UNIX COMMANDS

who Command

who ↵ It displays data about all the users ho have logged into the system currently.
who am i ↵ It displays only data about the user of that terminal, from which this command is being executed.

ls Command

ls ↵ It gives the listing of all files and directories of the present directory.
ls –l ↵ It gives the listing of all files and directories(line wise) with the following informations-

grant permissions, file name, size of the file, date and time of last modification.

ls –lu ↵ It prints everything like the above command except the last access time.
ls –a ↵ It just gives the listing of all files, directories and as well as the hidden files of the

current directory.

ls –f ↵ It only gives the listing of all files in the present directory.
ls –ld ↵ It only gives the listing of all directories in the present directory.
ls p* ↵ It gives the listing of all the files and directories starts with 'p'.

ls ?ain ↵ It gives the listing of all files and directories ends with 'ain' and starts with any one

character [Ex- gain, rain, pain etc]

ls ??ain ↵ It gives the listing of all files and directories ends with 'ain' and starts with any one

or two characters [Ex- gain, rain, pain, train, etc]

ls s?? ↵ It gives the listing of all files and directories starts with 's' and ends with any one or

two characters.

ls [abc]* ↵ It gives the filenames and directories whose first letter can be any one of the letters among a,b and c.

ls [!abc]* ↵ It complements the above condition. It would list all those files and directories whose first letter can be anything other than the letters in the bracket.

ls [a-c] [f-l] [m-p] ↵ It would list the filenames having only 3 characters, whose first letter is in the range a to c, the second letter is in the range f to l and the third letter is in the
range m-p.

ls [a-e][p-s]??? ↵ It would list the filename having only 5 characters, whose first letter is in the range a to e, the second letter is in the range p to s, and the remaining three characters can be any valid character. [For example, apple, epic etc].

ls /mydir/x* ↵ It gives all the files and directories starting with x in the directory 'mydir'(not the present directory).

ls *.c ↵	It gives the listing of all the c files in the present directory.
ls –i <filename> ↵	It gives the inode number of the specified file.

mkdir Command

mkdir dir1↵	It creates a directory named dir1 in the present directory.
mkdir dir1 dir2 dir3 ↵	It creates three directories dir1, dir2 and dir3 in the present directory.
mkdir –p World/Asia/India ↵	It first creates the directory 'World' in the present directory, then its creates the sub-directory 'Asia' under 'World' and then creates the sub-directory 'India' under 'Asia'.
mkdir –m 754 mydir ↵	It creates a directory named 'mydir' with the permissions 754.

cd Command

cd mydir ↵	It changes to a new directory 'mydir'.
cd.. ↵	Back to one directory up.

rmdir Command

rmdir mydir ↵	It removes the directory 'mydir'.
rmdir World/Asia ↵	It removes the sub-directory 'Asia' under the directory World.
rmdir –p World/Asia/India ↵	It removes directories recursively. First 'India', then 'Asia' and then 'World'.

touch Command

touch file ↵	It creates a zero byte file named 'file'..
touch file1 file2 file3 ↵	It creates three zero byte files named 'file1', 'file2' and 'file3'.

cat Command

cat> city ↵	It creates a file named 'city' . Write something and Press Ctrl+d to save the file and exit.
cat city ↵	It gives the content of the specified file 'city'.
cat >>city ↵	It appends the lines which will be written after hitting the Enter key in to specified file 'city'. Then Press Ctrl+d to save the file and exit.

rm Command

rm command is used to remove a file.

rm –i city ↵	User is asked for a confirmation before deleting a file.
rm –r city ↵	It recursively deletes a file/directory. To remove a directory it first removes the contents of a directory and then the specified directory.
rm –f city ↵	It forcefully deletes a file.

cp Command

cp file1 file2 ↵ — It copies the contents of the source file 'file1' into the target file 'file2'. It the target file doesn't exist then it creates it.

mv Command

mv file1 file2 ↵ — It moves the contents of the source file 'file1' into the target file 'file2' and also destroys the target file 'file2'.

