# BASICS OF OPERATING SYSTEM (4330701)

# CHAPTER-5 LINUX BASICS

COMPUTER ENGINEERING DEPARTMENT

AVPTI-RAJKOT

## Content

# Linux Overview

- More than 90% of todays 500 fastest supercomputer uses **Linux**. Most widely used search engine **Google uses Linux** as its Operating system.
- **Linux was first developed by Linus Torvalds, a student in Finland, in 1991.**
- Now Linux is not owned by anyone.
- **Linux** is the most famous *free and open source* Operating System.
- Open source OS are available in source code format. Free software means not by money point of view, but software is that is free like Linux is distributed along with its source code.
- So anyone who have this software **can make changes to it and redistribute it**.
- **Flavors of Linux.- Red Hat Enterprise Linux, Fedora, CentOS, Debian, Ubuntu, Linux Mint, Linspire, PCLinux.**
- It has been designed to run efficiently and reliably on common PC hardware, but also runs on a variety of other platforms.
- The core Linux operating system kernel is entirely original, but it can run much existing free UNIX software, resulting in an entire UNIX-compatible operating system free from proprietary code.

# Features of Linux

1. Free and Open Source Software
   - Available with source code, freedom to make changes in source code.
2. Flexibility in Usage
   - Can be used for server applications, desktop applications and embedded systems.
   - Available in phones, tablets, computers, network routers, pcs etc…
3. A Multi User System
   - Allows multiple users to work simultaneously. Example TELNET
4. A Multi Tasking System
   - Allows multiple programs to run simultaneously. Foreground processes and background processes.
5. High Performance and Reliability
   - High performance with minimum required hardwares.

# Features of Linux

6. The Building-Block Approach
   - Linux uses building block approach to perform complex tasks. Provides few hundred commands each of which can perform simple task. To perform complex tasks simple commands can be combined.

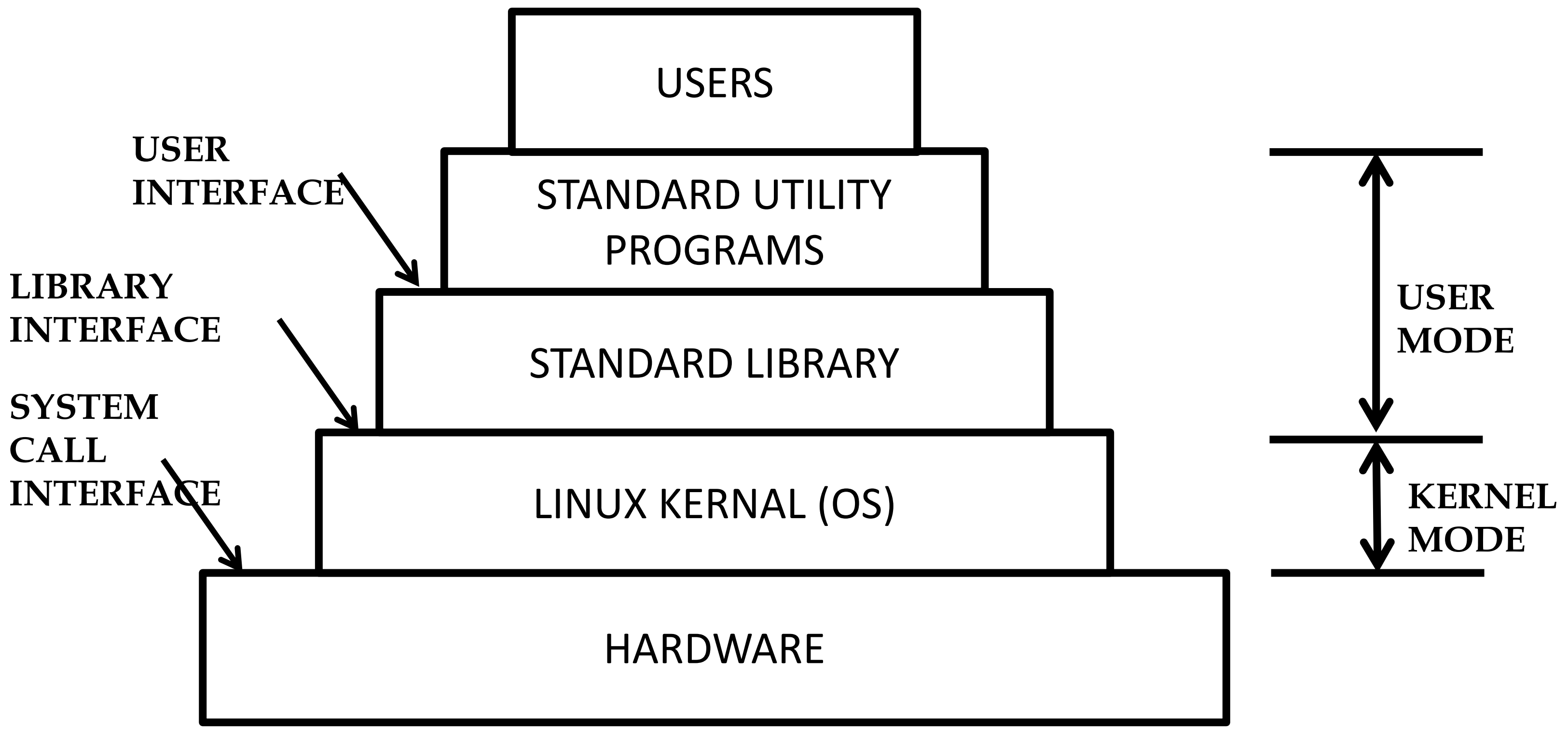
7. Flexibility Interface
   - Supports both types of interface GUI and CLI

