file can be examined with a command such as cat or less.

write permissions means the file can be edited and saved.

execute permission means the shell will attempt to execute the file when its name is entered as a command.

8.1.3 Directory Permissions in Linux

Permissions on directories for any particular user category are as follows:

read permissions means the contents of the directory can be listed with ls.

write permissions means the file can be created in that directory.

execute permission means the user can cd to that directory and do that directory and do a long listing (read permissions without execute permission permits a listing,but not a long listing).

A file may be removed by anyone who has write permissionsbto the directory in which the file resides regardless of the ownership or permissions on the file itself.

8.1.4 Examining Permissions

- File permissions may be viewed using ls – l

  $ ls – l /bin/login
  -rwxr – xr - x 1 root root 19080 Apr 1 18:26 /bin/login

- File types and permissions represented by a 10-character string.

8.1.5 Interpreting Permissions

- rwxr – x - - - 1 andersen trusted 2948 Oct 11 14:07 myscript

- Read , Write & Execute for the owner, Andersen
- Read & Execute for members of the trusted group
- No access for all others

8.1.6 Changing Permissions – Numeric Method

- Permissions are calculated by adding:
  4(for read)
  2(for write)
  1(for execute)
- Example :
  Chmod 640 myfile

8.2 SPECIAL FILE PERMISSIONS

8.2.1 The Sticky Bit

- Normally users with write permissions to a directory can delete any file in that directory regardless of that file's permissions or ownership

- With the sticky bit set on a directory, only the owner of a file can delete the file.

- Example:

  ls –ld /tmp
  drwxrwxrwt 12 root root 4096 Nov 2 5:44 /tmp

To set the sticky bit on a directory,use chmod:

chmod o+t /home / share

Alternately:

chmod 1777 /home / share

The sticky bit will appear as a T if directory's execute permission for "others" is off.

8.2.2 Default File Permissions

- Read & write (not execute) for all is the default for files
- Read , write & execute is the default for directories
- umask can be used to withhold permissions on file creation
- Non-system users' umask is 002
    - Files will have permissions of 664
    - Directories will have permissions of 775
    - These permissions facilitate group collaboration

8.3 SUID AND SGID

When a user starts a process, it runs with the permissions of that user. For example, if you run vi, and try to edit a file which you do not have permission to read or write, the operation will fail. However, if the SUID or SGID bit is set on an executable, it runs with the permission of its owner(or group owner). For example, consider the file/etc/ shadow that stores users encrypted passwords:

- r ———— 1 root root 805 Sep 29 11:19 /etc/shadow

The file is owned by root, who has exclusive read access. User may still change their passwords with the passwd command, because the passwd command has its SUID bit set, and is owned by root:

- rwsr – xr – x 1 root root 13536 Jul 12 05:56 /usr/bin/passwd

SUID and SGID bits are set using the chmod command:

SUID :

Chmod u+s filename

SGID:

Chmod g+s filename

Since the SUID and SGID permission are displayed overlying the executive permissions for either user or group, respectively the case of the permission indicates whether the execute permission is turned on or off. For example, if capital is in the execute field, the SUID or SGID permission is on and the execute permission is off. If lowercase is in the execute field, both the SUID or SGID and the execute permissions are on.
For security reasons, SUID and SGID permissions are not honored when set on non-compiled programs, such a shell scripts.

8.3.1 SGID for Directories

- Used to create a collaborative directory.

- Normally, files created in a directory belong to the user's default group.
- When a file is created in a directory with the SGID bit set,it belongs to the same group as the directory.

When a file is created in a directory, it belongs to the primary group of the user that created the file.

mkdir /shared
ls –ld /shared
drwxr – xr – x 2 root root 4096 Feb 27 11:28 /shared
touch /shared/first
ls –l /shared
- rw – r- - r - - 1 root root 0 Feb 27 11:33 first

Recall that permissions can be set with chmod:

Chmod g+s directory

This sets the SGID bit without affecting current permissions. Alternately:

Chmod 2770 directory

The above syntax sets the SGID bit & gives read, write,& execute permissions to the owner of the directory & members of the group whose ownership is on that directory.

Chgrp mail /shared
#ls –ld /shared
drwxr – xr – x 2 root mail 4096 Feb 27 11:28 /shared
#Chmod g+s /shared
ls –ld /shared
drwxr – sr – x 2 root mail 4096 Feb 27 11:28 /shared

In this example,all future files will be created as members of the mail group.

touch /shared/second
ls –l /shared
- rw – r- - r - - 1 root root 0 Feb 27 11:33 first
- rw – r- - r - - 1 root root 0 Feb 27 11:33 second

8.1, 8.2 & 8.3 Check Your Progress

Q.1 Fill in the blanks

1. .....shows the permission to read the file.
2. .....shows the permission to write into the file.
3. .....shows the permission to execute a program or change into a directory and do a long listing of the directory.
4. .....is the command used display the file permissions.
5. File types and permissions represented by a .....characterstring.
6. With the .....set on a directory, only the owner of a file can delete the file.
7. The newly created file will have the default permission set as .....
8. The newly created directories will have the default permission set as .....

8.4 SUMMARY

In the above chapter, we learnt about the files & directories permissions in the Linux operating System. Also, we studied about the default files & directories permissions when they are newly get created. We also studied the importance of the Sticky bit and how to set the Sticky bit. We studied how to change the permissions of the files & directories as per their requirement.

8.5 CHECK YOUR PROGRESS- ANSWERS

8.1, 8.2 & 8.3

1. r
2. w
3. x
4. ls –l
5. 10
6. Sticky bit
7. 664
8. 775

8.6 QUESTIONS FOR SELF-STUDY

Q.1 Explain files & directories permissions in Linux in detail.

Q.2 Explain the special permissions such as Sticky bit, SUID & SGID in detail.

8.7 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

NOTES

CHAPTER 9

Processes & Signals in Linux

9.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- explain the terminology “process”
- analyze processes states & their priorities.
- describe job scheduling and xineted service

9.1 INTRODUCTION

A process is a set of instructions loaded into the memory. Every process has state property running, sleeping, sleeping uninterruptible, zombie. Signals are the simple messages that can be communicated to processes with some commands such as kill, killall, pkill etc. Scheduling priorities determines access to the cpu usage.Priority is affected by ‘nice value’. The values ranges from -20 to +19.
The default nice value is 0. Services which are needed less frequently or requiring additional resource management, are typically controlled by the xinetd service. Xinets uses /etc/ services in its configuration of port-to-service management.

9.2 PROCESS

A process is a set of instructions loaded into the memory.

9.2.1 Process States

Every process has state property

1) Running:
   State is the process actively using the CPU usage.

2) Sleeping:
The process is in memory but not doing anything.

3) Sleeping uninterruptible:
Process is sleeping & can't be woken up until an event occurs. It can't even be woken up by a signal. It is the result of an I/O operation, such as failed network connection.

4) Zombie:
Just before a process terminates it sends a signal to its parent & waits for the acknowledgement before terminating. Even if the parent process doesn't acknowledge immediately, all resources except the 'pid' are released.
Zombie processes are cleared from the system in the next reboot.

9.2.2 Listing Process

#ps:

Shows the processes information.
-a includes process from all terminals
-x includes process non attached to terminals
-u prints process owner information
-f prints process percentage
-o prints custom information; pid, %cpu, %mem, etc.

