UNIX PROGRAMMING

UNIT-I

- **Introduction to Unix**
- **Brief History**
- **What is Unix**
- **Unix Components**
- **Using Unix**
- **Commands in Unix**
- **Some Basic Commands**
- **Command Substitution**
- **Giving Multiple Commands.**
- **Important and frequently asked questions**

1. Introduction to Unix :

1.1.What is an Operating system ?

An operating system is an organized collection of programs that specifically designed to manage the resources of the computer system and facilitate the creation of computer programs and control of their execution in that system. It is an user interface between the user and machine.

1.2.Why UNIX?

UNIX has developed into a very popular operating system for a variety of reasons:

- UNIX is **portable**, written in C and not tied to any specified computer hardware. It can ported to virtually any type of computer, PC's , Macintoshes, workstations etc.
- UNIX is a **Multi-user** operating system. More than one user can use UNIX at the same time. Hardware resources like printers and large file servers can be used by many users.
- UNIX is **Multitasking** operating system. User can perform two tasks simultaneously .Ex: User can formatting a text file in the background while reading through email.
- UNIX has built-in **networking** .one of the biggest challenges in today's computing world is to link different types of computers across small and large areas. With UNIX , the networking is built-in with various programs and utilities.

1.3.Overview of Unix :

It is an Open Operating system. Unix is found on virtually all computer hardware in use today. Unix is simple and easy to use. The Unix operating system was developed by **Ken Thompson** at **AT &T Bell laboratories** in **1969.** It is written in **C** – language.

It is a **Time sharing** operating system , that controls the resources (CPU,Memory , I/O & Files) of a computer and allocates them.

It provides a **standard Interface** (The Shell) to run user programs.

It provides a **File system** ,that manages the long term storage of information .such as Programs, data & documents etc.

It supports **Multiple processes & Multiple users**. A process can easily create and terminate other processes in unix environment.

It supports **Networking**, **Graphics**, **Real time operations ,Security** etc.

1.4.Major features of Unix :

- Multiuser capability
- Multitasking capability
- Time sharing
- System portability
- Security
- Device independence
- Networking
- Communications
- Hierarchical File System
- Pipes and Filters

1.4.1.Multiuser capability:

- Unix is a multi-user, multi-tasking, Time sharing operating system.
- Many users can logged into a system simultaneously, each one running many programs by sharing of CPU time , Memory space and IO devices.
- It's the kernel's job to keep each process and user separate and to regulate access to system hardware, including CPU, Memory, Disk and other I/O devices.

1.4.2.Multitasking capability:

- we can do more than one task like reading email &compiling programs at the same time.

1.4.3.Time Sharing:

- The CPU & Memory can be shared among various users to accomplish their jobs.

1.4.4.Portability:

- Applications in one platform can be ported to another platform easily without any modifications.

1.4.5.Security:

UNIX providing 3-levels of security to protect both data & system.

- System Level: By assign USERNAME & PASSWORD
- File Level: By assign file privileges such as Read, Write & Execute.
- Device Level: By use of Cryptography techniques.

1.4.6.Device independence:

- UNIX can work on a variety HW platforms. It treats everthing including memory and IO devices as FILES. There are a large number of files under UNIX environment .
- UNIX has a well organized file and directory system that allows users to organize and maintain these files/directories easily and efficiently.
- As UNIX views and and treats everything as a file it is device independent.

1.5.Advantages of UNIX:

- Provides Multitasking & Multi-User facility to improve the system performance
- High Security through Account validation & Authentication
- Efficient communication done through Network facility.

1.6.Disadvantages of UNIX:

- UNIX is not much user friendly .
- Its basic interface is Command line and even experienced user can may mistakes using this interface.
- More difficult to manage the entire system when compared to other environments.

2. Brief History of UNIX:

- It was born in the late 1960s.
- It was developed , which is based on MULTICS & CTSS operating systems by KEN THOMPSON at AT & T Bell labs in 1969.

What is a MULTICS?

- Stands for Multiplexed Information & Computing Service systems.
- It is not a simple system, joint effort by General Electric, MIT (Massachusetts Institute for Technology) & Bell Labs.

What is CTSS?