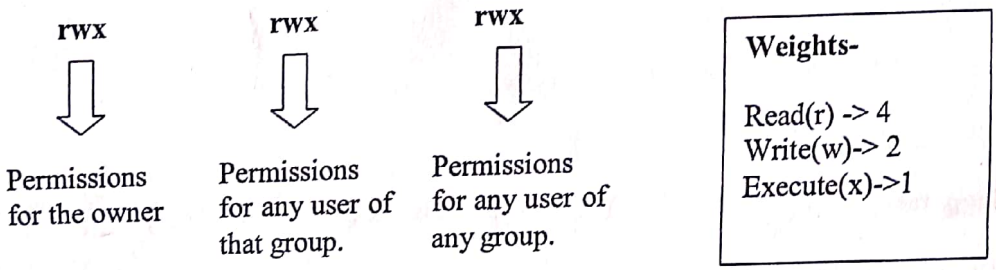
mv file1 file2 myfile ↵ — It moves the contents of both the files 'file1' and 'file2' into a target file 'myfile'.

ln Command

ln poem file ↵ — It creates one more link 'file' for the file 'poem'.

chmod Command

Permissions of a file-



chmod 742 myfile ↵ — It changes the existing file permissions to 742 to the file 'myfile'.

chmod +w myfile ↵
chmod go-x myfile↵ — It gives the write permission to all for the file 'myfile'. It takes away the execute permission from others as well as group from the file 'myfile'.

chmod go +r, go –w myfile ↵ — It gives a read permission to group and others and takes away their write permission from group and others for the file 'myfile'.

chmod go=r, u=rw myfile ↵ — It takes away all existing permissions and replaces them with only read permission for group and others and give the read & write permission to the user for the file 'myfile'.

bc Command

It does the mathematical calculation.

bc ↵
10/2*2 ↵
10 ↵
quit

bc ↵
ibase=2 ↵
obase=16 ↵
1111↵
F ↵
quit

bc ↵
sqrt(64) ↵
8 ↵
quit

bc ↵
for(i=1;i<=4;i=i++)i ↵
1
2
3
4
quit

pwd Command

pwd ↵ It gives present working directory.

expr Command

It does the mathematical calculation also. It can handle simple expressions easily.

expr 100 + 50 ↵
150

expr 100*2 ↵
50

factor Command

It calculate the prime factors. Once it printed one, then it waits for another input.

factor ↵
15 ↵
3
5
28 ↵
2
2
7
q ↵(to quit)

logname Command

logname↵ Prints the login name of the user.

uname Command

uname ↵ Prints the name of the UNIX system.

id Command

id ↵ Prints the group and user id.

tty Command

tty ↵ Prints the terminal name.

date Command

date ↵ It shows the current date and time.

cmchk Command

cmchk ↵ It shows the block size. Whenever a file is created one block is made available for starting this file's contents.

ulimit Command

ulimit ↵ It implies that the user can't create a file whose size is larger than the size specified in the ulimit command.User can decrease the maximum size of file but can't increase.

ulimit 1 ↵ Means, no file can be created whose size is bigger than 512 bytes.

passwd Command

(5)

passwd ↵ User can change his/her password by this command.

cat /etc/passwd

The content of file '/etc/passwd' is shown by the cat command. It gives important information about user. It contains 6 columns- login name, encrypted password, user id, group id, default working directory, default working shell.

cal Command

Prints calendar.

cal ↵ Prints the calendar of current month of current year.
cal 2008 ↵ Prints the calendar of 2008.
cal 10 2008-↵ Prints the calendar of October 2008.

banner Command

banner HELLO ↵ It prints the word specified in the command in large letters.

file* Command

It reports the type of each file in the current directory, whether the files are empty, directory, ascii text, commands text, c program text etc.

wc Command

It prints the number of lines, words and characters in the specified file(s). It has 3 options –l, -w, -c. –l for line, -w for words, and –c for characters.

wc –lc file1↵ Counts lines and characters from file1.
wc –l file1↵ Counts lines from file1.
wc –wc file1↵ Counts words and characters from file1
wc –wlc file1↵ Counts words, lines and characters from file1.

sort Command

sort myfile ↵ Sorts the contents of the file 'myfile'.
sort file1 file2 file3 ↵ Sorts the two files 'file1' and 'file2' and stores into 'file3'
sort –oresult file1 file2 file3↵ It doesn't display the sorted output on the screen.

sort –u –oresult file1 file2 file3↵ If there are repeated lines in each of these files and we want that such lines should occur only once in the output then use this command.

sort –m file1 file2 ↵ Merges two sorted files 'file1' and 'file2'
sort file1↵ It combines the contents of file1 with the input from the keyboard and then carry out the sorting.