9.2.3 Finding Processes

#ps options | other commands
e.g. #ps axo comm.,tty | grep ttyS0

By predefining patterns: pgrep
#pgrep –U root
#pgrep –G student
By exact program name: pidof
#pidof bash

9.2.4 Verifying Specific Process Information

Since there may be hundreds of processes on a system, a common technique to locate a specific process is to send output from ps to grep:
e.g. #ps axo pid,comm. | grep 'cups'

2734 cupsd
3502 eggcups

Compare the above output with that from pgrep:

#pgrep cups

2734
3502

A more exact method of obtaining the pid's of processes is the pidof, which matches exactly on the program name specified & t5herefore requires that you know the specific program name:
#pidof cupsd
2734

9.3 SIGNALS

Signals are the simple messages that can be communicated to processes with some commands such as kill, killall, pkill etc.
Signals are specified by name or number when they are sent.
e.g.
Signal 15 or TERM
Signal 9 or KILL
Signal 17 or STOP

9.3.1 Sending Signals to Processes

By PID: kill [signal] pid …

By Name: killall [signal] comm. …

By Pattern: pkill [-signal] pattern

Some signals can be sent as follows:

#kill -15 pid (Default: Terminate cleanly)
#kill -9 pid (Terminate Immediately)
#kill TERM pid
#kill -17 pid (STOP)

e.g.:
The following are all identical & will send the default TERM signal to the process with PID 3428:
#kill 3428
#kill -15 3428
#kill –SIGTERM 3428
#kill –TERM 3428

9.3.2 Scheduling Process Priorities

Scheduling priorities determines access to the cpu usage. Priority is affected by 'nice value'. The values ranges from -20 to +19. The default nice value is 0.
Lower the nice value means higher CPU priority.
The nice value can be viewed by the command:

#ps o comm,nice

9.3.3 Altering Priority

Nice value may be altered when a process or command starts, by using the command:

#nice –n 5 command

After starting i.e. Altering priority of running process

#renice 5 pid
#renice –p -5 pid
#renice +15 –p pid

Note:
Non-privileged users may not set niceness value to less than 0; that is they may not request a higher than normal priority for their processes. Only root can do this. All users may reduce the priority of their own jobs using the renice command:
$renice 15 –p pid

9.4 SCHEDULING JOBS USING CRONTAB

Recurring jobs can be scheduled using crontab. The cron mechanism is controlled by a process called 'crond'.

This process runs every minute and determines if any entry in users cron tables need to be executed. Cronjobs can be scheduled as often as once a min or as infrequently as once a year.

The command is:

#crontab –e

```
  *       *          *          *          *              command
Min      hrs       DOM      Month   DOW
(0-59) (0-23)  (1-31)   (1-12)   (0-6)  0-Sunday
```

e.g.

```
#crontab –e
00       8         20        1         2         /bin/echo "Hello"
:wq
```

#service crond restart
#crontab –l (lists the cron jobs)
#crontab –r (remove the cron jobs)

9.5 THE XINETD SERVICE

Services which are needed less frequently or requiring additional resource management, are typically controlled by the xinetd service. Xinets uses /etc/services in its configuration of port-to-service management.

The default installed configuration of xinetd is provided by the top level configuration file /etc/xinetd.conf & service specific files under the etc/xinetd.d directory tree.

The top level configuration file /etc/xinetd.conf sets the global configuration options shared by all managed services. It also provides the path to service specific configuration.

9.1 to 9.5 Check your Progress

Q.1 Fill in the blanks

1. .....state is the process actively using the CPU usage.
2. .....process is the process in memory but not doing anything
3. .....is the process sleeping & can't be woken up until an event occurs.
4. Just before a process terminates it sends a signal to its parent & waits for the acknowledgement before terminating is a .....process.
5. .....command shows the processes information.
6. Set of instructions loaded into memory is called as a .....
7. .....are the simple messages that can be communicated to processes with some commands.
8. Scheduling .....determines access to the cpu usage.
9. Recurring jobs can be scheduled using .....
10. .....defaults to the keyboard.

9.6 SUMMARY

In this chapter we studied the definition of a process, what exactly a process is. Also the chapter includes the study of Process priorities, how to set or change the priorities of the process, study of Signals, sending signals to processes. We studied the xinetd services and the standard input and output operators in Linux.

9.7 CHECK YOUR PROGRESS- ANSWERS

1. Running
2. Sleeping
3. Sleeping uninterruptible
4. Zombie
5. ps
6. process
7. Signals
8. priority
9. crontab
10. Standard Input

9.8 QUESTIONS FOR SELF-STUDY

Q.1 Define process and explain different processes states.

Q.2 Explain Listing processes, Finding processes & Verifying specific process information.

Q.3 Define Signals and explain sending signals to processes.

Q.4 What is scheduling of processes priorities? And altering priorities?

Q.5 Explain scheduling jobs using crontab with example.

Q.6 Explain Input/Output streams.

9.9 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

CHAPTER 10

Study of File System & Related Commands

10.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- describe installation of softwares in a Linux Operating Systems.
- explain, how to remove and upgrade software in linux
- brief explanation of "tar " command.

10.1 INTRODUCTION

The rpm command is used to install the packages (softwares) into the linux system. Archiving places many files into a single targeted file. Tar command is used for archiving the files. It supports compression using gzip & gunzip and also bzip & bunzip. Adding new filesystem to file system tree Identify Device, Partition Device, Label File System, Create new entry in /etc/fstab, Mount the new File System.

10.2 STUDY OF RPM COMMAND

Red Hat Package Management (RPM).
The rpm command is used to install the packages (softwares) into the linux system.
Primary RPM options

1) Install: rpm –i | —install rpm file
2) Upgrade: rpm –F | —freshen rpm file
3) Removal rpm –e | —erase package

Output Options: -v, -h

Syntax for Install:
#rpm –ivh packagename

Syntax for Upgrading:

#rpm – Uvh packagename

Syntax for Removing:

#rpm –e packagename

Syntax for Freshening:

#rpm –Fvh packagename

10.2.1 RPM Queries

#rpm –qa: lists all installed packages

#rpm –q packagename: lists the named package & version

#rpm –qf packagename: lists the package that owns the package

#rpm –qp rpmfile: lists the possibly uninstalled package

10.2.2 RPM Verification

Installed package file verification

#rpm –V packname

#rpm –Vp rpmfile

#rpm –Va

Verifying installed package compares the file sizes, permissions, type, owner, group, MD5 checksum & modify time against the RPM database.
An installed package can also be verified against a package file as well.

Example:

#rpm – V zip –verifies the installed zip rpm against the rpm database

#rpm –Va –verifies all installed rpms against the rpm database.

10.3 STUDY OF TAR COMMAND

Tar (tape archiving) command.

Archiving places many files into a single targeted file. Tar command is used for archiving the files. It supports compression using gzip & gunzip and also bzip & bunzip.

#tar –cvf filename.tar /data/

-c : create archive
-v : verbose mode
-f : archive name
-t : lists the contents
-x : extract the contents
-z : used for gzip compression
-j : used for bzip compression

e.g.

#tar –cvf backup.tar /storage/

#tar –xvf backup.tar

#tar –xvf backup.tar

#tar –czvf backup.tar.gz /storage/

#tar xzvf backup.tar.gz

#tar –cjvf backup.tar.bz2 /storage/

#tar –xjvf backup.tar.bz2

10.4 DISK RELATED COMMANDS

#df

Displays the amount of disk space available on the file system containing each file name argument.

#df -h :- print sizes in human readable format (e.g., 1K 234M 2G)
#df -a :- prints
#df -l :- —local limit listing to local file systems.

#du

Displays the memory of files residing in the current working directory.

#du -h

Displays the memory of files in the current working directory in human readable format.

#fdisk –l

Displays & lists the details of the hard disks partitions.

