SECTION 2 INTRODUCTION TO LINUX

Structure **Page Nos.**

2.0 Introduction 19
2.1 Objectives 20
2.2 Linux Operating System 20
2.3 Exploring Desktop 22
2.4 Using the Shell 24
2.5 Understanding Users and File Systems 24
2.6 Linux Utilities and Basic Commands 27
2.7 Understanding Text Processing 32
2.8 Managing Processes 34
2.9 Summary 35
2.10 References/Further Readings 35

2.0 INTRODUCTION

Linux is a free open-source operating system based on UNIX. Linus Torvalds originally created Linux with the assistance of developers from around the world. Exactingly speaking, Linux is a kernel. A kernel provides access to the computer hardware and controls access to resources. The kernel decides who will use a resource, for how long and when. You can download the Linux kernel from the official web site.

However, the Linux kernel itself is useless unless you get all the applications such as text editors, email clients, browsers, office applications, etc. Therefore, someone came up with idea of a Linux distribution. A typical Linux distribution includes:

- Linux kernel
- GNU application utilities such as text editors, browsers
- GUI (X windows)
- Office application software
- Software development tools and compilers
- Thousands of ready to use application software packages
- Linux Installation programs/scripts
- Linux post installation management tools daily work such as adding users, installing applications, etc

Linux is renowned for being both powerful and also secure. As a possible open source method Linux may be hard-wired and also enhancing by many people, several computer programmers. It's authorized Linux to be able to “evolve” extra time, developing with each development from the developer or even improvement organization. Whilst Linux began like a programmer’s just operating-system it's becoming increasingly well-liked in the customer degree. Linux can also be able to handle powerful equipment without flinching, and that's why it's a preferred amongst supercomputers and also machines.

In this section, we are going to explore the practical process related to the loading and installing of Proprietary and Open Source operating system, so far as the Proprietary operating system is concerned we are going to discuss for Windows XP and on the end of Open source operating system we are going to work with Fedora. Before beginning the actual technicalities, Let us discuss some fundamentals of operating system.

2.1 OBJECTIVES

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

- list the features of Linux;
- know the components of Linux Desktop and the Shell;
- understand Linux users and file systems
- know the Utilities and Basic Commands of Linux; and
- understanding text processing and process management issues

2.2 LINUX OPERATING SYSTEM

Linux is a freely available, open source, Unix-like operating system. Written originally for the PC by Linus Torvalds, with the help of many other developers across the Internet, Linux now runs on multiple hardware platforms. Because of its speed, stability, and low cost, Linux became the fastest growing operating system for servers. Today, Linux is widely used for both basic home and office uses. It is the main operating system used for high performance business and in web servers. Linux has made a huge impact in this world.

Red Hat Linux

The first public release of Red Hat Linux (version 1.0) is dated 1994, after that there are many versions of Red Hat Linux 1.1, 2.0, 2.1, 3.0.3, 4.0, 4.1, 4.2, 5.0, 5.1, 5.2, 6.0, 6.1, 6.2, 7, 7.1, 7.2, 7.3, 8.0 and 9 over a period till the year 2003. After that Redhat and Fedora project were merged.

Fedora

Fedora is a Linux-based operating system that showcases the latest in free and open source software. Fedora is always free for anyone to use, modify, and distribute. It is built by people across the globe who work together as a community: the Fedora Project. The Fedora Project is open and anyone is welcome to join.

Fedora Core 1 was the first version of Fedora and was released in the year 2003. Since, then different versions of Fedora such as Fedora Core 2, 3, 4, 5 and 6 were released till 2006. Then new versions such as Fedora 7, 8, 9 and 10 are also released till 2008.

2.2.1 Features of Linux Operating System

The following are various features of Linux operating system:

Low Cost

There is no need to spend time and huge amount money to obtain licenses since Linux and much of its software come with the GNU General Public License. There is no need to worry about any software that you use in Linux.

Stability

Linux has high stability compared with other operating systems. There is no need to reboot the Linux system to maintain performance levels rarely. Its freezes up or slow down. It has a continuous up-times of hundreds of days or more.

Performance

Linux provides high performance on various networks. It has the ability to handle large numbers of users simultaneously.

Networking

Linux provides a strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks like network backup faster than other operating systems.

Flexibility

Linux is very flexible. Linux can be used for high performance server applications, desktop applications, and embedded systems. You can install only the needed components for a particular use. You can also restrict the use of specific computers.