sort –r file1↵ It reverses the sort.
sort –r +1 -2 students↵ It only sorts from the 2nd field of 'students' to the 3rd field.

cut Command

cut –f 2,7 emp↵	It prints only the 2nd and 7th fields of the file 'emp',
cut –f 2-7 emp↵	It prints the fields from 2 through 7 of the file 'emp'.
cut –f 3, 4-6 emp↵	It prints the fields 3rd,4th, 5th and 6th.
cut –f 2,7 –d ":" emp↵	It works if the fields are separated by ":" delimiter.
cut –c 1-10 emp↵	It prints first 10 columns of 'emp'.

grep Command

It searches patterns from a file.

grep picture myfile↵	It searches the word 'picture' in the myfile and if found, the lines containing it would be displayed on the screen.
grep picture myfile story↵	It searches the word 'picture' would be searched in both the files 'myfile' and 'story' and then the lines containing it would be displayed along with the name of that file.
grep [Hh]ello myfile↵	It searches all occurrences of Hello as well as hello and display the lines which contain one of these words.
grep b??k myfile↵ a 'b' and	It searches and displays all four letter words whose first letter is last letter'k'. The two ?? can by any character each.
grep –v picture myfile↵	It prints the lines that doesn't contain the word 'picture'.
grep –n picture myfile↵	It gives the line no. which contains the word 'picture'.
grep –c 'director' emp*.lst	It will print filenames prefixed to the line count.
grep –l 'manager' *.lst	It will display filenames *only*
grep –e 'Agarwal' –e 'aggarwal' –e 'agrawa ' emp.lst	It will print matching multiple patterns

BRE character subset

The basic regular expression character subset uses an elaborate meta character set, overshadowing the shell's wild-cards, and can perform amazing matches.

*	Zero or more occurrences
g*	nothing or g, gg, ggg, etc.
.	A single character
.*	nothing or any number of characters
[pqr]	a single character p, q or r
[c1-c2]	a single character within the ASCII range represented by c1 and c2

^ for matching at the beginning of a line

$ for matching at the end of a line

grep "^2" emp.lst Selects lines where emp_id starting with 2

grep "7…$" emp.lst Se Selects lines where emp_salary ranges between 7000 to 7999
grep "^[^2]" emp.lst Selects lines where emp_id doesn't start with 2

Matching multiple patterns (|, (and))

grep –E 'sengupta|dasgupta' emp.lst

We can locate both without using –e option twice, or

grep –E '(sen|das)gupta' emp.lst

dd Command

It converts the contents of the file from one form into another.

dd if=report of=document conv=ucase↵

It converts the input file report to uppercase and copy them to the output file called document.

Options- conv=ascii convert the file to ascii file
conv=lcase convert alphabets to lower case.

head Command

It shows only the specified no. of lines of a file.

head -15 myfile↵ It shows first 15 lines of the file 'myfile'. By default it prints first 10 lines.

tail Command

tail -15 myfile ↵ It shows last 15 lines of the file 'myfile'. By default it prints last 10 lines.

Creating Background Processes

sort emp > emp.out &↵ It sorts the input file 'emp' and copies that sorted file into 'emp.out'. But the result is not displayed on the screen vecause of background processes(&). It gives a proves id number such as 4577.

kill Command

kill 4577 ↵ To kill the process having process id as 4577.

unique Command

cat emp|sort|unique temp↵ It sorts the 'emp' files and deletes repeated occurrences of a line(if any) and then stores it in the file emp.

paste Command

paste –d "|" temp1 temp2 >> temp3↵

Merges the contents of temp1 and temp2 with a delimiter "|" and stores the result in temp3.

ps Command

ps ↵ Gives the list of the running processes of the user.
ps –a ↵ Gives the list of all the running processes of all users.
ps – t tty ↵ Gives the list of all the running processes from a particular terminal (for ex. tty)..

ps –f ↵	Same as ps with some additional information.
ps –e ↵	Gives the list of all the running processes with time.

tput Command

tput clear ↵	Clear the screen.
tput cup r c ↵	Move the cursor to row r and column c.
tput bold ↵	Bold display.
tput blink ↵	Blinking display.
tput rev ↵	Reverse video(black on white) display.
tput smul ↵	Start mode underline.
tput rmul ↵	End underline mode.
tput bel ↵	Echo the terminal's "bell" character.
tput lines ↵	Echo number of lines on the screen.
tput cols ↵	Echo number of columns on the screen.
tput ed ↵	Clear to end of the display.
tput el ↵	Clear to end of the line.