10.5 FILE SYSTEM MANAGEMENT

(Disk Partitioning & Formatting)

Adding new filesystem to file system tree:

1) Identify Device
2) Partition Device
3) Label File System
4) Create new entry in /etc/fstab
5) Mount the new File System

10.5.1 Identifying Device

#fdisk /dev/sda

-l : lists partition id
-p : print partition table
-n : create new partition
-w : save the partition
-t : toggle the partition
-d : delete partition

Partition id's:

83 : ext2 or ext3 linux partition
82 : swap partition
8e : Linux LVM
Fd : Linux RAID

Linux maximum number of partitions supported by the Kernel:

63 for IDE drives

15 for SCSI drives

10.5.2 Partitioning Device

#fdisk /dev/sda

Press 'n' to create new partition

Specify desired or required size (e.g. +100M)

Press 'w' to save the partition

Use the command **#partprobe** to save changes without reboot

10.5.3 Making the File System

#mkfs

-b : specify block size

-i : specify bits/inode ratio

-N :specify no of inode reserved for the file system

-m : specify percentage of reserved blocks

-L : set the label

-j : create ext3 journal inode (default for ext3)

-t : specify file system type

e.g. #mkfs –t ext3 <device name>

#mke2fs –j <device name>

10.5.4 Labeling the Device

#e2label <device name> <label name>

#mount LABEL=label name <device name>

e.g. #e2label /dev/sda7 dbdisk1

#mount LABEL=dbdisk1 /mnt/data

10.5.5 Creating Entry in /etc/fstab (Study of /etc/fstab)

#vi /etc/fstab

Device	mount point	filesystem type	Options	Dump frequency	FSCK order
/dev/sda1	/	ext3	defaults	1	1
/dev/sda2	/boot	ext3	defaults	1	2
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:

10.5.6 Mounting the File System

#mount [options] <device name> <mount point>

e.g. #mount –t ext3 /dev/sda5 /data

#mount –o,remount rw /data

10.2 - 10.5 Check your Progress

Q.1 Fill in the blanks

1. The .....command is used to install the packages (softwares) into the linux system.

2. .....command lists all installed packages.

3 .....places many files into a single targeted file.

4. .....command is used for archiving the files.

5. .....command displays the amount of disk space available on thefile system containing each file name argument.

6. .....command print sizes in human readable forms.

7 .....commands displays the memory of files residing in the current working directory.

8 The partition ID .....shows ext2 or ext3 partition.

10.6 SUMMARY

In this chapter we studied to install the new software packages in the Linux Operating System. Also we studied about the upgrading and removal of the installed software packages.

The chapter also includes how to list the already installed package as per our requirement.

We studied the tape achieving command to place a number of files in to a single targeted file. The chapter includes some disk related commands which can be used to display the information about the hard disk, free space availability etc. The most important part of the chapter is disk partitioning and formatting in which we studied from the identification of the device to mounting the device.

10.7 CHECK YOUR PROGRESS- ANSWERS

10.2, 10.3, 10.4, & 10.5

1. rpm
2. rpm –qa
3. Archiving
4. Tar
5. df
6. df-h
7. du –h
8. 83

10.8 QUESTIONS FOR SELF-STUDY

Q.1 Explain the Red Hat Package Management in detail.

Q.2 Explain tar command with example.

Q.3 List and explain disk related commands with example.

Q.4 Explain disk partitioning and formatting in detail.

10.9 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

CHAPTER 11

Accessing File Systems & Related Devices

11.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- describe process of accessing a file system, as well as accessing a pen drive & other devices in the Linux system.
- analyse some scripting basics & troubleshooting part.

11.1 INTRODUCTION

The mount or df command will enable you to see which disks are mounted. Linux is a great system when it comes to doing remote administration.You can connect to a remote system using many different techniques. A runlevel is defined when the computer starts up. When it boots, Linux starts the kernel which loads a first process called init. This process monitors the system run state and then consults the init table (located at /etc/inittab) to start daemons and the other processes. Shell scripts are the files that contain a series of commands to be executed. Shell scripts are used for automating commonly used commands, performing system administration tasks and also for creating simple applications while creating the shell scripts. Linux, being based on one of the oldest network operating systems (UNIX), is loaded with standard troubleshooting tools.

11.2 MOUNTING DEVICE

In order to use a disk device, it needs to be active or “mounted”. The mount or df

command will enable you to see which disks are mounted.

Figure - The Mount command gives you the mount points and device status

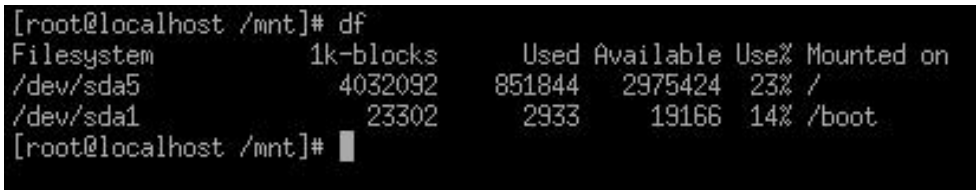


Figure -The df command gives you physical disk information

This indicates which disks are currently active. The mount command activates on startup. You can access most devices starting with the root. Removable devices will be placed under the /mnt folder. However, if you want to mount a different cd drive without rebooting the system, you will need to mount it.

To unmount a device, use the umount command.

11.3 MANAGING REMOTE SYSTEMS

Linux is a great system when it comes to doing remote administration. You can connect to a remote system using many different techniques. Here are the most common ones:

- Telnet : This command will enable you to connect to another computer and establish a shell session. You will then be able to enter commands just as if you were directly in front of the remote computer.
- Ssh: ssh is more or less the same thing as telnet except it is a more secure way of doing it. Telnet uses clear text authentication and no encryption. Ssh is using a more secure authentication mechanism that can even use security public certificates and it then encrypts the whole session.
- Ftp: The ftp command enables you to connect to an ftp server enabled machine and manage files. This is a very common technique on the internet and most people don't really know about its potential. Ftp stands for File Transfer Protocol and can move files from one computer to another. It contains many commands that you should have basic knowledge of.
- You can also redirect an Xwindows session or use a remote desktop software like AT&T's VNC.

11.4 RUNLEVELS AND INIT & STUDY OF /ETC/INITTAE

Think of runlevels as different modes in which linux can operate (just as windows can start in safe mode or regular mode). A runlevel is defined when the computer starts up. When it boots, Linux starts the kernel which loads a first process called init. This process monitors the system run state and then consults the init table (located at /etc/inittab) to start daemons and the other processes. The init table file contains information on the runlevel. There are 7 levels:

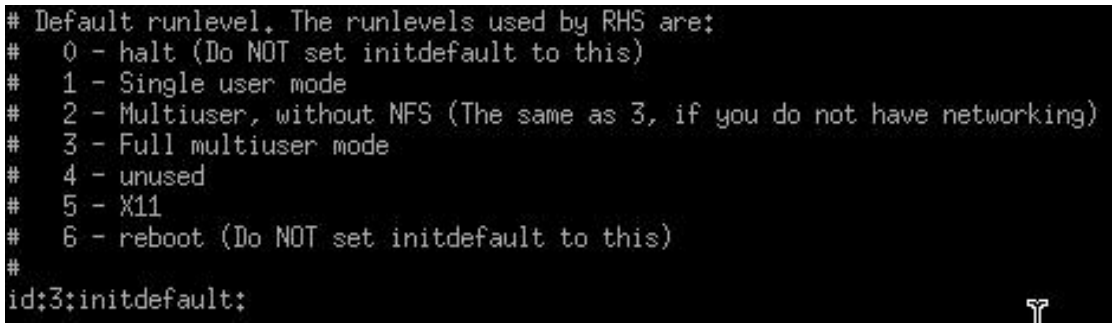


Figure - /etc/inittab

11.5 SHELL PROGRAMMING

11.5.1 Scripting Basics

Shell scripts are the files that contain a series of commands to be executed. Shell scripts are used for automating commonly used commands, performing system administration tasks and also for creating simple applications while creating the shell scripts. The first line contains #! (shebang sequence).