Compatibility

It runs all common Unix software packages and can process all common file formats.

Fast and Easy Installation

Linux distributions come with user-friendly installation.

Better use of Hard Disk

Linux uses its resources well enough even when the hard disk is almost full.

Multitasking

Linux is a multitasking operating system. It can handle many things at the same time.

Open Source

Linux is an Open source operating systems. You can easily get the source code for Linux and edit it to develop your personal operating system.

Powerful Kernel

The kernel is heart of the Linux operating system. It manages the resources of Linux. Resources include:

- File management
- Multitasking
- Memory management
- I/O management
- Process management
- Device management
- Networking support including IPv4 and IPv6

- Advanced features such as virtual memory, shared libraries, demand loading, shared copy-on-write executables etc

The kernel decides who will use these resources and for how long and when. It runs your programs or sets up to execute binary files. The kernel acts as an intermediary between the computer hardware and various applications.

2.3 EXPLORING DESKTOP

A **desktop** essentially refers to a graphical user interface (GUI) and applications which permit the user of the OS to interact with the system (the OS and applications) graphically using familiar keyboard and mouse commands. Unlike in the Windows world, where you essentially end up with the Windows desktop when you install Windows, in the Linux world there are different desktops that one can install and use. Historically, the two most popular were GNOME20 (pronounced either "guh nome" or simply "nome") and KDE21. GNOME is part of the GNU project and is based on the GTK+ graphical user interface toolkit/library; it was first released in 1996, and since it is now 17 years old, it is a quite mature environment. GNOME 2 was widely supported by all the major Linux distros, but with the release of GNOME 3, fan support has dropped, as the familiar desktop metaphor has been replaced by something called the Gnome Shell22—which many former GNOME users are rebelling against, for various reasons23. Nonetheless, many popular distros ship with GNOME 3 including Fedora, Mint 12, openSUSE 12.1, Arch, OpenBSD, and Debian. KDE is similar in functionality to GNOME and was first released a couple of years later in 1998. It is based on a different GUI toolkit called Qt (pronounced "cute")The desktop of Linux is termed GNOME. GNOME stands for GNU Network Object Model Environment; and GNU, as you probably know, stands for GNU's Not Unix. GNOME is a GUI and a programming environment. From the user's perspective, GNOME is like the Motif-based Common Desktop Environment (CDE) or Microsoft Windows. Behind the scenes, GNOME has many features that enable programmers to write graphical applications that work together well. The next few sections provide a quick overview of GNOME, highlighting the key features of the GNOME GUI. You can explore the details on your own.

Taking Stock of GNOME

Although what you see of GNOME is the GUI, there is much more to GNOME than the GUI. To help you appreciate it better, note the following points and key features of GNOME:

- GNOME runs on several UNIX systems, including Linux, BSD (FreeBSD, NetBSD, and OpenBSD), Solaris, HP-UX, AIX, and Silicon Graphics IRIX.

- GNOME uses the Gimp Tool Kit (GTK+) as the graphics toolkit for all graphical applications. GTK+ relies on X for output and supports multiple programming languages, including C, C++, Perl, and others. Users can easily change the look and feel of all GTK+ applications running on a system. GTK+ is licensed under the GNU Library General Public License (LGPL). See Chapter 23 for more information about GPL and LGPL.

- GNOME also uses Imlib, another library that supports displaying images on an X display. Imlib supports many different image formats, including XPM and PNG (Portable Network Graphics).

- GNOME uses the Object Management Group's Common Object Request Broker Architecture (CORBA) Version 2.2 to enable GNOME software components to communicate with one another regardless of where the components are located or what programming language is used to implement the components.

- GNOME applications support drag-and-drop operations.

- GNOME application developers write documentation using DocBook, which is a Document Type Definition (DTD) based on Standard General Markup Language (SGML). This means that you can view the manual for a GNOME application on a Web browser, such as Mozilla.

- GNOME supports standard internationalization and localization methods. This means that you can easily configure a GNOME application for a new native language.

Exploring GNOME

Assuming that you have enabled a graphical login screen during Red Hat Linux installation, you should get the GNOME GUI whenever you log in to your Linux system. The exact appearance of the GNOME display depends on the current session. The session is nothing more than the set of applications (including a window manager) and the state of these applications. The metacity window manager, which is the default window manager in GNOME, stores the session information in files located in the ~/.metacity/sessions directory (this is in your home directory). The session files are text files; if you are curious, you can browse these file with the more command.