8. File System Support
   - Wide range of file systems like ext, ext2 etc…

9. Programming Facility
   - Linux shell is also a programming language, which supports all programming features suck as variable, control structures, loops etc…

10. On Line Help
    6. For example 'man' command for getting help of any linux command.

# Linux Architecture



## Linux Architecture

- **Hardware:**
  - Bottom layer
  - Consists of various devices such as CPU, memory, disks, ,monitors, printers.
  - These devices provide various services. For example printer provides printing services.
- **Kernel:**
  - It is an actual operating system.
  - Next higher level, It manages underlying hardwares
  - Directly interacts with hardwares and provides the services to users.
  - Provides simple interface between user programs and hardware.
  - Provides services like process mgmt, file mgmg, i/o mgmgt

## Linux Architecture

- **Standard libraries:**
  - It contains set of procedures (functions), one procedure per system call. Procedures are used to invoke the system calls
- **Standard utility programs:**
  - Are command interpreter (shell), compilers, editors, text processing programs, file manipulation utilities, GUI etc…
  - this programs help user tasks more simpler.
- **Users:**
  - The top most layer is of users.
  - They directly interacts with system.

# Linux Terminology

- **Kernel**
  - ✓ The **kernel** is core of Linux OS. Kernel is program, which is loaded in memory when system is turned on.
  - ✓ Device drivers can be loaded and unloaded into kernel in form of kernel modules. Kernel interact with hardware directly.
  - ✓ When user wants to use any hardware, it has to use service provided by kernel. Special functions called **system calls** are used to request kernel.
- **Shell**
  - ✓ The shell is an **interface** between user and the kernel.
  - ✓ When user logs-in to the system, process for shell starts execution. It terminates when user logs-out from system
  - ✓ User can directly interact with shell. It works as command interpreter. Command will be translated them into kernel readable form.
- **System Call**
  - ✓ System calls are special functions.
  - ✓ They are used to request kernel service.
  - ✓ They can be invoked via library procedures.

## Kernel

- It is the core of the UNIX operating system
- It is a program which is loaded in memory when system is turned on.
- It stays there and provides various services until the system is turned off.
- Kernel interacts with the hardware directly.
- When user program needs to use any hardware, it has to use services provided by the kernel.
- In short, kernel manages entire computer system.