#!/bin/bash : used for bash scripts.

#!/bin/sh : used for bourne shell scripts.

#!/bin/csh : used for c shell scripts.

#!/usr/bin/perl : used for perl shell scripts.

#!/usr/bin/python : used for python shell scripts

11.5.2 Executing Scripts

Shell scripts files have the extension ".sh". Scripts files are executed with the command:

#sh filename.sh

#./ filename.sh

For executing the script files, user should have execute permission.

11.6 CONDITIONAL STATEMENTS

Fundamentals to all programming languages is the ability to test conditions and perform different actions based on those decisions. A shell script can test the exit code of any command that can be invoked from the command line, including the scripts that you have written yourself.

11.6.1 The test or [Command

In practice, most scripts make extensive use of the [or test command, the shell's Boolean check. On some systems, the [or test commands are synonymous, except that when the [command is used, a trailing] is also used for readability. Having a [command might seem a little odd, but within the code it does make the syntax of commands look simple, neat and more like other programming languages.

We will introduce the test command using one of the simplest conditions:

checking to see whether a file exists. The command for this is test –f <filename>, so within a script you can write as:

```
If test –f fred.c
then
….
fi
```

You can also write like this:

```
If [ -f fred.c]
then
….
fi
```

The condition types that you can use with the test command are of three types: string comparison, arithmetic comparison, file conditionals.

11.6.2 Control Structures

The shell has a set of control structures, which are very similar to other programming languages

1) If :

   The if statement is very simple: It tests the results of a command and then conditionally executes a group of statements:

```
If
        condition
then
        statements
else
        statements
fi
```

   Example:

   A common use for if is to ask a question & then make a decision based on the answer:

```
#!/bin/sh

echo “Is it morning? Please answer yes or no”
read timeofday

if  [ $timeofday =  “yes” ]
        echo “Good morning”
else
        echo “Good Afternoon”
fi
exit 0
```

   This would give output as:

```
Is it morning? Please answer yes or no
Yes
Good morning
$
```

2) Elif

   Unfortunately, there are several problems with this very simple script so that it reports an error message if the user types in anything other than yes or no.

```
#!/bin/sh
echo “Is it morning? Please answer yes or no”
```

```
read timeofday

if  [ $timeofday =  “yes” ]; then
        echo “Good morning”
elif [ $timeofday = “no” ]; then
        echo “Good  afternoon “
else
echo “Sorry, timeofday not recognized. Enter yes or no”
exit  1
fi
exit 0
```

3) For

Use the for construct to loop through a range of values, which can be any set of strings. They could be simply listed in the program or more commonly, the result of a shell expansion of filenames.

Syntax:

```
for variable in values

do

statements

done
```

4) While

Because all shell values are considered strings by default, the for loop is good for looping through a series of strings, but is not so useful when you don't know in advance how many times you want the loop to be executed.

When you need to repeat a sequence of commands, but don't know how many times they should execute, you will normally use a while loop.

Syntax:

```
while condition do
        statements
done
```

Example:

```
#!/bin/sh
echo “enter password”
read trythis

while [  “$trythis” != “secret” ]; do
        echo “Sorry, try again”
        read trythis
done
exit 0
```

Output will be as:

```
Enter password
Password
Sorry, try again
Secret
$
```

5) Until

The until statement has the following syntax:

```
until condition
```

do

statements

done

11.7 SOME EXTRA THINGS TO LEARN

11.7.1 Troubleshooting

n order to make troubleshooting as easy as possible, you should always use an organized methodology. Using simple best practices will do just that.

Best practices:

The best tip when it comes to troubleshooting best practices is to document all of your operations. This will prove helpful in critical situation as you will be able to find out about service dependencies, permission issues, etc. Start with quick fixes: if a problem sounds familiar, try using a couple of quick tricks. This often addresses the issue. Do not act randomly: use a proper order to find a problem. E.g. beginning by looking at hardware, then software, looking at recent changes, looking at logs, asking the user about the nature of the problem (sometimes the problem can be the user), etc. If all symptoms seem to point at a certain service or process, you can kill and restart it.

You are expected to be able to inspect and determine cause of errors from system log files using such commands as locate, find, grep, ? , <, >, >>, cat, tail.

A lot of error messages in linux come from different versions of software and the dependencies associated with them. If you change or update a php package for example, a php based program might stop working. You should use the *rpm* command to view proper dependencies, document any changes and verify dependencies before making any changes.

11.7.2 Troubleshooting the File System

To verify and repair a file system, you can use the *mount* command to enumerate the different partitions on the system and the *fsck* command to repair them.

You can use the df command to see the space used on each disk. Problems can occur when a disk is full.

11.7.3 Troubleshooting the Boot Process

Even with the strongest file systems, failure will happen. You may encounter situations where the system boots in single user mode. This is an operating mode that doesn't start all daemons and is useful for troubleshooting. In this mode you will be given the opportunity to use different troubleshooting tools including file system integrity using *fsck*. In the case where a system won't boot, it is a good idea to boot from a floppy and inspect the filesystem and boot sector. A boot disk should always contain fsck as it will enable you to repair and rescue a damaged file system.

11.7.4 Troubleshooting Backup and Restore Errors

Backups can fail for many reasons. The most common causes are media and drive related issues. Most media requires proper maintenance and cleaning. Tape corruption, low device space or write failures are common problems. Proprietary software will have specific error messages and you should refer to your software provider to verify them. Backups should always be handled with care. You should do a regular restore test as it is not uncommon to see successful backups that cannot be successfully restored.

11.7.5 Troubleshooting Networking

Linux, being based on one of the oldest network operating systems (UNIX), is loaded with standard troubleshooting tools. Some of these tools are:

- Ping: the ping utility enables you to verify basic connectivity between two machines.

```
[root@localhost /etc]# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=0 ttl=255 time=83 usec
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=1 ttl=255 time=82 usec
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=2 ttl=255 time=114 use
c

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.082/0.093/0.114/0.014 ms
[root@localhost /etc]#
```

- Route: the route utility helps you take a look at the various routes defined within the Kernels routing table. You will be able to add, delete, and modify routing information here. This is very helpful when using your Linux box as a router or firewall.

```
[root@localhost /etc]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
[root@localhost /etc]#
```

- Traceroute: this utility enables you to see every router between your Linux machine and a given host. This way it is possible to see any failing point between you and this host.

```
traceroute [ -dFInrvx ] [ -f first_ttl ] [ -g gateway ] [
-i iface ]
       [ -m max_ttl ] [ -p port ] [ -q nqueries ]
       [ -s src_addr ] [ -t tos ] [ -w waittime ]
       host [ packetlen ]
```

- Netstat: this utility helps you see your network interfaces statistics.

```
[root@localhost /var]# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  1      [ ]         STREAM     CONNECTED     7332   @000000bb
unix  1      [ ]         STREAM     CONNECTED     7319   @000000ba
unix  1      [ ]         STREAM     CONNECTED     6888   @00000090
unix  1      [ ]         STREAM     CONNECTED     7213   @000000ac
```