Initially, when you don't yet have a session file, the GNOME session comes from the /usr/share/gnome/default.session file. However, as soon as the session starts, the GNOME session manager (/usr/bin/gnome-session) saves the current session information in the~/.metacity/sessions directory. A typical initial GNOME desktop produced by the session description in the default session file is shown in *Figure 1*.

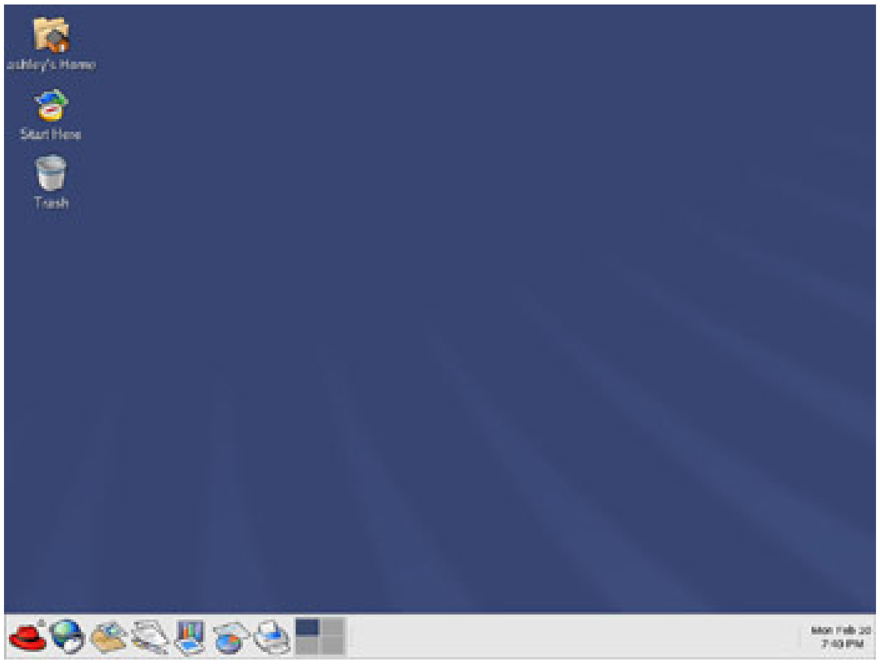

Figure 1: The Initial GNOME Desktop, with the Default Session File.

As you can see from the icons on the left side of the GNOME desktop, GNOME enables you to place folders and applications directly on the desktop. This is similar to the way in which you can place icons directly on the Microsoft Windows desktop.

2.4 USING THE SHELL

Computer understands the language of 0's and 1's called binary language.

In early days of computing, instruction are provided using binary language, which is difficult for all of us, to read and write. So in Os there is special program called Shell. Shell accepts your instruction or commands in English (mostly) and if it's a valid command, it is passed to kernel.

Shell is a user program or its environment provided for user interaction. Shell is a command language interpreter that executes commands read from the standard input device (keyboard) or from a file.

Shell is not part of system kernel, but uses the system kernel to execute programs, create files etc.

Several shell available with Linux including:

BASH(Bourne Again Shell), CSH(C Shell), KSH(Korn Shell) and TCSH

Tip: To find all available shells in your system type following command:
$ cat /etc/shells

Note that each shell does the same job, but each understand a different command syntax and provides different built-in functions.

In MS-DOS, Shell name is COMMAND.COM which is also used for same purpose, but it's not as powerful as our Linux Shells are!

2.5 UNDERSTANDING USERS AND FILE SYSTEMS

This subsection describes how the Linux kernel maintains the files in the file systems that it supports. It describes the Virtual File System (VFS) and explains how the Linux kernel's real file systems are supported.

One of the most important features of Linux is its support for many different file systems. This makes it very flexible and well able to coexist with many other operating systems. At the time of writing, Linux supports 15 file systems; ext, ext2, xia, minix, umsdos, msdos, vfat, proc, smb, ncp, iso9660, sysv, hp fs, affs and ufs, and no doubt, over time more will be added.