# Shell

A Linux shell is a command interface, programming language, process. These various roles of shell are described below:

1. **Shell as an interface:** Shell is an interface between user and kernel.
   1. A user gives commands on the command prompt of shell, in order to communicate with the shell.
   2. Thereafter, shell interacts with kernel and executes these commands.

**2 Shell as Command interpreter**

- As shell is the interface between user and kernel, all the commands written on command prompt of shell are interpreted by the shell.
- If any special character is present in the command, they are simplified for the kernel.
- Also unnecessary white spaces are removed from the command and then presented to the kernel for execution.

3) Shell as Programming language

- Shell also provides all the features of programming language.
- Such as loops, branching, variable declairation,etc.
- The programs developed using shell are called shell script, it also includes Linux commands.

4) Shell as a process

When the user logged-in , the shell starts its execution in form of shell process.

This process is under execution till the user gets logged-out.

During execution of this shell process, it creates an environment for the users to work.

This process executes in user mode, this for various tasks, it requires kernel services, which are requested by generating system call.

## System call

- System calls are special functions.
- They are used to request kernel to provide various services,such as reading from a file stored on a disk.
- They can be invoked via library procedures, or via commands provided by shell, or even directly from C program in UNIX.
- System calls are similar to user defined functions.
- Difference is that they execute in the kernel mode,having fully access to all the hardware; while user defined functions execute in user mode,having no direct access to the hardware.

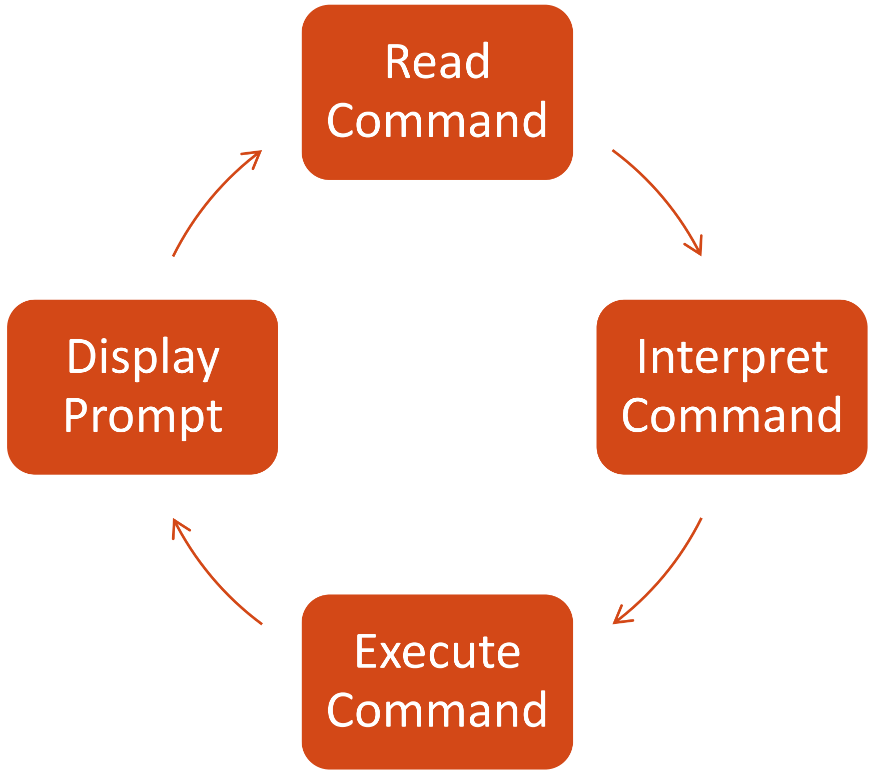
## Shells available in Linux:

a) Bourne shell (sh) : The sh shell was the earliest shell, being developed for UNIX back in the late 1970s.

b) C shell (csh) : The csh shell was originally developed for BSD UNIX. It uses a syntax that is very similar to C programming.

c) Korn shell (ksh) : It is superset of Bourne shell

# Interpretive cycle of a shell



1. The shell displays prompt and waits for user to enter command.
2. As soon as the command is entered, it is read by the shell and scans it for any meta characters, if found the command is recreated in simplified form.
3. The command is then passed to kernel for execution. During execution, shell does not do any work.
4. Prompt appears after execution of command completes.