- Stands for Compatible Time Sharing Systems@ MIT computation center.
- **Key origin**: Ken Thompson wanted to create a Multiuser operating system to run Space Wars game (first **video game**)
- This game was implemented on DEC PDP-1(Digital Equipment Corporation's Programmed Data Processor version 1).
- Ken's Philosophy was to create an operating system with 'commands & utilities' and wanted to do things well. That system was called "UNIX".
- In 1969, he wrote the first version of the Unix called as "UNIC" . It stands for "UNi-plexed Information & Computing System" and eventually shortened to UNIX.
- The first version of unix were written in "Machine dependent " program on a PDP-7 along with a few utilities. But It was not portable system . And given the name UNIX by Brian Kernighan.
- 00:00:00 Hours, Jan 1, 1970 is time zero for UNIX. It is also called as **epoch**.
- Then, Ken Thompson approached "Dennis Ritchie" wanted to make the UNIX as portable system Both were worked together and written in C-language to made the UNIX as Portable system. So The UNIX is Multiuser Operating System with Portability.
- During the development of Unix, there are **TWO** major contributions.

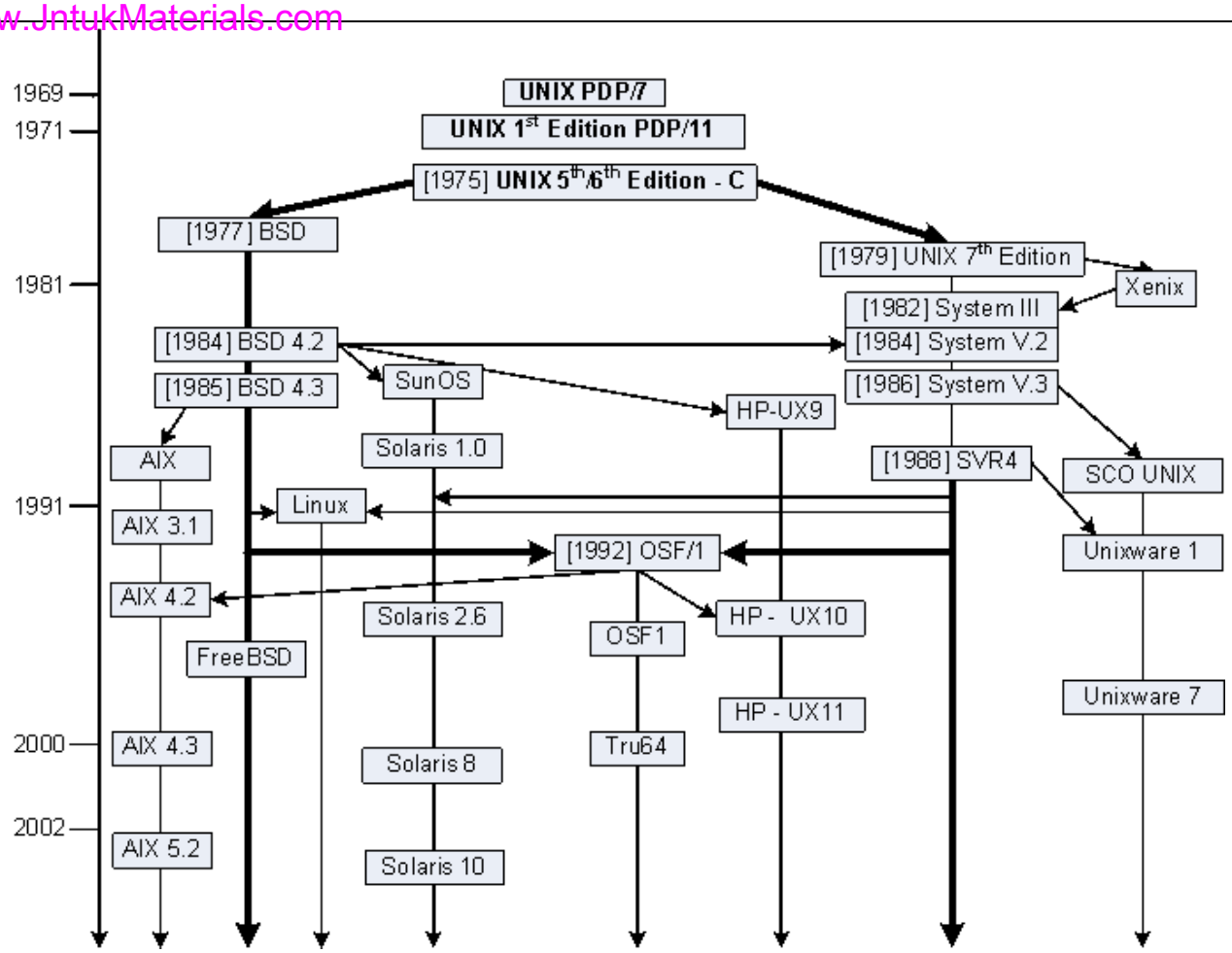
1) Bell laboratory's contribution (MIT,General (Electric , Bell labs)

2)UCB's contribution(University of California at Berkeley) .