In Linux, as it is for Unix ^{TM}, the separate file systems the system may use are not accessed by device identifiers (such as a drive number or a drive name) but instead they are combined into a single hierarchical tree structure that represents the file system as one whole single entity. Linux adds each new file system into this single file system tree as it is mounted. All file systems, of whatever type, are mounted onto a directory and the files of the mounted file system cover up the existing contents of that directory. This directory is known as the mount directory or mount point. When the file system is unmounted, the mount directory's own files are once again revealed.

When disks are initialized (using fdisk, say) they have a partition structure imposed on them that divides the physical disk into a number of logical partitions. Each partition may hold a single file system, for example an EXT2file system. File systems organize files into logical hierarchical structures with directories, soft links and so on held in blocks on physical devices. Devices that can contain file systems are known as block devices. The IDE disk partition /dev/hda1, the first partition of the first IDE disk drive in the system, is a block device. The Linux file systems regard these block devices as simply linear collections of blocks, they do not know or care about the underlying physical disk's geometry. It is the task of each block device driver to map a request to read a particular block of its device into terms meaningful to its device; the particular track, sector and cylinder of its hard disk where the block is kept. A file system has to look, feel and operate in the same way no matter what device is holding it. Moreover, using Linux's file systems, it does not matter (at least to the system user) that these different file systems are on different physical media controlled by different hardware controllers. The file system might not even be on the local system, it could just as well be a disk remotely mounted over a network link. Consider the following example where a Linux system has its root file system on a SCSI disk:

```
A     E     boot    etc    lib    opt    tmp    usr
C     F     cdrom   fd     proc   root   var    sbin
D     bin   dev     home   mnt    lost+found
```

Neither the users nor the programs that operate on the files themselves need know that /C is in fact a mounted VFAT file system that is on the first IDE disk in the system. In the example (which is actually my home Linux system), /E is the master IDE disk on the second IDE controller. It does not matter either that the first IDE controller is a PCI controller and that the second is an ISA controller which also controls the IDE CDROM. I can dial into the network where I work using a modem and the PPP network protocol using a modem and in this case I can remotely mount my Alpha AXP Linux system's file systems on /mnt/remote.

The files in a file system are collections of data; the file holding the sources to this chapter is an ASCII file called filesystems.tex. A file system not only holds the data that is contained within the files of the file system but also the structure of the file system. It holds all of the information that Linux users and processes see as files, directories soft links, file protection information and so on. Moreover, it must hold that information safely and securely, the basic integrity of the operating system depends on its file systems. Nobody would use an operating system that randomly lost data and files[1].

Minix, the first file system that Linux had is rather restrictive and lacking in performance.

Its filenames cannot be longer than 14 characters (which is still better than 8.3 filenames) and the maximum file size is 64MBytes. 64Mbytes might at first glance seem large enough but large file sizes are necessary to hold even modest databases. The first file system designed specifically for Linux, the Extended File system, or EXT, was introduced in April 1992 and cured a lot of the problems but it was still felt to lack performance.

So, in 1993, the Second Extended File system, or EXT2, was added.

An important development took place when the EXT file system was added into Linux. The real file systems were separated from the operating system and system services by an interface layer known as the Virtual File system, or VFS.

VFS allows Linux to support many, often very different, file systems, each presenting a common software interface to the VFS. All of the details of the Linux file systems are translated by software so that all file systems appear identical to the rest of the Linux kernel and to programs running in the system. Linux's Virtual File system layer allows you to transparently mount the many different file systems at the same time.

The Linux Virtual File system is implemented so that access to its files is as fast and efficient as possible. It must also make sure that the files and their data are kept correctly. These two requirements can be at odds with each other. The Linux VFS caches information in memory from each file system as it is mounted and used.