tr command

head –n 3 emp.lst | tr '[a-z]' '[A-Z]'

translating characters The tr filter manipulates the individual characters in a line. It translates characters using one or two compact expressions

Changing case of text is possible from lower to upper for first three lines of the file

tr –d	Deleting characters (-d)
tr –s ' '	Compressing multiple consecutive characters (-s)
tr –cd '/' < emp.lst	Complementing values of expression (-c)

sed – The Stream Editor

sed is a multipurpose tool which combines the work of several filters. sed uses **instructions** to act on text.

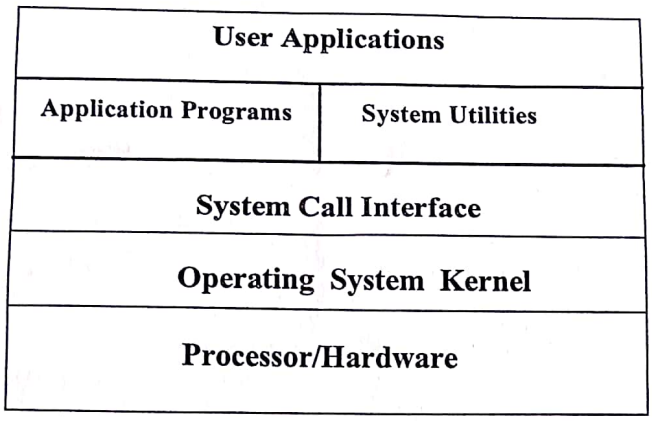
Line Addressing

sed '3q' emp.lst	Just similar to *head –n 3 emp.lst*. Select first three lines and quits
sed–n'1,2p'emp lst p	prints selected lines as well as all lines. To suppress this behavior, we use –n whenever we use p command
sed –n '$p' emp.lst	Selects last line of the file
sed –n '9,11p' emp.lst	Selecting lines from anywhere of the file, between lines from 9 to 11
sed –n '1,2p7,9p $p' emp.lst	Selecting multiple groups of lines
sed –n '3,$!p' emp.lst	Negating the action, just same as 1,2p

The UNIX Operating System

Operating System

An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers, network cards etc.) in a **safe, efficient** and **abstract** way.



UNIX Operating system allows complex tasks to be performed with a few keystrokes. It doesn't tell or warn the user about the consequences of the command.

Features of UNIX OS

Several features of UNIX have made it popular. Some of them are:

Portable

UNIX can be installed on many hardware platforms. Its widespread use can be traced to the decision to develop it *using the C language.*

Multiuser

The UNIX design allows multiple users to concurrently share hardware and software

Multitasking

UNIX allows a user to run more than one program at a time. In facomtmorethan one program can be running in the background while a user is working foreground.

Networking

While UNIX was developed to be an interactive, multiuser, multitasking system, networking is also incorporated into the heart of the operating system.

Organized File System

UNIX has a very organized file and directory system that allows users to organize and maintain files.

Device Independence

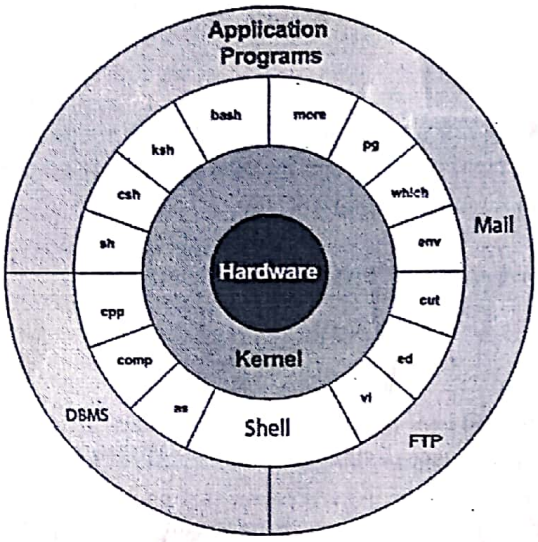
UNIX treats input/output devices like ordinary files. The source or destination for file input and output is easily controlled through a UNIX design feature called redirection.

Utilities

UNIX provides a rich library of utilities that can be use to increase user productivity.