- Lsof: this utility lets you see any open files.

```
[root@localhost /var]# lsof |more
COMMAND     PID USER   FD   TYPE     DEVICE    SIZE      NODE NAME
init          1 root  cwd    DIR        8,5    4096         2 /
init          1 root  rtd    DIR        8,5    4096         2 /
init          1 root  txt    REG        8,5   27452     96317 /sbin/init
init          1 root  mem    REG        8,5  434945    384774 /lib/ld-2.1.92
init          1 root  mem    REG        8,5 4776568    384781 /lib/libc-2.1.
o
```

- Ifconfig: this utility lets you see your network interfaces and modify certain settings.

```
[root@localhost /var]# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:54 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

11.1-11.7 Check Your Progress

Q.1 Fill in the blanks

1. .....are the files that contain a series of commands to be executed.
2. .....shell is used for bash scripts.
3 .....shell is used for bourne shell scripts.
4. .....shell is used for c shell scripts.
5. .....shell is used for perl shell scripts.
6. .....shell is used for python shell scripts.
7. Shell scripts files have the extension .....
8. The .....utility enables you to verify basic connectivity between two machines.
9. The .....utility helps you take a look at the various routes defined within the Kernels routing table.
10. The .....utility enables you to see every router between your Linux machine and a given host.
11. The .....utility helps you see your network interfaces statistics.
12. The .....utility lets you see any open files.
13. The .....utility lets you see your network interfaces and modify certain settings.

11.8 SUMMARY

In the last chapter we studied the mounting of the devices such as CDs, DVDs etc. We studied how to manage a Linux system by using a remote login using some protocols such as Telnet. Also we studied about the run levels in which a Linux system can be used. Some shell scripting basics are also included in the chapter above. The study of conditional statements and control structures helps in shell programming.

The very important part of the chapter is the troubleshooting of the Linux system. It helps to preserve the data onto a Linux system and the Linux system can be recovered if it gets crashed without losing your important data after studying the troubleshooting of Linux.

11.9 CHECK YOUR PROGRESS- ANSWERS

11.1 - 11.6

1. Shell scripts
2. /bin/bash
3. /bin/sh
4. /bin/csh
5. /usr/bin/perl
6. /usr/bin/python
7. .sh
8. ping
9. route
10. Traceroute
11. netstat
12. Lsof
13. ifconfig

11.10 QUESTIONS FOR SELF-STUDY

Q.1 Explain mounting device and Runlevels of Linux system.

Q.2 Explain how to manage a Linux system remotely.

Q.3 Explain conditional statements with example.

Q.4 Explain troubleshooting of a Linux system in your language.

Q.5 Explain /etc/inittab file.

11.11 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

Chapter 12

Network Configuration in Linux

12.0 OBJECTIVES

After studying this chapter you will be able to understand following things:

- explain the use of mii-tool
- describe various configuration utilities

12.1 INTRODUCTION TO NETWORK CONFIGURATION

While you can compile specific Ethernet card driver code into the kernel statically .Red Hat compiles network card drivers as kernel modules for easy adaptability to any hardware configuration. Network interface modules are loaded at boot time if networking has been enabled. The appropriate module is loaded based on a alias line in `/etc/modprobe.conf`.
Below is an example of the contents of `modprobe.conf`:

```
alias eth0 3c59
alias parport _low level  partport _pc
alias sound –slot-0 es 1371
alias usb-controller usb-uhci
```

The logical Ethernet interface name (i.e. etho,eth1,etc)is used in through configuration file and scripts to associate kernel driver module with a specific interface. If you have an ISA network card ,you can specify options for each card by its IRQ and/or I/O address:

```
alias eth0 3c509
alias eth1 tulip
options 3c509 io=0x210
```

Documentation for networking module options is in `/usr/src/linux-2.6/Documentation/networking/net-modules.txt`

Interface names
The Linux kernel names interfaces with a specific prefix depending on the type of interface .For example, all Ethernet interfaces start with eth. Notice that regardless of the specific hardware vendor, the interfaces start with a common prefix .Following the prefix , each interface is numbered ,starting at zero. For example,eth0, eth1,eth2 would refer to the first .second and third Ethernet interface.

Layer 2 hardware address
The hardware address of network interfaces can be determined by running the ifconfig command. Another method is to examine the output from device driver (kernel module) as it loads. Check the output of the dmesg command and or `/var/log/dmesg`.

12.2 MII TOOL

mii-tool

mii-tool allows a system administrator to view , monitor ,log and change the negotiated speed of Ethernet network cards. This capability depends upon the card having a chipset compatible with the mii-tool and not all cards are compatible

```
$mii-toool-v
eth0: negotiated 100baseTx-FD,link ok
product info: Level one LXT970/971 rev 0
basic mode :autonegotiation enabled
basic status :authonegtiation  complete ,link ok
capabilities : 100 baseTx-FD 100baseT-FD 10baseT-HD
advertising : 100baseTx-fd 100base Tx-HD 10baseT-FD 10baseT-HD
link partner : 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
```

To force 100Mbps full duplex operation of eth1

```
$ifdown eth1
$mii-tool –v – force 1000baseTx-FD eth1
$ifup eth1
```

To restart auto negotiation :

```
$ ifdown eth1
$ mii–tool -vr eth1
$ if up eth1
```

When ifconfig is run with no arguments, it displays information on all currently active interfaces .The information displayed includes the type of interface , IP address information ,hardware addresses ,various statistics ,and hardware resources .

```
[root@localhost/tmp]#ifconfig
Eth0 Link encap :Ethernet HWaddr 00 : A0 CC : 37 :38:88
inet addr:10.100.0.1 Bcast:10.100.0.255 Mask:255.255.255.255.0
UP BROADCAST NOTRAILERS RUNNING MTU :1500 Metric :1
RX packets:151110 errors:0 dropped:0overruns:0 frame:0
TX packets:88699 errors:1 dropped:0 overruns:0 carrier:2
Collisions : 33100
RX bytes:7591398(7.2Mb) TX bytes:173492(169.4 Kb )

lo      link encap:Local Loopback
        Inet addr :127.0.0.1 Mask ;2555.0.0.0
        UP LOOPBACK RUNNING MTU :3924 Metric : 1
RX packets:166 errors :0 dropped :0 overruns:0 frame :0
TX packets:166 errors :0 dropped : 0 overruns:0 carrier :0
        Collisions:0
RX bytes :51066 ( 499 .0 Kb )TX bytes : 5066 (499.0 Kb)
```

In order to view inactive interfaces ,use the –a option

Bringing up and down a network interface

The process of bringing up a network interface involves several possibilities and options. It isn't as simple as just configuring an IP address. For example , what IP address should the interface be configured with ?Should DHCP or BOOTP be used ? If the interface is a PPP interface associated with a modern ,the modern needs be instructed to dial ,and the pppd daemon started.

Another important task that is performed with bringing up an interface is the addition ,deletion , or changing of routes in the routing table.
The ifup and ifdown scripts take care of all the extra tasks that need to be performed when activating and deactivating a network interface. The ifdown command is used to disable the Ethernet card. For e.g.

#ifdown eth0

This command will disable 1st Ethernet card.
The ifup command is used to enable to Ethernet card. For e.g.

#ifup eth0

This command will enable 1st Ethernet card.

Interface Configuration Files

Red Hat Enterprise Linux stores network interface configuration information in files in the directory /etc/sysconfig/network-scripts**.** The file names are prefixed with ifcfg- and then the name of the interface. For example, the file for the first Ethernet interface would be ifcfg–eth0.
For example, to put the eth0 interface under dhcp control and have it activated at boot time, your interface config file would need these contents:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

For an interface with static IP address configuration ,the following is needed