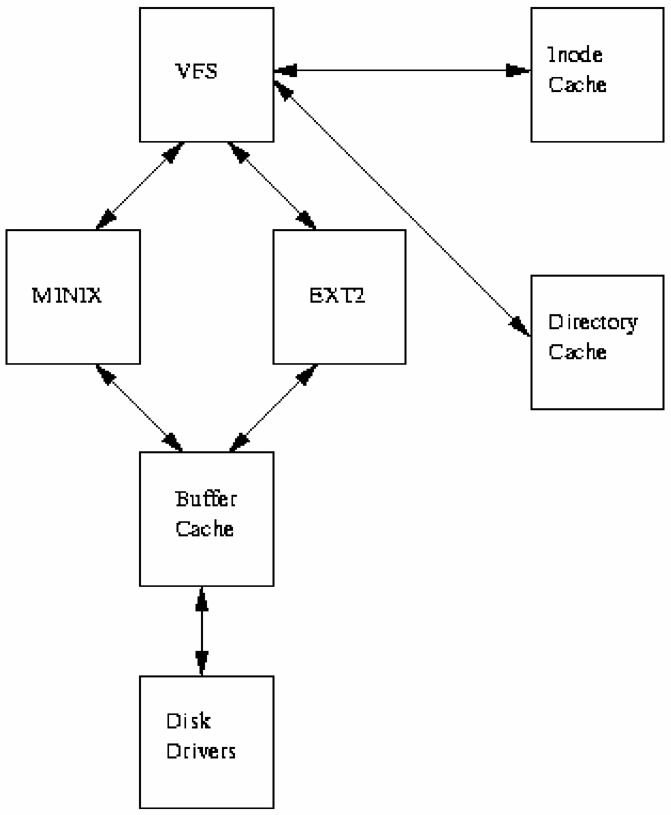

Figure 2: A Logical Diagram of the Virtual File System

The figure 2 shows the relationship between the Linux kernel's Virtual File System and it's real file systems. The virtual file system must manage all of the different file systems that are mounted at any given time. To do this it maintains data structures that describe the whole (virtual) file system and the real, mounted, file systems.

The VFS Superblock

Every mounted file system is represented by a VFS superblock; amongst other information, the VFS superblock contains the:

Device

This is the device identifier for the block device that this file system is contained in. For example, /dev/hda1, the first IDE hard disk in the system has a device identifier of *0x301*,

Inode pointers

The mounted inode pointer points at the first inode in this file system. The covered inode pointer points at the inode representing the directory that this file system is mounted on. The root file system's VFS superblock does not have a covered pointer,

Blocksize

The block size in bytes of this file system, for example 1024 bytes,

Superblock operations

A pointer to a set of superblock routines for this file system. Amongst other things, these routines are used by the VFS to read and write inodes and superblocks.

File System type

A pointer to the mounted file system's file_system_type data structure,

File System specific

A pointer to information needed by this file system,

The Linux Operating System has several user accounts. A few are preconfigured and an option of defining new is also available.

Preconfigured user accounts

- Root
  - Administrative account
  - Also called superuser
  - Can perform any operation on Linux system
  - Do not log in as root for normal work
  - Change temporarily to root user
- Regular user accounts
  - Users who log in at keyboard and use Linux system
  - Commonly associated with named individuals
- Special user account
  - Used by Linux programs
  - Created during installation of Linux
  - Vary depending on services installed

User information commands:

- id command
  - Shows effective UID
- logname command
  - View user name that you used to log in
- whoami command
  - Shows user name of currently effective UID
- groups command
  - Lists all groups you are a member of

2.6 LINUX UTILITIES AND BASIC COMMANDS

The following are some of the basic commands used in the CLI Linux operating system, however GUI option is always available, as in the case of Windows:

a) **alias** Alias is used to substitute a small or more familiar name in place of a long string. It is commonly used for long strings that are frequently used.

Syntax alias [name=['command']]

Name	Specifies the alias name.
Command	Specifies the command the name should be an alias for.
-a	Removes all alias definitions from the current shell execution environment.
-t	Sets and lists tracked aliases.
-x	Sets or prints exported aliases. An exported alias is defined for scripts invoked by name.

Example $alias home 'cd public_html' - Sets home to type cd public_html. Use the command ‘unalias’ to remove this alias.

Alias command can be used even for the following:

$alias clr clear	$alias cls clear
$alias copy cp –I	$alias del rm –I
$alias delete rm –I	$alias home cd ~
$alias md mkdir	$alias move mv –I
$alias type more	

b) awk awk utility is powerful data manipulation/scripting programming language (In fact based on the C programming Language). Use awk to handle complex task such as calculation, database handling, report creation etc.

Syntax awk -f {awk program file} filename

awk Program contains are something as follows:

```
Pattern{
     action1
     action2
     actionN
}
```

awk reads the input from given file (or from stdin also) one line at a time, then each line is compared with pattern. If pattern is match for each line then given action is taken. Pattern can be regular expressions.

Following is the summery of common awk metacharacters:

Metacharacter	Meaning
(Dot)	Match any character
*	Match zero or more character
^	Match beginning of line
$	Match end of line
\	Escape character following
[ ]	List
{ }	Match range of instance
+	Match one more preceding
?	Match zero or one preceding
|	Separate choices to match

c) cd The cd sets the working directory of a process.