Summary :

- First Version was created in Bell Labs in 1969.
- 1973 Unix is re-written mostly in C, a new language developed by Dennis Ritchie.
- 1977 There were about 500 Unix sites world-wide.
- 1980 BSD 4.1 (Berkeley Software Development)
- 1983 SunOS, BSD 4.2, System V
- 1988 AT&T and Sun Microsystems jointly develop System V Release 4 (SVR4). This later developed into UnixWare and Solaris 2.
- 1991 Linux was originated.

Year	Release	Features
1969	—	Ken Thompson wrote first version of UNIX
1975	UNIX V6	First version becomes widely available to industry and academia
1984	UNIX System V Release 2	Shared memory, more commands
1997	Single UNIX Specification 2	Support for real-time processing, threads, and 64-bit processors
2001	Mac OS X	From Apple, a UNIX-based core with Macintosh graphics and GUI
2005	FreeBSD 6.0	Multithreaded file system; expanded support for wireless technology
2009	Mac OS X 10.6 Server	A 64-bit kernel that increased the total number of simultaneous system processes, threads, and network connections for the server to use
2012	Mac OS X 10.8	New support for cloud sharing (see *www.apple.com/osx*).
2012	FreeBSD 9.1	Better load balancing for multithreading systems and more (see *www.freebsd.org*).



Following are the example of The proprietary flavors of unix that have been designed to run only (or mainly) on proprietary hardware sold by the same company

- AIX - developed by IBM for use on its mainframe computers
- BSD/OS - a commercial version of BSD developed by Wind River for Intel processors
- HP-UX - developed by Hewlett-Packard for its HP 9000 series of business servers
- IRIX - developed by SGI for applications that use 3-D visualization and virtual reality
- QNX - a real time operating system developed by QNX Software Systems primarily for use in embedded systems
- Solaris - developed by Sun Microsystems for the SPARC platform and the most widely used proprietary flavor for web servers
- Tru64 - developed by Compaq for the Alpha processor

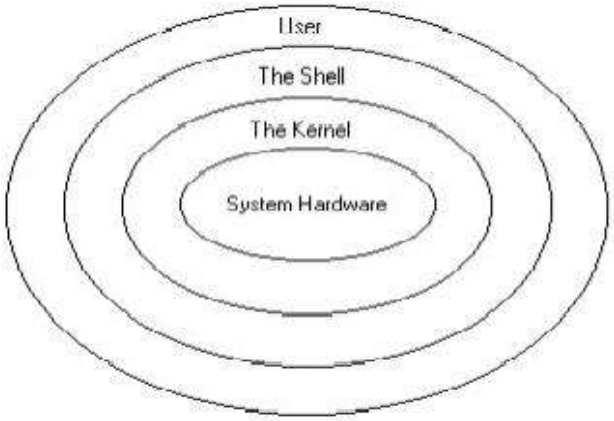
3. What is Unix?

- Unix is an **operating system**. All computers have operating systems. An operating system is a software that acts as an interface between the user and the computer hardware.
- An operating acts as a **resources manager**. It is a **multiuser** operating system, Unix gives its users, the feeling of working on an **independent** computer system.
- In other words, Unix provides **virtual** computers to different users by creating simulated processors, multiple address spaces and so on.
- Unix also provides **communication** facility with other users who are connected to the system either directly or independently via network.
- It is highly **portable** and has a large number of utilities and can work both on Desktops as well as on Networked systems.
- **From Users's perspective**, Unix is a **tool** to run user application programs. It provides various applications such as word processors, electronic spreadsheets, database management systems etc. All application programs access the computer's hardware via operating system like Unix.

4.Unix Components:

Unix has 3 –major components:

1. The Kerenl
2. The Shell
3. The File system



The Kernel:

The kernel is the heart and brain of the operating system. The kernel is a layer of software that sits between the user of a computer and its hardware, and is responsible for efficiently

managing the system's resources. It also schedules the work being done on the system so that each task gets its fair share of system resources.

The main control program in a UNIX OS is called the kernel. However, the kernel does not allow the user to give it commands directly; instead when the user types commands on the keyboard they are read by another program in the OS called a shell which parses, checks, translates and massages them in various ways to be described later, then passes them to the kernel for execution.

Functions & services of Kernel:

- file management & Security
- IO services
- Process scheduling & Management
- System Accounting
- Memory management
- Interrupt &Error handling
- Date & Time services
- Inter Process Communication

The Shell: The shell is not part of the kernel, but it does communicate directly with the kernel. It is the "shell around the kernel."