```
DEVICE =eth0
IPADDR =xxx.xxx.xxx.xxx
NETMASK=static
BOOTPROTO=static
ONBOOT=yes
```

12.1 &12.2 Check Your progress

1. Fill in the blanks

1] The hardware address of network interfaces can be determined by running the ..... command.

2] All Ethernet interfaces start with .....

3] ..... allows a system administrator to view , monitor ,log and change the negotiated speed of Ethernet network cards

4] ..... command is used to disable Ethernet card.

5] ..... is used to enable Ethernet card.

6] ..... command is used to set Ethernet interface.

2. State True or False

1] All Ethernet interfaces start with eth.

2] ipconfig command shows the configuration of Ethernet card.

3] dhclient command is used to get IP address temporarily from DHCP server.

4] Ping stands for Packet Internet grouper.

5] The traceroute command will attempt to show the path of routers network packets take between the local system.

Configuring Utilities

Netconfig:

netconfig is a curses-based tool that to configure network interfaces, either as DHCP client or with a static IP address, nameserver ,and gateway .By default it modifies the settings for the first Ethernet interface (eth0),but the " - -*device interface* "argument can be used to set up other network interfaces :

```
netconfig –device eth2
```

The Red Hat Network Administration Tool is a X-based utility that can be used to set up Ethernet, ISDN, PPP, xDSL, token ring ,CIPE, or wireless network interfaces. It can be started from the command line with system-config-network command, or by selecting the "Network Configuration" tool through the panel menu in GNOME or KDE.
Note that using system-config-network creates an alternate file hierarchy under `/etc/sysconfig/networking`. Modifying an interface with this tools will create a hard link between the files in `/etc/susconfig/network-scripts` and in `/etc/sysconfig/networking/profiles/<profile>.` Both files will Have the same name. (e.g. ifcfg –eth0.)

Binding Multiple IP Addresses

Virtual Interface

Linux is commonly deployed in the role of web or ftp server. Often a situation calls for many web sites or ftp sites running on the same server .This called virtual hosting. If you plan on supporting SSL or ftp for different servers , then you are required to use a separate IP address for each server .Note that non-SSL web hosting requires only one IP address. If you have many IP addresses to bind to a single network card , it is more convenient to use an interface configuration range file than to create many separate interface configuration files .For example the file ifcfg–eth0–range0 with the following contents would bind the IP addresses 10.100.13.75
Through 10.100.13.210 to the eth0 network card :

```
IPADDR_START =10.100.13.75
IPADDR_END = 10.100.13.210
CLONENUM_START=0
```

You can have multiple range files ,just be sure that you don't create conflicts .Currently each range fie is limited to addresses within the same class C-sized block .A maximum of 256 IP addresses may be bound to a single NIC.

DHCP/BOOTP

dhclient stores interface configuration in

`/var/lib/dhcp/dhclient-ethX.leases.`
dhclient command is used to get IP address temporarily from DHCP server.

Below is a example of a typical file :

```
Lease {
Interface "eth1"; Network Configuration in Linux / 90
Fixed –address 192.168.0.6;
Filename "/kickstart/workstation.cfg";
Option subnet-mask 255.255.255.0;
Option routers 192.168.0.254;
Option dhcp –lease –time 21600;
Option dhcp –message –type 5 ;
```

```
Option domain –name –servers 192.168.0.254 ;
Option dhcp –server –identifier 192 . 168.0.254 ;
Option domain –name “example.com”;
Renew 1 2003/10/6 01:16:30;
Rebind 1 2003/10/6 03:49:38;
Expire 1  2003/10/6 04:34:38 ;
```

Global network parameters

```
/etc/sysconfig/network :
NETWORKING=yes|no
HOSTNAME=<fqdn by default , but whatever hostname you want >
GATEWAY=<gateway IP>
NISDOMAIN=<nis domain name >
```

The Default Route

The default route specifies where IP packets should be sent that are destined for networks that your machine doesn't know how to reach.

With RHEL ,if you don't set a system global default route in /etc/sysconfig/network, you can define default routes associated to specific interfaces .When such an interface is activated that default route is set in the machine's routing table.

Static routes are set to active on a per-interface basis. Either of two files may be used to set a static route.
Lines in the `/etc/sysconfig/network-scripts/route-(ifname)` file for the interface use the same syntax as the 'ip route add' command:

IP/CIDR via GATEWAY

For example :

192.168.2.0/24 via 192.168.0.128

Adds a static route to the 192.168.2 network through the 192.168.0.128 router.
The system –config-network command uses a different File,/etc/sysconfig/networking/devices/(ifname).route , to set static routes . This file uses a different syntax :

```
ADDRESS0 =192.168.2.0
NETMASK0 =255.255.255.0
GATEWAY =192.168.0.128
```

The second static route uses ADDRESS1/NETMASK1/GATEWAY1 and so on.

Local Name Resolution

If you have a small network of computers , you can eliminate DNS lookups for communications between the hosts by creating a standard/etc/hosts file and using it on all your machines .Once you grow beyond a handful of computers ,DNS provides better scalability and manageability . At a minimum , your /etc/hosts should contain localhost and the IP address for your Ethernet interface.
A typical `/etc/hosts` file :

```
[root@station1 /root] # cat /etc/hosts
127.0.0.1          localhost.localdomain localhost
10.100.0.1         station1.example.com station1
```

The file /etc/host.conf sets the order of name resolution for the localhost (whether to check the /etc/hosts file or query the name server first):

```
[root@dawg /root] $ cat /etc/host.conf
order hosts,bind
```

Client –side DNS configuration

Domain name service is responsible for associating domain names (i.e. joe.somewhere.com)with IP addresses (i.e.,192.168.1.24) A typical /etc/resolv.conf file looks like the following:

```
Search somewhere .com dyn .somewhere.org
              Nameserver 192.168.1.2
              Nameserver 192.168.1.3
```

Nameservers are checked in order .The first name server listed will be queried and if it is unavailable the second name server will queried and so on down the list.
The search specification allows up to six domains to be searched when the system is attempting to resolve a hostname that is not fully qualified.
See the resolv.conf(5) man page for information about other options you can include in this file.

Ping
Ping stands for Packet Internet grouper. It is used to test TCP/IP connectivity. Ping attempts to test network connectivity by sending ICMP packets to a specific system on the network .If the remote system is accessible via the network ,it will reply.
The default behavior of ping is to send a 64 byte ICMP packet to the specified host very second until you cancel the operation with Ctrl-C .When the ping operation is canceled will report summary statistics such as average packet loss, number of packets sent/ received,etc.

```
$ ping www.redhat.com
PING   www.portal.redhat.com   (206.132.41.202)from
10.100.0.1:56(84)byte  of data.
64 bytes from 206.132.41.202: icmp_seq=0 ttl =242 time=100.760
msec
64 bytes from 206.132.41.202: icmp_seq=0 ttl =242 time=90.170
msec
64 bytes from 206.132.41.202: icmp_seq=0 ttl =242 time=100.708
msec
64 bytes from 206.132.41.202: icmp_seq=0 ttl =242 time=100.312
msec
—— www.portal.redhat.com ping statistics——
4 packets transmitted , 4 packets received ,0% packey loss
Round-trip min /avg/max/mdev = 90.170/93.987/100.760/4.233 ms
```

Traceroute

As network traffic travels across the Internet it usually travels through multiple routers.When connectivity between a local system and a remote system is sluggish and inconsistent ,it is useful to investigate which is responsible for the network problem.
The traceroute command will attempt to show the path of routers network packets take between the local system and a remote system**. This command by default uses UDP not ICMP**. See the man page for details.