Syntax cd <directory name>

Example $cd /etc

d) chmod Chmod is a utility that changes the permission of a file.

Syntax *chmod [OPTION]... MODE[,MODE]... FILE...*

Permissions

u - User who owns the file.	*g* - Group that owns the file.
o - Other.	*a* - All.
r - Read the file.	*w* - Write or edit the file.
x - Execute or run the file as a program.	

Numeric Permissions:

CHMOD can also to attributed by using Numeric Permissions:

400 read by owner	040 read by group	004 read by anybody
200 write by owner	020 write by group	002 write by anybody
100 execute by owner	010 execute by group	001 execute by anybody

Example $chmod 644 file.htm

This gives the file read/write by the owner and only read by everyone else (-rw-r--r--).

$chmod 755 file.cgi

$chmod 666 file.txt

e) chown Chown is a utility that is also used to change file ownership.

Syntax chown [R] *owner*[:*group*] *file* ...

-R recursively change file user and group IDs. For each *file* operand that names a directory, *chown* shall change the user ID (and group ID, if specified) of the directory and all files in the file hierarchy below it.

Example #chown root file.txt

f) cp The cp command is used to copy files.

Syntax $cp source destination

Example $cp abc.txt pqr.txt [copy the file abc.txt as pqr.txt]

$cp *.txt /etc/ [copy all files with .txt extension into /etc directory]

$cp a* /etc/ [copy all files starts with the letter 'a' into /etc directory]

g) date An essential command to set the date and time. Also a useful way to output current information when working in a script file.

Example $date

h) df The df command reports filesystem disk space usage.

With no arguments, 'df' reports the space used and available on all currently mounted filesystems (of all types). Otherwise, `df' reports on the filesystem containing each argument *file*.

Syntax df [*option*]... [*file*]...

Normally the disk space is printed in units of 1024 bytes, but this can be overridden.

OPTIONS **'-i'** List inode usage information instead of block usage. An inode (short for index node) is contains information about a file such as its owner, permissions, timestamps, and location on the disk.

'-k' Print sizes in 1024byte blocks, overriding the default block size.

'-m' Print sizes in megabyte (i.e.; 1,048,576-byte) blocks.

Example $df –k $df –m $df –i

i) pwd To know the current working directory

Syntax pwd

Example $pwd

j) ln The ln command makes new, alternate file names for a file by hard linking, letting multiple users share one file. The ln command creates pseudonyms for files which allows them to be accessed by different names. These pseudonyms are called links. There are two different forms of the command and two different kinds of links that can be created.

Syntax **ln** [*options*] *exiting_path* [*new_path*]

ln [*options*] *exiting_paths directory*

In the first form, a new name is created called *new_path* which is a pseudonym for *existing_path*. In the second form, the last argument is taken to be a directory name and all the other arguments are paths to existing files. A link for each existing file is created in the specified directory with the same filename as the existing files.

Example Create a link named my_file in the current directory to the file /home/bill/his_file:

$ln /home/bill/his_file my_file

As above but the link is created in /home/joe/my_file:

$ln /home/bill/his_file /home/joe/my_file

To create a symbolic link, all works as above except you need to include the -s option.

For example, to make a symbolic link called Linux that points to fedora:

$ln -s Linux fedora

The only way to see that Linux is a symbolic link is by using the ls -l command (ls -l Linux). The output of this command will look much like this:

```
lrwxrwxrwx  1 joe users    3 2009-03-21 17:26 Linux -> fedora
```

k) ls The ls command shows information about files. It lists the contents of a directory in order to determine when the configurations files were last edited.

Syntax ls [-a] [-A] [-c] [-d] [-i] [-l] [-L] [n] [-r] [-R] [pathnames]

-a: shows all files, even files that are hidden (these files begin with a dot.)

-A: list all files including the hidden files. However, does not display the working directory (.) or the parent directory (..).

-c: use time of last modification of the i-node (file created, mode changed, and so forth) for sorting (-t) or printing (-l or -n).

-d: if an argument is a directory it only lists its name not its contents.

-i: for each file, print the i-node number in the first column of the report.

-l: shows you huge amounts of information (permissions, owners, size, and when last modified.)

-L: if an argument is a symbolic link, list the file or directory the link references rather than the link itself.