The shell is a **command line interpreter** ,that executes the commands you type in.

It translates your commands from a human-readable format into a format that can be understood by the computer.

In addition to carrying out your commands, the shell also allows you to manage your working environment and is an effective programming language.

It is a user program, which provides an interface between User and Unix Kernel Program.

The user can communicate with a UNIX system through a command program known as a ***shell***.The shell interprets the commands that the user type on the keyboard.

Note: Shell acts as both **Command processor** and a **small programming language.**

There are many different shells available for UNIX computers, and on some systems you can choose the shell in which you wish to work.

You can use shell commands to write simple programs (scripts) to automate many tasks

When you log in, you are in one of **five** shells.

1. Bourne Shll – /bin/sh – Steve Bourne (On AT & T UNIX)
2. Bourne Again SHell – /bin/bash – Improved Bourne shell (On Linux)
3. C Shell – /bin/csh – Bill Joy (On BSD UNIX)
4. TC Shell – /bin/tcsh –Ken Greer, paul placeway,Christos Zoules etc.(On Linux)
5. Korn Shell – /bin/ksh – David Korn (On AT & T UNIX)

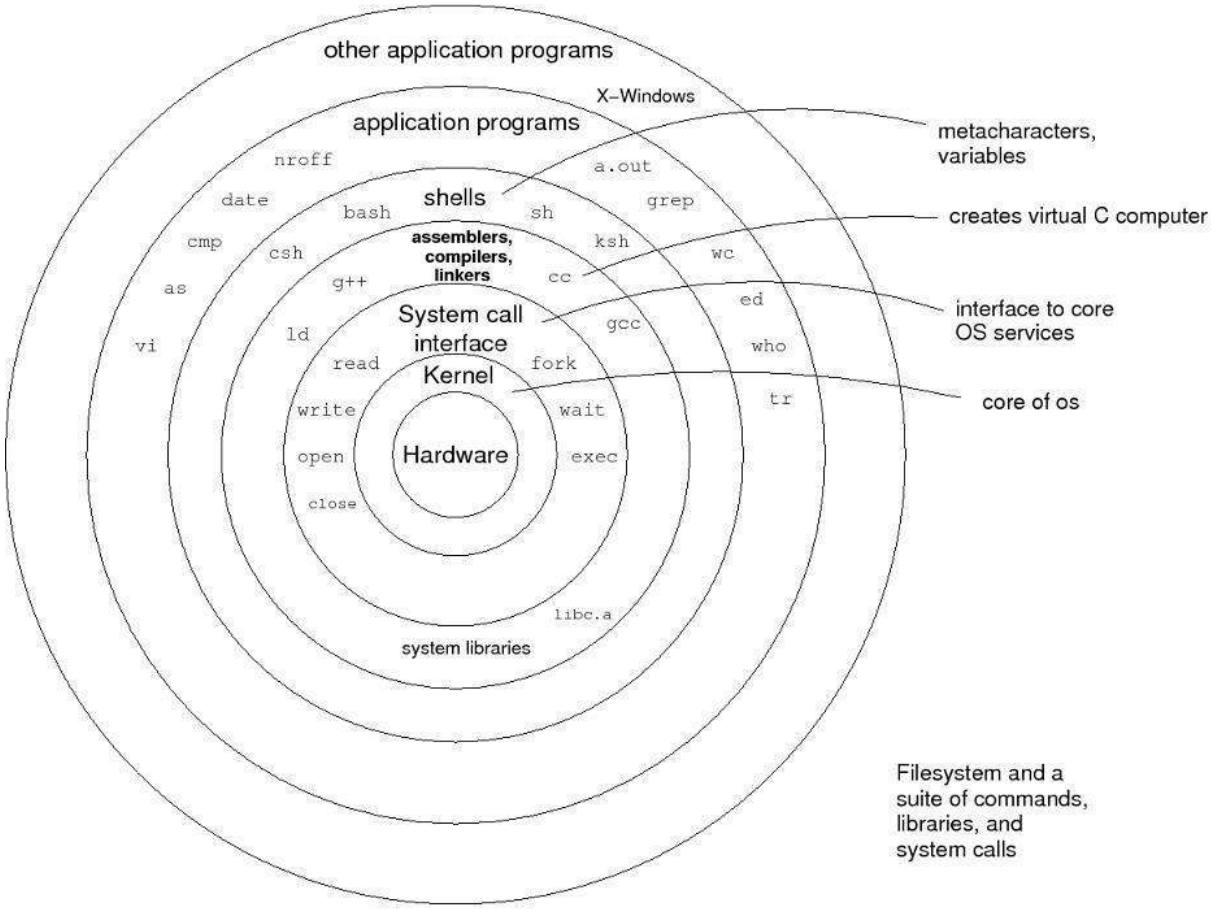
Shell Name	**Developed by**	**Name of the organization**	**Key points**
BASH (Bourne-Again SHell)	Brian Fox and Chet Ramey	Free Software Foundation	Most common shell in Linux. It's Freeware shell.
CSH (C SHell)	Bill Joy	University of California (For BSD)	The C shell's syntax and usage are very similar to the C programming language.
KSH (Korn SHell)	David Korn	AT & T Bell Labs	It is compatible with sh and bash. It improves on the bourne shell by adding floating –point arithmetic, job control, and command aliasing and command substitution.
TCSH(Tenex C-Shell)	Ken Greer, paul placeway,Christos Zoules etc	University of California , Berkeley.	TCSH is an enhanced but completely compatible version of the Berkeley UNIX C shell (CSH).