1.2. The UNIX Architecture and Command Usage

The UNIX Architecture



UNIX architecture comprises of two major components viz., the shell and the kernel. The kernel interacts with the machine's hardware and the shell with the user.

Kernel

The kernel is the core of the operating system. It is a collection of routines written in C. It is loaded into memory when the system is booted and communicates directly with the hardware. User programs that need to access the hardware use the services of the kernel via use of system calls and the kernel performs the job on behalf of the user. Kernel is also responsible for managing system's memory, schedules processes, decides their priorities.

Shell

The shell performs the role of command interpreter. Even though there's only one kernel running on the system, there could be several shells in action, one for each user who's logged in. The shell is responsible for interpreting the meaning of metacharacters if any, found on the command line before dispatching the command to the kernel for execution.

Commands and Utilities

There are various command and utilities which you would use in your day to day activities. cp, mv, cat and grep etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various optional options.

Files and Directories:

All data in UNIX is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system.

The File and Process

A file is an array of bytes that stores information. It is also related to another file in the sense that both belong to a single hierarchical directory structure.
A process is the second abstraction UNIX provides. It can be treated as a time image of an executable file. Like files, processes also belong to a hierarchical structure.

The File System

Types of files

On a UNIX system, everything is a file; if something is not a file, it is a process.

A UNIX system makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system. Most files are just files, called *regular* files; they contain normal data, for example text files, executable files or programs, input for or output from a program and so on.

Directories: files that are lists of other files.

Special files or Device Files: All devices and peripherals are represented by files. To read or write a device, you have to perform these operations on its associated file. Most special files are in /dev.

Links: a system to make a file or directory visible in multiple parts of the system's file tree.

(Domain) sockets: a special file type, similar to TCP/IP sockets, providing inter-process networking protected by the file system's access control.

Named pipes: act more or less **like** sockets and form a way for processes to communicate with each other, without using network socket semantics.

Ordinary (Regular) File

This is the most common file type. An ordinary file can be either a text file or a binary file.
A text file contains only printable characters and you can view and edit them.
A binary file, on the other hand, contains both printable and nonprintable characters that cover the entire ASCII range. The object code and executables that you produce by compiling C programs are binary files. Sound and video files are also binary files.
This means that when you execute ls command, the shell locates this file in /bin directory and makes arrangements to execute it.

Directory File

A directory contains no data, but keeps details of the files and subdirectories that it contains. A directory file contains one entry for every file and subdirectory that it houses. Each entry has two components namely, the filename and a unique identification number of the file or directory (called the *inode number).* When you create or remove a file, the kernel automatically updates its corresponding directory by adding or removing the entry (filename and inode number) associated with the file.

Device File

All the operations on the devices are performed by reading or writing the file representing the device. It is advantageous to treat devices as files as some of the commands used to access an ordinary file can be used with device files as well.
Device filenames are found in a single directory structure, /dev. A device file is not really a stream of characters. It is the attributes of the file that entirely govern the operation of the device. The kernel identifies a device from its attributes and uses them to operate the device.

Filenames in UNIX

On a UNIX system, a filename can consist of up to 255 characters. Files may or may not have extensions and can consist of practically any ASCII character except the / and the Null character . It is recommended that only the following characters be used in filenames:

Alphabets and numerals.

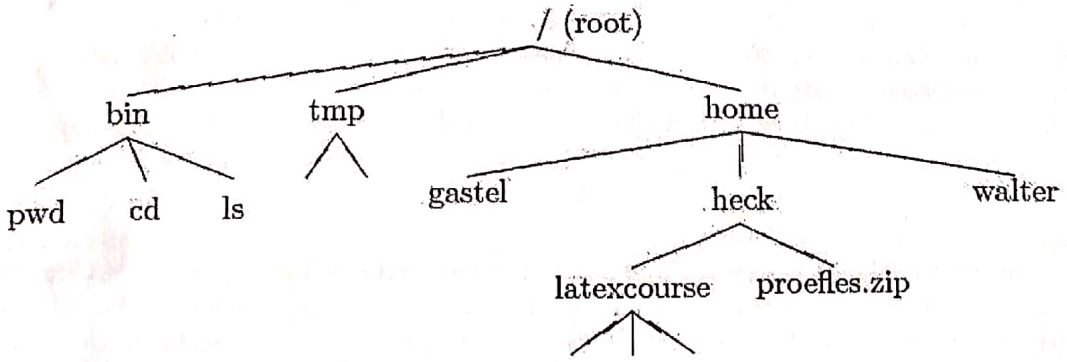
The period (.), hyphen (-) and underscore (_).