-n: The same as -l, except that the owner's UID and group's GID numbers are printed, rather than the associated character strings.

-r: reverses the order of how the files are displayed.

-R: includes the contents of subdirectories.

Example $ls –l $ls –d $ls -r

l) man Short for "manual," man displays information about commands and a keyword search mechanism for needed commands.

Syntax man <command>

Example $man find $man chown

m) passwd A quick and easy way to change passwords on a system.

Syntax passwd [-d account name]

Example #passwd

#passwd –d xyz [to delete the password of the account 'xyz']

n) Shutdown Shutdown is a command that turns off the computer and can be combined with variables such as -h for halt or -r for reboot.

Syntax Shutdown [-h][-r]

Example #shutdown

o) top Top provides an ongoing look at processor activity in real time. It

displays a listing of the most CPU-intensive tasks on the system, and can provide an interactive interface for manipulating processes.

Syntax top

Example $top

p) vmstat The vmstat command is used to get a snapshot of everything in a system, helping admins determine whether the bottleneck is CPU, memory or I/O. Run this command to get virtual memory statistics. vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity.

Syntax	vmstat [-a] [-n] [*delay* [*count*]]
	vmstat [-f] [-s] [-m]
	vmstat [-S unit]
	vmstat [-d]
	vmstat [-p disk partition]
	vmstat [-V]
Example	$vmstat

2.7 UNDERSTANDING TEXT PROCESSING

The most commonly used text editor for Linux is vi. and its version that is usually found in PCs is vim. Following is detailed user guide for using vi editor for text processing. The vi editor has two modes:- editing mode and command mode.

Starting vi

At the Bash command prompt, type vi or vi some-filename. vi is a modal editor in which you are either in editing mode or command mode.

Getting out of vi

This is the most important thing you need to know about vi: how to get out of it. If you are in editing mode, hit ESC to enter command mode. Enter :wq to write your file and quit. To quit without writing the file, hit :q. If you really want to quit without saving the file you have made changes to hit :q!. To save the file under a different name, try :wq new-filename.

Switching Between Editing Mode and Command mode:

If you are in editing mode, hit ESC to enter command mode. If you are in command mode hit i to enter insert (editing) mode. If you cannot tell which mode you are in, trying hitting ESC several times to make sure you are in command mode, then type :set showmode. This may tell vi to let you know when you are in insert mode (it depends on the version of vi you are using).

Moving Around a Character at a Time

The original vi was written by Bill Joy (at UC Berkeley) using a computer system and keyboard that lacked arrow, page up, and page down keys. This is the reason for the stupid assignment of the keys for moving around: h (move left one character), j (move down one line), k (move up one line), and l (move right one character). On Joy's terminal, these four keys also had arrow keys on them and that is the historical reason they are still mapped that way today. However, if you have your terminal program, i.e., PuTTY, configured correctly, you should be able to use the arrow keys.

Moving Around a Word at a Time

Switch to command mode. Press w to move forward one word. Press b to move backward one word. Press nw to move forward n words, e.g., 3w to move forward three words. Press nw to move backward n words.

Moving Around the File

Switch to command mode. To move to the end of the line, hit $. To move to the beginning, hit 0. Hit 1G to go the first line of text. Hit nG to go line number n. Hit G to go to the end of file. To display line numbers, in command mode type :set number. Note that:13 is equivalent to 13G. To page down, hit Ctrl+F. To page up hit Ctrl+B.

Deleting Characters

Switch to command mode, move to the character you want to delete, and hit x to delete that character. To delete n characters hit nx, e.g., 17x to delete 17 characters. To delete all the characters from the cursor position to the end of the line hit D.

Copying a Line of Text

To copy a line of text from one place in the file to another, switch to command mode. Move to the line you want to copy. Hit yy (yank something or other, I dunno). Move the cursor to the location where you want to make the copy. Hit p (lowercase p for paste) to put the copied line after the current line. Hit P (uppercase P) to put the copied line before the current line.

Moving a Line of Text

To move a line of text from one place in the file to another, switch to command mode. Move to the line you want to copy. Hit dd. Move the cursor to the location where you want to make the move. Hit p (lowercase p) to move the line after the current line. Hit P (uppercase P) to move the line before the current line.

Deleting Lines of Text

Switch to command mode. Move to the line you want to delete. Hit dd. Hit ndd to delete n lines of text, e.g., 3dd will delete the line of text the cursor is on and the next two lines.