Main features & functions of Shell:

- Interactive processing(i.e interface between user & the kernel)
- Command Interpretaion & Processing
- Backgroud processing
- I/O redirection
- Pipes
- Meta characters/wild-card character matching
- Shellscripts
- Shell variables

File System: UNIX organises information into **files**, and related files may be conveniently organised in **directories**. Files may contain text, data, executable programs, scripts (which are actually just data for a scripting program such as a shell), and may also be links to other files, or to physical devices or communications channels.

The directory structure is hierarchical with the root directory, indicated by a forward slash (/), at the base of the tree. This may contain files as well as other directories such as /bin, /etc, /lib, /tmp, /usr etc.



Conceptual Architecture of UNIX SYSTEMS

Commands and Utilities − There are various commands and utilities which you can make use of 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 options.

Files and Directories − All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the file system.

5.Using UNIX:

When a user want to use Unix, that user has to get into the unix enivironment.

The process of getting into unix environment is known as "**logging** " in to the system.

Login process steps:

1)**Init** starts **getty** process

2)**getty** process initiates **login prompt** on terminal

3)**login** command **check user credentials** from **/etc/passwd**

4)**getty** starts **user shell** process

5)**shell** reads the system wide files **/etc/profile, /etc/bashrc**

6)**Shell** reads user specific files .**profile, .login**

7)Now it **reads** shell specific configuration file **.bashrc**

8)**Shell** displays the **default prompt**

Init** starts **getty:** **(both are processes):

When Linux system boots it goes through various booting stages. In last stage, it starts init, which reads a file called **inittab** which is located in /etc. Once init process completes execution and executing commands in /etc/rc.local, it will start a process called **getty**. Getty is the process which will take care of complete login process.

Getty shows login prompt:

getty is short for "get terminal". A getty program initiates login command, it opens the terminal device, initializes it, prints login: and waits for a user name to be entered.

getty starts /etc/login:

Once user enters his login name getty starts /ets/login, this in-turn will prompt for user password. The password what user typed will be hidden and will not be shown on screen.

getty verifies the credential and starts users shell:

Here, getty checks the user credentials by verifying it with /etc/passwd and /etc/shadow file, if password matches then it initiates user properties gathering and starts users shell. If password doesn't match then getty terminates login process and re-initiates again with new login: prompt.

In next stage the getty process reads the user properties (username, UID, GID, home directory, user shell etc.) from /etc/passwd file. After that it reads /etc/motd file for displaying content banner message.

Shell reads system wide default files and specific default files:

For getting shell related properties, user aliases and variables getty process reads appropriate system wide default files /etc/profile or /etc/bashrc . After the system wide default files are read the shell reads user specific login files .profile or .login .

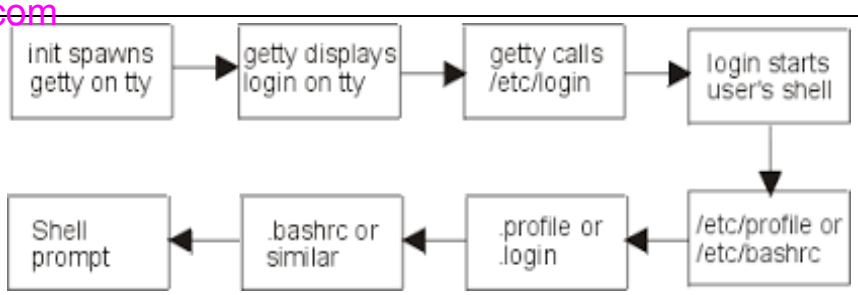
Shell specific file read:

At last stage it reads shell specific configuration files (.bashrc, .bash_profile etc. for bash shell) of that user which it gets on the users home directory.