## Pattern Matching (Wild Card)

Shell provides a set of special characters to match patterns.

| Wild-card | Matching Patterns |
|---|---|
| ? | A single character |
| * | Any number of characters including NULL |
| [abc] | Single characters, either a,b or c |
| [a-z] | Single character within the range a to z |
| [a-dp-s] | Single character within the range ato d, p to s |
| [!abc] | Single character, not a,b,c (other than a,b,c) |
| [!a-c] | Single character not within the range of a to c |
| . | Strings starting with '.' must be matched explicitly |

## Standard files

When a program is executed from shell, it opens three file, namely:

1. The standard input file: It provides a way to read input data for a process from standard input.

   By default standard input is keyboard.

2. The standard output file: It provides a way to write output data for a process.

   By default the standard output goes to monitor.

3. The standard error file: It provides a way to report errors encountered during process execution.

   By default the standard error goes to the monitor.

# Redirection

The default standard input, output and error can be changed using Redirection

1. Redirecting Input: The process is told to read the input from file stored on disk, instead of keyboard.

   Syntax: command < filename

   e.g. wc < student.txt

2. Redirecting Output: Process is told to write output to some file on disk, instead of monitor.

   Syntax: command > filename

   e.g. wc student.txt > test.txt

3. Redirecting error: Process is told to write error messages to file on disk, instead of monitor.

   Syntax: command 2> filename

   e.g. wc student.txt 2> err.txt

## Filters

Various Linux commands make use of standard inputs and outputs differently.

Some commands use either standard input or output, whereas some commands use both.

Filters use both standard input and output.

e.g. cat, head, tail, wc, grep, etc

## Pipes

It is used to connect two processes.

i.e. standard output of one process is fed to another process.

Syntax: Command1 | Command2

e.g. who | wc –l

who list the users, this is fed to wc –l, this gives count of number of users currently logged-in to the system.

## Metacharacters

These are special characters reserved for pre-defined purposes. These characters are not understood by kernel.

Thus whenever a command is entered, the shell reads the command to find metacharacters in it.

If any metacharacter is found in the command, the shell recreates the command in simplified form to present it to the kernel.

| Symbol | Meaning |
|---|---|
| > | Output redirection |
| >> | Output redirection (append) |
| < | Input redirection |
| * | File substitution wildcard; zero or more characters |
| ? | File substitution wildcard; one character |
| [ ] | File substitution wildcard; any character between brackets |
| `cmd` | Command Substitution |
| $(cmd) | Command Substitution |
| \| | The Pipe (\|) |
| ; | Command sequence, Sequences of Commands |
| [ ] | File substitution wildcard; any character between brackets |
| \|\| | OR conditional execution |
| && | AND conditional execution |
| ( ) | Group commands, Sequences of Commands |
| & | Run command in the background, Background Processes |
| # | Comment |
| $ | Expand the value of a variable |
| \ | Prevent or escape interpretation of the next character |
| << | Input redirection |

# Cont…

If metacharacter is to be used as normal character, escaping or quoting of the character is to be done.

a. Escaping: Provide backslash (\) before the character.

   e.g. cat>> stud*

b. Quoting: Enclose the metacharacter/ entire expression within quotes.

   e.g. cat>> “stud*”

## Shell Variables

Variables are used to store values.

They are divided into two types:

1. User defined variables
2. Linux defined variables

## User Defined Variables

These variables are defined by user, either using command prompt or using shell script.