```
#traceroute 192.168.1.100
```

12.4 SUMMARY

Red Hat compiles network card drivers as kernel modules for easy adaptability to any hardware configuration. The hardware address of network interfaces can be determined by running the ifconfig command.

mii-tool allows a system administrator to view , monitor ,log and change the negotiated speed of Ethernet network cards.

netconfig is a curses-based tool that to configure network interfaces,either as DHCP client or with a static IP address, nameserver ,and gateway .By default it modifies the settings for the first Ethernet interface (eth0),but the " - -*device interface* "argument can be used to set up other network interfaces

Ping stands for Packet Internet grouper. It is used to test TCP/IP connectivity.

12.5 CHECK YOUR PROGRESS- ANSWERS

12.1 & 12.2

1) 1] (ifconfig)
   2] (eth)
   3] (mii tool)
   4] (ifdown)
   5] (ifup)
   6] (netconfig)

2) 1] True
   2] False
   3] True
   4] True
   5] False.

12.6 QUESTIONS FOR SELF STUDY

1. What is mii tool?
2. Explain ping command in detail.

12.7 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus

Chapter 13

Troubleshooting

13.0 OBJECTIVES

Friends, After studying this chapter you will be able to

- describe X Windows Troubleshooting
- analyze Network Troubleshooting
- explain order of Boot Process
- describe file system corruption & Rescue Environment

13.1 INTRODUCTION

The process of troubleshooting any system ,including those running Red Hat Enterprise Linux (RHEL) is both science and art .The science comes from the concepts of hypothesis testing, experimentation, comparison, and reproducing results. The art of troubleshooting comes from the realization that operating systems, services and applications do not always work s we hope or anticipate, or even as their creators hope or anticipate.

Regardless of whether the problem is something wrong in a single user's environment or a system-wide crisis that has rendered a system unusable, sensible troubleshooting begins with easy fixes. This may mean running configuration tools, or it may mean looking at service specific log. Logs often provide explicit information on problems, sometimes even identifying the exact line of configuration file that is causing problems, so it is often far easier to look in a log for an answer then to parse a configuration file for what might be trivial syntax error. Many services provide switches that enable higher levels of debugging output, and some may be run as foreground applications for debugging purposes. Some services such as Samba, provide syntax checkers that may also useful.

Sometimes the problem you are having is a bug, not misconfiguration. If you have followed instructions and documentations, only to find this do not work as you assume they should, then perhaps you should run bug tracking tools.

13.2 X WINDOWS TROUBLESHOOTING

X can be problematic. The first thing to try when confronting X problems is system-config-display. If the X problem is occurring in run level 5, it could make logging in impossible in X or in virtual consoles. Reboot or change the system run level to 3.

Sometimes viewing the output of the X command itself is revealing. The –probeonly switch will perform all tasks necessary to start X server without actually starting it. Thus, the startup messages are displayed. If the text scrolls off the screen, the material off screen can be viewed by holding down shift key and pressing Page Up key (Shift /Page Down to scroll down again) The output may be viewed by holing down Shift and using

Page Up and Page Down. /usr/share/hwdata/Cards may provide useful information about optional configuration features for your hardware.

A user's inability to create files in user's home directory or in /tmp directory either because of a full file system or because of a hard quota limit typically inhibits the ability of the user to run the X. Although the exact symptoms differ between runlevels, messages will appear stating (in runlevel 3) or suggesting (in runlevel 5) that a file system is full.

The xfs font server is the only font path entry in the /etc/X11/xorg.conf file, so if it is not running ,X will not run. Make sure that xfs is configured to run in the appropriate runlevels. Occasionally it may be necessary to delete stale lock,pid, or socket files. Once in a while ,the font indexes in a font directory may be corrupt, and it will be necessary to run mkfontdir to recreate them. When this happens, xfs may seem to start correctly, but then dies. Try commenting out font paths in /etc/X11/fs/config ,then run xfs from a terminal to determine which directory has problems.

Remember that X is a network service, even when you have not enabled access to your display. Unsuccessful hostname resolution can produce various behaviors, including the inability to launch applications from the panel. Changing hostnames can cause similar problems; if you need to change your computer's hostname, switch out of runlevel 5 and make sure X is not running(through startx),change the hostname ,and only then restart X.

Things to check for X Windows:-

1] Never debug X while in run level 5.
2] Try system-config-display first.
3] Check whether /home or /tmp directory is full or not.
4] Check whether user has reached a hard quota?
5] Is xfs service is running?

13.1 And 13. 2 Check Your Progress

1. Fill in the blanks

1] The first thing to try when confronting X problems is .....

2] Never debug X while in run level .....

3] If .....service is not running then X will not work.

4] .....command is used to view IP address of a LAN card.

5] .....command is used to enable the LAN card.

6] For host name resolution use .....command.

2. State True or False

1] if / tmp directory is full X windows will not work.

2] system-config-video command is used to set the display.

3] netstat command is used to check the routing table.

13.3 TROUBLESHOOTING : NETWORKING

Hostname resolution problems can create problems can create problems for clients and servers alike. Aside from requiring successful forward lookups, reverse lookups are essential

for many host-base security mechanisms. Tools like host and dig are invaluable for determining whether hostname resolution problem exist.
IP configuration may be checked using the ifconfig command, which will print information such as an interface's IP, the subnet mask ,and other important settings. The netstat -r and netstat-rn commands will show if a system's routing table is correct. The absence of a default gateway or the existence of multiple default gateways can create problems. Inability to contact the default gateway (and thus, to reach the gateway to get outside the local network)can also cause networking problems.
It is possible that the kernel module for your particular network interface card has been mis-specified. For example, the Red Hat Enterprise Linux installer sometimes probes a de4x5-based card as a tulip-based card. Unfortunately, the tulip module will only work enough to enable the interface, but not enough to work.
Don't overlook the obvious may be the interface has not been activated, or was deactivated for some reason.

Things to Check for Networking

1] Check LAN card is enabled or not using ifconfig command.
2] If LAN card is not enabled use ifup command to enable it.
3] Check the IP configuration of the LAN card using ifconfig command.
4] For host name resolution use dig command for e.g. dig server.example.com
5] Use netstat –r and netstat-r n commands to check routing table.

13.4 ORDER OF THE BOOT PROCESS

In order to troubleshoot boot time problems, one must understand the boot process itself ,remember how things look when they are working correctly ,and narrow down how far into the process a failure is occurring.

No boot loader splash screen or prompt appears

- GRUB is misconfigured
- Boot sector is corrupt
- A BIOS setting, such as disk addressing scheme, has been modified since the boot sector was written Kernel does not load at all ,or loads partially before a panic occurs
- Corrupt kernel image
- Incorrect parameters passed to the kernel by the boot loader

Kernel loads completely but panics or fails when it tries to mount root file system and run/sbin/init

- Boot loader is misconfigured
- /sbin/init is corrupted or /etc/inittab is misconfigured
- Root filesystem is damaged and un mountable Kernel loads completely, and/etc/rc.d/rc.sysinit is started and interrupted
- /bin/bash is missing or corrupted
- /etc/fstab may have an error –evident when file systems are mounted or fsck'ed
- Errors in software RAID or quota specifications
- Corrupted non-root file systems(due to a failed fsck) Run level errors (typically services)
- Another service required by a failing service was not configured for a given runlevel
- Service=specific errors
- Misconfigured X or related services in run level 5

13.5 FILE SYSTEM CORRUPTION

File system corruption is one of the most common boot-time problems. It can occur after a system crash causes the machine to shut down without correctly un mounting its file systems. File system corruption happens because the operating system users

RAM as disk buffers to improve performance; when power fails , information written to memory buffers which is not yet synchronized to the disk is lost.