Shell Prompt: Successful login into a Unix system is indicated by the appearance of a prompt is called "Shell prompt" or "system prompt " on the terminal.

When all startup files are read the shell displays the **default prompt**, normally **PS1 prompt** (**Prompt string 1**)for user to execute their commands, here user can type any command to execute.

$ for Bourne & korn Shells(bh,bash,ksh)
% for C shell (csh & tcsh)
# for Any shell as root.



6.Commands in Unix

Unix has a large number of commands. A list of some **general features** of a Unix are..

1)A Unix command is a **program**, written for performing some task. All such programs have a name. Ex: date, ls,man ,cat etc.

2)All Unix commands are written using lower case letters.

3)Almost all commands are cryptic i.e short code. Ex: cat stands for concatenation, ls for listing and so on.

4)Unix commands have zero , one or more no.of arguments .

5)Unix commands also have “format specifiers ” .

6)A current Unix command can be killed by using either “delete” or “ctrl+u” command.

UNIX commands are classified into two types

1)Internal Commands - Ex: cd, source, date ,pwd,echo etc.

2)External Commands - Ex: ls, cat etc.

Internal Command:

A command that does not have an independent existence in the form of a separate file is called an external command.

Internal commands are something which is built into the shell. For the shell built in commands, the execution speed is really high. It is because no process needs to be spawned for executing it.

For example, when using the "cd" command, no process is created. The current directory simply gets changed on executing it.

External Command:

A command with an independent existence in the form of a separate file is called an external command.

External commands are not built into the shell. These are executables present in a separate file. When an external command has to be executed, a new process has to be spawned and the command gets executed.
For example, when you execute the "cat" command, which usually is at /usr/bin, the executable /usr/bin/cat gets executed.

7)Some Basic Commands:

1)THE "ECHO" COMMAND:

'echo' command- It is used to display the data on the terminal and also display the values of variables. One such variable is 'HOME'. To check the value of a variable precede the variable with a $ sign.

Ex: $ echo $HOME

/home/mani

Ex: $echo "hi"
hi

To get the name of the shell with path name:

Ex: $echo $SHELL
/bin/bash

To get the name of the shell :

Ex: $echo $0
bash

To get simple message:

Ex: echo today date is 'date'
today date is date

To get the current date:

Ex: $ echo today date is `date`

today date is Fri Jun 29 10:45:02 IST 2018

To display the path in PATH variable

Ex: echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

Ex: $ echo "10^3"
10^3

Echo with bc command for calculation

Ex: echo "10^3" | bc
1000

Note: The echo command without an argument prints a **blank line**.

2)THE **TPUT** COMMAND:

This command is used to control the movement of the cursor on the screen as well as to add certain features like blinking,boldface and underlying to the displayed messages on the screen.

This command along with the **clear** argument **clears** the screen.

Ex: tput clear

This command along with the **cup** argument is used to **position** the cursor at any required position on the screen.

Ex: tput cup 20 40

Here , the cursor will be placed at the 20th row and 40th column. If you want to put a cursor at the **center** of the screen, use 50 50 .

Ex:tput cup 50 50

Getting the no.of rows and columns on the current screen:

This command along with the **lines** argument is used to get **number of lines** on the current screen.

Ex: tput lines

24

This command along with the **cols** argument is used to get **number of columns** on the current screen.

Ex: tput cols

80

3)**TTY** COMMAND: (Terminal Type)

This command is used to know the **name of current working device file**. In Unix, every file is associated with a special file called the **Device** file. All the devices will be present in the /**dev** directory.

Ex: $tty

/dev/tty01

Here, tty01 is the **device file name** and it is available in the directory **/dev**.

Note: Terminals are either Hardware terminals or Psudo terminals.

For **Hardware** type terminals , it will be displayed as **tty01**

For **Psudo** type terminals , it will be displayed as **pts01**

For Serial type devices, it will be displayed as **ttys0**

For **USB** based terminals , it will be displayed as **usb0 .**

4)THE **UNAME** COMMAND **: (UNIX NAME)**

It is used to know the details of one's Unix system. it allows various options such as r,v,m & a.

Ex: $uname

Linux

Ex: uname –r …to get release details

2.4.18-3

Ex:uname –m ….to get machine details

i686

uname –v …to get version details

uname –a …..to get all details.