Cutting, Copying, Pasting Multiple Lines of Text

Use nyy to copy n lines of text to the copy buffer. Move to where you want the lines to be and hit p or P to copy them. Use ndd to delete n lines of text to the copy buffer, and then move to where you want to paste the lines and hit p or P.

Moving and Copying Text in Visual Mode

To cut, copy, and paste text that is not an entire line, enter visual mode by hitting v in command mode. Use the arrow keys to move around and select the text you want to copy or move. Once selected, hit y to copy the text to the copy buffer or d to delete the text to the buffer. Once copied (with y) or deleted with (with d) move to where you want the text to be. Hit p or P to paste the text.

Finding Text

Switch to command mode. Hit / (forward slash) and enter the string you are searching for, e.g., /cookies. The cursor will be placed on the first occurrence of cookies following the current location. Hitting n or Enter over-and-over will continue searching.

Find and Replace

Switch to command mode. To replace all occurrences of "homer" by "bart", try something like :1,100 s/homer/bart/g. The 1,100 part means search from line 1 to line 100 inclusive (I think 1,$ will search the entire file). The s stands for search. We are searching for "homer" and replacing all occurrences by "bart". The g stands for global which means it will replace all occurrences without prompting you for each one. If you want to be prompted for each one, use c instead (for confirmation).

Undo and Redo

Since you will use it the most because of vi's horrible user interface the most useful command in vi (other than :q!) is the undo command. Hitting u will undo your last change. Hitting u multiple times or using nu will perform multiple undos. Redo is Ctrl+R.

More Help

Let's say you're either insane or a masochist and want to learn more about vi. In command mode, hit :h to bring up help. The most useful thing you need to know here is how to get out of help, hit :q (like in less).

2.8 MANAGING PROCESSES

Process is kind of program or task carried out by your PC. For e.g.

$ ls -lR

ls command or a request to list files in a directory and all subdirectory in your current directory - It is a process.

Process defined as:

"*A process is program (command given by user) to perform specific Job. In Linux when you start process, it gives a number to process (called PID or process-id), PID starts from 0 to 65535.*"

Linux is multi-user, multitasking Os. It means you can run more than two process simultaneously if you wish. For e.g. To find how many files do you have on your system you may give command like:

$ ls / -R | wc -l

This command will take lot of time to search all files on your system. So you can run such command in Background or simultaneously by giving command like

$ ls / -R | wc -l &

The **ampersand** (**&**) at the end of command tells shells start process (**ls / -R | wc -l**) and run it in background takes next command immediately.

Process & PID defined as:

"*An instance of running command is called **process** and the number printed by shell is called **process-id (PID)**, this PID can be use to refer specific running process.*"

Table 1 shows most commonly used command(s) with process:

Table 1: Commonly used command

For this purpose	Use this Command	Examples*
To see currently running process	ps	**$ ps**
To stop any process by PID i.e. to kill process	kill {PID}	**$ kill 1012**
To stop processes by name i.e. to kill process	killall {Process-name}	**$ killall httpd**
To get information about all running process	ps -ag	**$ ps –ag**
To stop all process except your shell	kill 0	**$ kill 0**
For background processing (With &, use to put particular command and program in background)	Linux-command &	**$ ls / -R | wc -l &**
To display the owner of the processes along with the processes	ps aux	**$ ps aux**
To see if a particular process is running or not. For this purpose you have to use ps command in	ps ax | grep process-U-want-to see	For e.g. you want to see whether Apache web server

combination with the grep command		process is running or not then give command **$ ps ax | grep httpd**
To see currently running processes and other information like memory and CPU usage with real time updates.	top See the output of top command.	**$ top** **Note** that to exit from top command press q.
To display a tree of processes	pstree	**$ pstree**

* To run some of this command you need to be root or equivalent user.

Note that you can only kill process which are created by yourself. A Administrator can almost kill 95-98% process. But some process can not be killed, such as VDU Process.

2.9 SUMMARY

The section widely covers the introduction of Linux operating system. This familiarizes the reader with its potential and its usefulness. The section gives an enriched source of action for practical learning of text editor vi and the process management.

2.10 REFERENCES/FURTHER READINGS

- http://www.linux.org
- http://www.fedoraproject.org
- http://www.redhat.com/docs/manuals/linux
- UNIX and Linux System Administration Handbook, Prentice Hall