UNIX imposes no restrictions on the extension. In all cases, it is the application that imposes that restriction. Eg. A C Compiler expects C program filenames to end with .c, Oracle requires SQL scripts to have .sql extension.
A file can have as many dots embedded in its name. A filename can also begin with or end with a dot.
UNIX is case sensitive; cap01, Chap01 and CHAP01 are three different filenames that can coexist in the same directory.

Directories and Files

A file is a set of data that has a name. The information can be an ordinary text, a user-written computer program, results of a computation, a picture, and so on. The file name may consist of ordinary characters, digits and special tokens like the underscore, except the forward slash (/). It is permitted to use special tokens like the ampersand (&) or spaces in a filename.

Unix organizes files in a tree-like hierarchical structure, with the *root directory*, indicated by a forward slash (/), at the top of the tree.



Absolute and relative paths

A path, which is the way you need to follow in the tree structure to reach a given file, can be described as starting from the trunk of the tree (the / or root directory). In that case, the path starts with a slash and is called an absolute path, since there can be no mistake: only one file on the system can comply.

Paths that don't start with a slash are always relative to the current directory. In relative paths we also use the . and .. indications for the current and the parent directory.

The root directory has many subdirectories. The following table describes some of the subdirectories contained under root.

Directory	Content
/bin	Common programs, shared by the system, the system administrator and the users.
/dev	Contains references to all the CPU peripheral hardware, which are represented as files with special properties.
/etc	Most important system config ration files re in /etc, this directory contains data similar to those in the Control P nel in Window
/home	Home directories of the common er .
/lib	Library files, includes files for all kinds of programs needed by the system and the users.
/sbin	Programs for use by the system and the system administrator.
/tmp	Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work!
/usr	Programs, libraries, documentation etc. for all user-related programs.
/var	Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet, or to keep an image of a CD before burning it.

vi Basics

To add some text to a file, we invoke,

vi *<filename>*

In all probability, the file doesn't exist, and vi presents you a full screen with the filename shown at the bottom with the qualifier. The cursor is positioned at the top and all remaining lines of the screen show a ~.

For, text editing, vi uses 24 out of 25 lines that are normally available in the terminal. To enter text, you must switch to the input mode. First press the key i, and you are in this mode ready to input text. Subsequent key depressions will then show up on the screen as text input.

After text entry is complete, the cursor is positioned on the last character of the last line. This is known as current line and the character where the cursor is stationed is the current cursor position. This mode is used to handle files and perform substitution. After the command is run, you are back to the default command mode. If a word has been misspelled, use ctrl-w to erase the entire word.

Now press esc key to revert to command mode. To save the entered text, you must switch to the execute mode (the last line mode). Invoke the execute mode from the command mode by entering a : which shows up in the last line.
It is possible to display the mode in which is user is in by typing,

:set showmode

Messages like INSERT MODE, REPLACE MODE, CHANGE MODE, etc will appear in the last line.

- **INSERT Mode** - For adding text to a file
  The three most common ways to enter the Insert Mode are:

Letter	Action
I	Starts inserting in front of the current cursor position
I	Starts adding at the front of the line
A	Starts adding after the cursor
A	Starts adding at the end of the line
O	Starts opening a new line underneath the cursor
O	Starts opening a line above the cursor.
<Esc>	Gets out of Insert Mode

- **EDIT Mode**

  - Generally for moving the cursor and deleting stuff.
  In the Edit Mode, the keys do not type letters, but do other actions such as cursor movement, deletions, copying lines, etc.

Letter	*Simple Cursor Movement*	
h	Moves cursor left one space	Note: the Arrow keys do work locally, but sometimes mess up over a network.
j	Moves cursor down one line	
k	Moves cursor up one line	

15

l	Moves cursor right one space
Fast Cursor Movement	
w	Moves the cursor a full word at a time to the right
b	Moves the cursor back to the left a word at a time
^	Moves the cursor to the front of a line
$	Moves the cursor to the end of a line
<ctrl>f	Moves the cursor forward a full page of text at a time
<ctrl>b	Moves the cursor backward a full page of text at a time
Modifying Text	
x	Deletes the character under the cursor
dd	Deletes the line where the cursor is located (type d twice!)
n dd	Delete ***n*** consecutive lines (***n*** is an integer)
r	Replaces the character under the cursor with the next thing typed
J	Joins current line with the one below (Capital J!)
u	Undoes the last edit operation
<ctrl> r	Redo (Undoes the last undo operation)
Cut and Paste Operations	
yy	Copies or yanks a line (5yy yanks 5 lines)
p	Puts the yanked text on the line below the cursor (lower case p)
P	Puts the yanked text above the current line (capital P)
Note: If vi is already in the input mode, text from that or another window may be highlited using the left mouse button, and copied into place by pressing the middle mouse button.	