5)THE **DATE** COMMAND:

It is used to display the system date and time. date command is also used to set date and time of the system. By default the date command displays the date in the time zone on which unix/linux operating system is configured.

syntax: date [OPTION]... [+FORMAT]

$ date
Fri Jun 29 10:54:20 IST 2018

u - Option: Displays the time in GMT(Greenwich Mean Time)/UTC(Coordinated Universal Time)time zone.

$date -u

Fri jun 29 06:11:31 UTC 2018

–date or -d Option: Displays the given date string in the format of date. But this will not affect the system's actual date and time value.

Syntax:

$date --date=" string "

Examples:

$date --date="2/06/2018"

$date --date="Feb 6 2018"

Tue Jun 2 00:00:00 PST 2010

Tue Feb 6 00:00:00 PST 2010

Date and time of 2 years ago.

$date --date="2 year ago"

Sat Oct 10 23:42:15 PDT 2015

Date and time of previous day.

$date --date="yesterday"

Mon Oct 9 23:48:00 PDT 2017

-s or –set Option: To set the system date and time -s or –set option is used.

Syntax:

$date --set="date to be set"

$date --set="Tue Nov 13 15:23:34 PDT 2018"

$date

Tue Nov 13 15:23:34 PDT 2018

Formats:

%D: Display date as mm/dd/yy.

%d: Display the day of the month (01 to 31).

%a: Displays the abbreviated name for weekdays (Sun to Sat).

%A: Displays full weekdays (Sunday to Saturday).

%h: Displays abbreviated month name (Jan to Dec).

%b: Displays abbreviated month name (Jan to Dec).

%B: Displays full month name(January to December).

%m: Displays the month of year (01 to 12).

%y: Displays last two digits of the year(00 to 99).

%Y: Display four-digit year.

%T: Display the time in 24 hour format as HH:MM:SS.

%H: Display the hour.

%M: Display the minute.

%S: Display the seconds.

Example:

$date "+%D"

Output:

10/11/17

6)CAL COMMAND :
cal command will print the calendar on **current month** by default. If you want to print calendar of January of 2020. That is the first month of 2020.

Example 1: for getting current month of the year

aliet@aliet-Veriton-Series**:~$ cal**

```
June 2018
Su Mo  Tu We Th Fr Sa
              1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Example2: to get jan 2020.

aliet@aliet-Veriton-Series:~$ **cal jan 2020**

```
    January 2020
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

e**xample3: To get 2020 calendar.**

aliet@aliet-Veriton-Series:~$ **cal 2020**

```
                          2020
      January             February            March
      .....                 .....               .....
      April               May                 June
     .....                .....                .....
      July                August              September
      .....                 .....               .....
     October             November            December
      .....                 .....               .....
```

7)THE **CALENDAR** COMMAND**:**

This command reads calendar file and displays only lines with current day and next day.

It is like an engagement dairy that contains text information and offers a remainder service based on a file called the **calendar**. This file must be in the present working directory/home directory. This file is created and managed by the user with the help of an editor(like vi editor) on the systemThis command works on today and tomorrow dates concept.The present working day's date is taken as **today** and the next working day is considered as **tomorrow.**

Consider the contents of a calendar file are ..

Sep 28 , 2018 freshers day.

on 30/09/18 mock G.R.E test.
First test begins from oct 5,2018.

$calendar
Sep 28 , 2018 freshers day.
on 30/09/18 mock G.R.E test

8)THE **PASSWD** COMMAND**:** (password change)
It is used to change the password by the user during the addition of new users , the system administrator permits or authorizes the new user by assigning unique password to them.
$passwd
old password: *********
new password : **********
new password : **********

9)THE **LOCK** COMMAND**:**
For security reasons, it is necessary to lock the system, when system not in use. It is used for locking session for required amount of time.By default the user can lock it for 30 minutes. This locking period can be changed by assigning a different value for the system variable DEFLOGOUT.
When the lock command is given, the terminal asks for password twice.
password: ********
re-enter password : *********
terminal loced by user1 0 min ago.

$lock -45 …..lock for 45 minutes.

The locked terminal can be unlocked by re-entering the password.
The banner command:
10)BANNER COMMAND:
It is used to writes ASCII character strings in large letters to standard output.
This command is available on SCO (Santa Cruz Operation) Unix system, not on Linux. Banner is used to print characters or display banner or posters . This command simply produces a blown-up version of the characters. It prints a maximum of 10 characters per line.

The **banner** command writes ASCII character *String*s to standard output in large letters.
Each word you input appears on a separate line on the screen. When you want to display more than one word to a line, use quotation marks to specify which words will appear on one line.

syntax: banner *String*

Ex: $banner UBUNTU

The output will be made up of #(hash) characters.