Rules for variable names: same as C language

**Syntax of variable assignment:**

variable = value

e.g. abc = 10

**Variable evaluation:**

Syntax: $variable

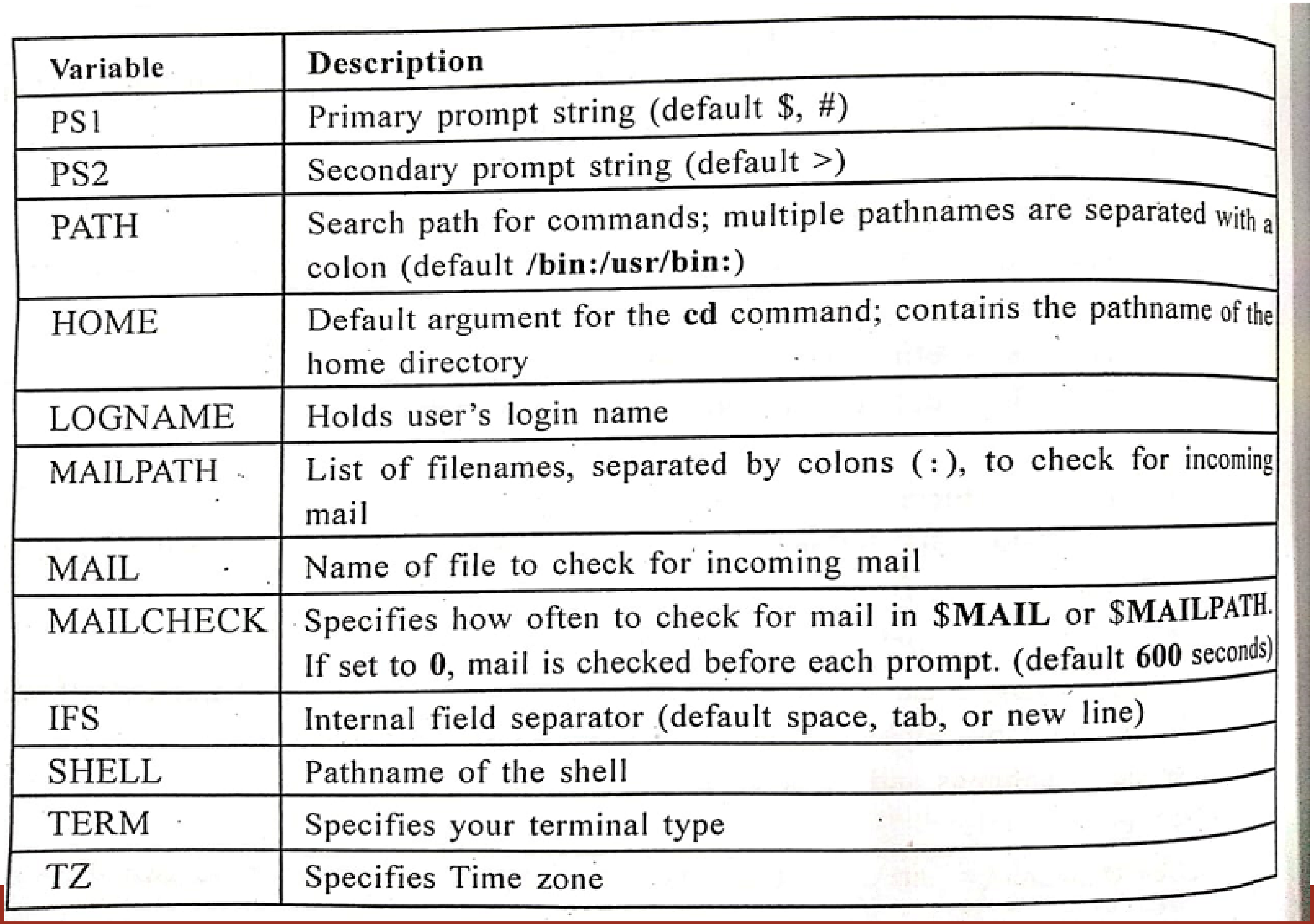
e.g. echo $abc    or    xyz=$abc

## Linux defined variables/ System variables

These variables are defined by Linux to customize the working environment.