When a device using an traditional file system such as ext2 is mounted read-write, the file system is marked on disk as “dirty”. Only when the un mount command is issued and all the data is safely written to the disk then the disk is marked as “cleaned” and un mounted. Therefore, a file system that is not mounted but is marked as “dirty” hasn’t been properly un mounted and may be corrupted. Since we don’t know what files were being written at the time of crash, the entire file system will need to be examined to repair any problems. This can take a lot of time.

The ext3 file system adds a transactional journal to ext2. Even when mounted read-write, the system is marked as “clean”. However, if at boot time the journal contains information about incomplete writes, corruption may exist. In this case, since we know exactly which files were being written at the time of crash (from the information in the journal), we can very quickly check and repair just the affected parts of the file system.

The fsck program is a front end tool to the standard file system checking files systems errors. The one for ext2/ext3, e2fsck, is used both to repair ext3 file system’s journal and to exhaustively examine file systems of either type.

If the root file system was not un mounted properly and has a journal, then when the kernel mounts the file system read-only, the kernel will read the journal and repair the file system. If rc.sysinit detects possible disk corruption, you will be presented with a prompt that reads “Press Y within 5 seconds to force file system integrity check…”. If you press Y, then all file systems marked for checking in /etc/fstab will be treated as if they were “dirty”, whether they are “dirty” or “clean”. This includes ext3 file systems. Normally it is okay to wait until the prompt times out.

Then rc.sysinit will fsck the file systems listed in /etc/fstab that have a positive integer in the sixth column. Any file systems that have journals will use them to mark rapid repairs, and any file systems marked “dirty” will be exhaustively examined and repaired. If the file system is badly corrupted, fsck may “fail” and need to be rerun manually. You must provide the root password to get a shell from which you can run fsck, or the actual programs fsck runs; e2fsck, dosfsck and so on. The e2fsck is used to repair both ext2 and ext3 file systems.

Recovery Run-levels

In recovery situations it often helpful (and depending on the problem possibly necessary) to boot to run-level where less services are active. For example, consider if you have service that causes the machine to panic each time it tries to start. In this case, the road to recovery starts by preventing the service from starting, so you can successfully boot the machine to a stable state and determine the problem with the service. The below listed run-levels are of particular importance in system recovery situations.

Run-level 1

Booting to run –level 1 will cause the system to process the /etc/rc.sysinit script followed by each of the/etc/rc.d/init.d scripts called in /etc/rc1.d/*.By default, RHEL will only call the single script in this runlevel, which after some basic checks and cleanup will exec init S. Run level 1 is also called as maintenance mode.

Run-level s, S, single

Booting to run –level 1 will cause the system to process the /etc/rc.sysnit script (if /etc/ inittab is intact) If /etc/inittab is missing or corrupt, you can still boot to single mode ,and in that case, you are given the root shell with no scripts processed.
Sometimes going to single user mode is overkill: interactive startup mode, invoked by typing “I” hen “Welcome to Red Hat Enterprise Linux” appears at boot time, allows you to choose which services will run.

Run-level emergency

While technically not a run-level emergency mode shares many characteristics of the above listed run-levels. You can only access emergency mode during boot by passing emergency as a parameter from the grub prompt. No scripts will be processed, and you are given a root shell.

13.6 RESCUE ENVIRONMENT

If the root file system is available and mountable, then you should be able to use it to fix problems that may occur. When it is not, then you must use a rescue environment. A rescue environment is a streamlined RHEL system that does not require the installed OS to run. Rather than working on the broken system itself, you work outside of the system in an environment that, while more limited than single user mode(or even slogin mode), should provide enough tools to recover root

There are several ways to boot into rescue environment:

- Boot from CDROM then type linux rescue at the isolinux prompt
- Boot from a diskboot.img USB drive ,then type linux rescue at the prompt

Rescue Environment Utilities

The rescue environment exists within a ramdisk image (referenced as /dev/root).Because of limitations on sixe and the number of inodes, many familiar utilities and device nodes are not available. However, tools related to disk maintenance (the probable reason for being in the rescue environment)and network connectivity are provided.
The following is an (incomplete) list of utilities provided by the rescue environment:

Disk Maintenance Utilities
including a complete set of LVM utilities, for managing physical volumes, volume groups, and logical volume ;software RAID tools, sap commands, disk partition utilities, file system creators, checkers, debuggers, and labelers for ext2,ext3,jfs,msdos,vfat,and resizer file systems.

Networking Utilities
including :network debuggers (ifconfig ,route, traceroute, host);network connectivity tools (ftp, rcp, rlogin)

Miscellaneous Utilities

including shell commands (bash, chroot);process management tools (ps, kill, killall);editors(vi), mtools command kernel module management commands ;archiving and compression tools(dd, tar, cpio, gzip),rpm,file manipulation commands (cd, lsmkdir, cp, mvrm).
Within the rescue environment system logging information can be found in the file /tmp/syslog. Booting information is in /tmp/anaconda.log Some configuration files (modprobe.conf, netinfo, and device files ([sh]da,loop0) are located in /tmp as well.

The rescue environment will attempt to reconstruct the hard disk's file system under the mount point /mnt/sysimage. Since the rescue environment is often used on systems with damaged or misconfigured file systems, however this operation might or might not work .A corrupted partition table will appear to hang the rescue environment (a shell with fdisk is available under Alt –F2, however).Using linux rescue non mount as the boot prompt directive disables automatic mounting of file systems and circumvents the hanging caused by bad partition tables .Careful inspection of the output of the mount command should determine the state of the reconstructed file system.

Because the standard installation provides nearly 7000 devices nodes ,administrators seldom need to create device nodes directly .In the rescue environment , device nodes

are only provided for the most basic devices ,including any fixed disks the kernel was able to autodetect.

In order to access any other devices ,such as a floppy drive ,the relevant device node must be created with mknod. Fortunately ,the rescue environment's version of mknod automatically associates the appropriate device driver major/minor numbers with well known device names. For example ,the device node for the hard disk on the secondary IDE controller can be created with mknod /dev/hdc.

13.5 ,13.6 Check Your Progress

Fill in the blanks

1] ..... is one of the most common boot-time problems.

2] The ..... program is a front end tool to the standard file system checking files systems errors.

3] If root file system is not available then you must use .....

4] The rescue environment will attempt to reconstruct the hard disk's file system under the mount point .....

State True or False

1] Run level 1 is also called as maintenance mode.

2] Rescue environment requires installed operating system to run.

3] In order to access any other devices ,such as a floppy drive, the relevant device node must be created with mknod

13.7 SUMMARY

The Process of Troubleshooting any system including running RHEL is both Art & Science . Hostname resolution problems can creat problems for clients and servers. IP configuration can be checked using ipconfig command.

File system corrouption is one of the most common boot time problem . It can occur after a system crash problem. If root file system is avialable and mountable, then you should be able to use it to fix problems.

13.8 CHECK YOUR PROGRESS-ANSWERS

13.1 -13.2

1) 1] system-config-display
2] 5
3] xfs
4] ifconfig
5] ifup
6] dig

2) 1] True
2] False
3] True

13.5 - 13.6

1) 1] File system corruption.
2] fsck
3] Rescue Environment.
4] /mnt/sysimage

2) 1] True
2] False
3] True

13.9 QUESTIONS FOR SELF-STUDY

1. List the types of Troubleshooting & explain any one in your language.
2. Explain what rescue environment in your language is.
3. Explain file system corruption in detail.

13.10 SUGGESTED READINGS

1. The Complete Linux Reference by Christopher Negus
2. UNIX Concepts and Application by Sumitabha Das
3. Begining Linux Programming by Christopher Negus