- **COMMAND Mode** - For interacting with the operating system.
  To enter the Command Mode, a colon " **:** " must precede the actual command.

Letter	Action
: r <file>	reads a file from disk into the vi editor
: w <file>	writes current file to disk
: wq	writes the file and quits vi
: q!	quits without writing (useful if you've messed up!)
:w *<filename>*	save as
:w! *<filename>*	save as, but overwrites existing file
:q	quits editing mode by rejecting changes made
:recover	recovers file from a crash

Navigation

A command mode command doesn't show up on screen but simply performs a function. To move the cursor in four directions,

k moves cursor up

j moves cursor down

h moves cursor left

L moves cursor right

Word Navigation

Moving by one character is not lways enough. You will often need to move faster along a line. vi understands a word as a navigation unit which can be defined in two ways, depending on the key pressed. If your cursor is a number of words away from your desired position, you can use the word-navigation commands to go there directly. There are three basic commands:

b moves back to beginning of word
e moves forward to end of word

w moves for ard to beginning word

Example,

5b takes the cursor 5 words back

3w takes the cursor 3 words forward

Moving to Line Extremes

Moving to the beginning or end of a line is a common requirement.

To move to the first character of a line

0 or |

30| moves cursor to column 30

$ moves to the end of the current line

The use of these commands along with b, e, and w is allowed

Scrolling

Faster movement can be achieved by scrolling text in the window using the control keys. The two commands for scrolling a page at a time are

ctrl-f scrolls forward

ctrl-b scrolls backward

10ctrl-f scroll 10 pages and navigate faster

ctrl-d scrolls half page forward

ctrl-u scrolls half page backward

Absolute Movement

The editor displays the total number of ine in the last line

Ctrl-g to know the current line number

40G goes to line number 40

1G goes to line number 1

G goes to end of file

Editing Text

The editing facilitates in vi are very elaborate and invoke the use of operators. They use operators, such as,

d delete

y yank (copy)

Deleting Text

x deletes a single character

dd delete entire line

yy copy entire line

6dd deletes the current line and five lines below

Moving Text

Moving text (p) puts the text at the new location.

p and P place text on right and left only when you delete parts of lines. But the same keys get associated with "below" and "above" when you delete complete lines

Copying Text

Copying text (y and p) is achieved as,

yy copies current line

10yy copies current line & 9 lines below

Joining Lines

J to join the current line and the line following it

4J joins following 3 lines with current line

Undoing Last Editing Instruct serions

In command mode, to undo the last change made, we use u

To discard all changes made to the current line, we use U

vim (LINUX) lets you undo and redo multiple editing instructions. u behaves differently here; repeated use of this key progressively undoes your previous actions. You could even have the original file in front of you. Further 10u reverses your last 10 editing actions. The function of U remains the same.

19

You may overshoot the desired mark when you keep u pressed, in which case use ctrl-r to redo your undone actions. Further, undoing with 10u can be completely reversed with 10ctrl-r. The undoing limit is set by the execute mode command: set undolevels=n, where n is set to 1000 by default.

Repeating the Last Command

The . (dot) command is used for repeating the last instruction in both editing and command mode commands

For example:

2dd deletes 2 lines from current line and to repeat this operation, type. **(dot)**

Search and repeat commands

Command	**Function**
/pat	searches forward for pattern pat
?pat	searches backward for pattern pat
N	repeats search in same direction along which previous search was made
N	repeats search in direction opposite to that along which previous search was made

Substitution – search and replace

We can perform search and replace in execute mode using :s. Its syntax is,

:address/source_pattern/target_pattern/flags

:1,$s/director/member/g can also use % instead of 1,$

:1,50s/unsigned//g	deletes unsigned everywhere in lines 1 to 50
:3,10s/director/member/g	substitute lines 3 through 10