They contain default values which can be read by user. If user desires, the default values can be changed.

## System Variables



| Variable | Description |
|---|---|
| PS1 | Primary prompt string (default $, #) |
| PS2 | Secondary prompt string (default >) |
| PATH | Search path for commands; multiple pathnames are separated with a colon (default **/bin:/usr/bin:**) |
| HOME | Default argument for the **cd** command; contains the pathname of the home directory |
| LOGNAME | Holds user's login name |
| MAILPATH | List of filenames, separated by colons (:), to check for incoming mail |
| MAIL | Name of file to check for incoming mail |
| MAILCHECK | Specifies how often to check for mail in $**MAIL** or $**MAILPATH**. If set to **0**, mail is checked before each prompt. (default **600** seconds) |
| IFS | Internal field separator (default space, tab, or new line) |
| SHELL | Pathname of the shell |
| TERM | Specifies your terminal type |
| TZ | Specifies Time zone |

## Vi & Vim

Vi is one of the most powerful and most widely used command-prompt based text editor. Vim is the improved vi. Vim stands for ‘ Vi improved ’.

They can be launched using following command syntax.

Syntax:- vi filename or vim filename

Example:- vi hworld.c or vim hworld.c

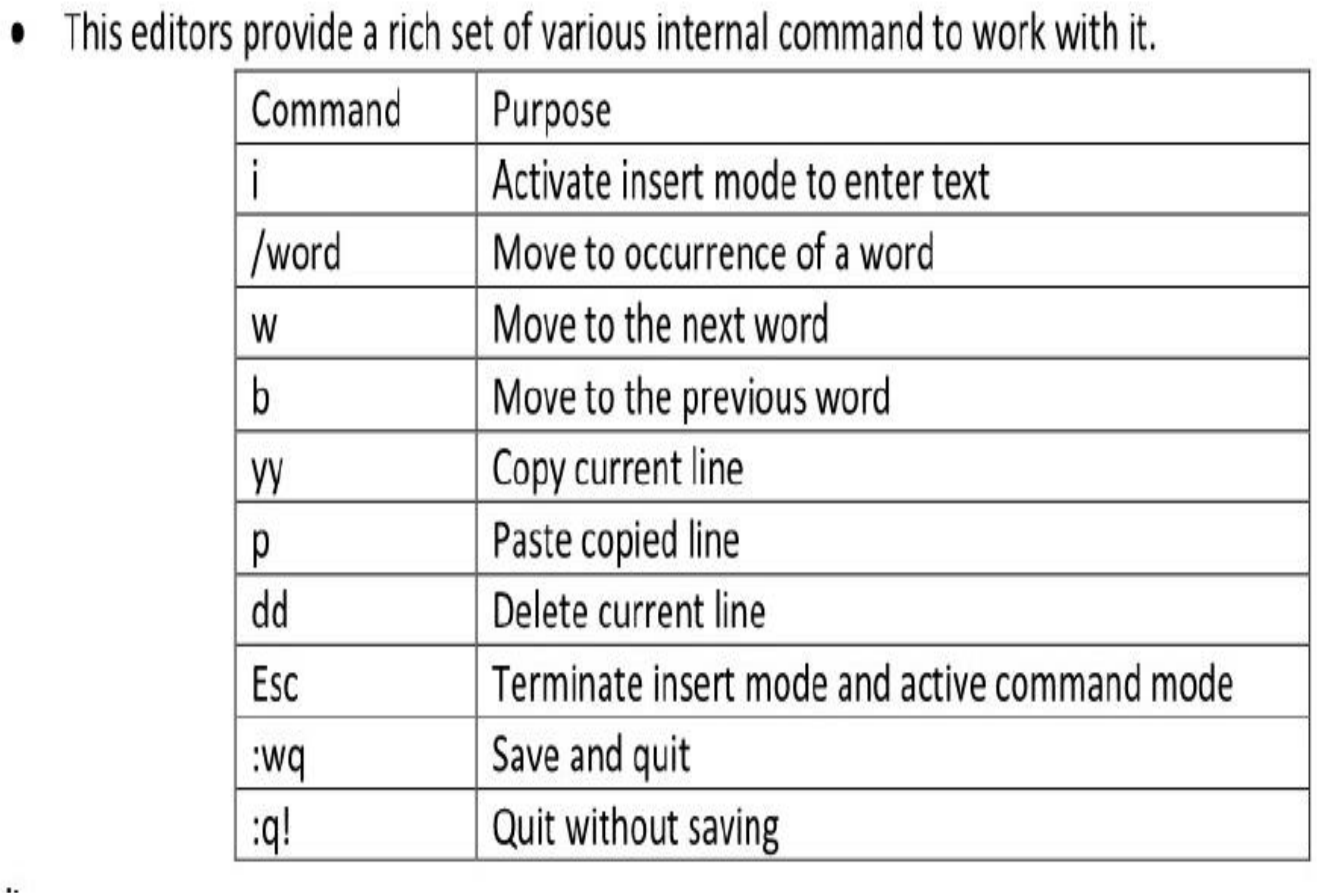
Both of this editors work in three modes as described below :

1)Command mode :- This isthe default mode. Various commands, described below, can be executed in this mode to perform various tasks.

2)Insert mode :- Text can be inserted in this mode. This mode can be activated by pressing ‘ i ’ in command mode. ‘ Esc ’ key ends insert mode and returns to the command mode.

3)Command line mode :-This mode can be activated by pressing ‘ : ’ in command mode. This puts the command line entry at the bottom of the screen. Various command, as described below , can be executed in this mode to perform various tasks.

## Vi & Vim



- This editors provide a rich set of various internal command to work with it.

| Command | Purpose |
|---|---|
| i | Activate insert mode to enter text |
| /word | Move to occurrence of a word |
| w | Move to the next word |
| b | Move to the previous word |
| yy | Copy current line |
| p | Paste copied line |
| dd | Delete current line |
| Esc | Terminate insert mode and active command mode |
| :wq | Save and quit |
| :q! | Quit without saving |

# gedit

- It is GUI based text editor and can be used with Linux as well as other Operating Systems such as windows, Mac OS X.
- It is similar to ‘ notepad ’ editor of  Windows Operating Systems. But , it is far more powerful than ‘ notepad ’.
- It is free software and provides simplicity and ease of use.
- It includes syntax highlighting for various program code and text markup formats.
- It also has GUI tabs for editing multiple files. Users can work with multiple files simultaneously using these tabs.
- It supports a full undo and redo system as well as search and replace. Other typical code oriented features include line numbering , bracket matching , text wrapping , current line highlighting , automatic indentation and automatic file backup.
- It also support multi-language spell checking and a flexible plug-in system allowing the addition of new features.
- It automatically detects  when an open file is modified on disk by another application and offers to reload that file.
- It also supports printing , including print preview and printing to PostScript and PDF files. Printing option include text font , and page size , orientation , margins , optional printing of page headers and line numbers , as well as syntax highlighting.

# gcc

- gcc refers to GNU compiler collection.
- Linus Torvalds used gcc with developing Linux Kernel to compiles his programs.
- gcc  is a compiler for c , c++ , JAVA , Fortran and other programming languaeges that can be used in Linux.
- Here steps are given to compile a c program.

1) Devlop a simple C program to display “hello world”

   using  any of the above text editors and store it in a file named  *hword.c* ( Hope , you are familiar with C language and provide code for this program.)

1) Compile the program by issuing following code.

- **gcc*test.c*-0test.*out***
- Here ,test.out represents executable file similar to  .exe file of Windows.

1) Run the program by issuing following syntax.

- ***.l test .out***

# Thank you…