11)CAT COMMAND**:**

It is used to Create small unix files , concatenate and display files.

Syntax: cat [options] filename

Syntax for creating a new file :

cat > filename

Syntax for displaying contents of a file:

cat filename

syntax for concatenate a file:

cat >> filename

The cat (short for "**concatenate**") command is one of the most frequently used command in Linux/Unix like operating systems. **Cat** command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.

Syntax: cat [options] name

> Redirect the output of the cat command to a file rather than standard output. The destination file will be created if it doesn't exist and will be overwritten if it does.

>> Append the output of the cat command to the end of an existing file. The destination file will be created if it doesn't exist.

| Send (or pipe) the output of the cat command into another command for further processing.

Examples

1. **To create a new file ,** use **cat command**

$ **cat > file1**

welcome to unix/linux programming lab

:

:

ctrl+d ------ input termination command

$ cat > file2

working on linux/unix basic commands

ctrl+d ------ input termination command

$ cat > file3

working on cat command with examples

ctrl+d ------ input termination command

2.To display a file at the workstation, enter:

$ cat file1 ------file1 contents

welcome to unix/linux programming lab

$ cat file2 ------file2 contents

working on linux/unix basic commands

$ cat file3 ------file3 contents

working on cat command with examples

$ cat file1 file2 file3 - - - for all 3 - files

welcome to unix/linux programming lab
working on linux/unix basic commands
working on cat command with examples

3.**To copy the data from 2 or more files & placed into another new file, enter:**

$ cat file1 file2 file3 > file4

$ cat file4

welcome to unix/linux programming lab
working on linux/unix basic commands
working on cat command with examples

3)To append one file to the end of another, enter:

cat file1 >> file2

Note: The >> operator appends a copy of file2 to the end of file1 . If you want to **replace** the file, use the > operator

4)To number a file contents including blank lines with –n Option

$ cat -n file4 -----Numbering a file

1welcome to unix/linux programming lab
2working on linux/unix basic commands
3working on cat command with examples

5) To display all contents , we use this option

-A --show-all

$ **cat -A file4**

welcome to unix/linux programming lab
working on linux/unix basic commands
working on cat command with examples

6) **To number the file for non-blank lines only, we use this command**

-b --numbering the file for nonblank lines of text

$ **cat -b file4**

1welcome to unix/linux programming lab
2working on linux/unix basic commands
3working on cat command with examples

7) **To show a $ symbol at the end of each line**

-E --show $ at the end of each line

$ **cat -E file4**

welcome to unix/linux programming lab$
working on linux/unix basic commands$
working on cat command with examples$

8) To disply non-printing characters

-vet ---- displaying non-printing characters in a file

$ cat>npc --- creating a new file npc

this is unix lab
unix is operating system

$cat npc -----displaying contents

this is unix lab
unix is operating system

$ cat -vet npc -----to display non printable characters

this^Iis^Iunix^Ilab$
unix is^Ioperating^Isystem$

Note: In this npc file, tabspace is replaced with ^I and end of every line ends with $.

8.Command Substitution:

Sometimes we want to use the output of one command as part of another command. This may be because we can only effectively determine the desired information by running another command or because it is just easier to run a command than to type the output produced from the command. Command substitution allows us to do just that. The resulting command contains the embedded output of running the command substitution command.

In unix, it is possible to run a command within another command. For example date command can be run within the echo command .

Ex: $echo Toady date is `date`

Today date is Sun jul 1 9:00:00 IST 2018

Ex: $ echo `20+50' | bc

70

EX: $echo "10^3" | bc
1000
Ex: $echo 'ibase=2; obase=A; 1111' | bc --to convert data (1111)from binary to decimal(15).

15

Here, **bc** (basic calculator) command is used as both a calculator and a small language for writing numerical programs.

$bc

sqrt 25

5

Ex: $bc -----to convert data (1111)from binary to decimal (15).

ibase =2

obase=A (10—decimal)

1111

15

9. Giving Multiple Commands:

Unix allows the user to give more than one commands in a single line by use of a semicolon(;) between successive commands.

Syntax: command1;command2;command3,…..command n

Ex: echo “Giving multiple commands”; date; who

It will perform all commands one by one . All commands are executed independently one after another from left to right. It saves time and improves performance of a time. It allows both background & foreground processing.

Note: The bourne shell (sh) does not permit processing of jobs in background , where as Korn shell does.

10.IMPORTANT AND FREQUENTLY ASKED QUESTIONS

1) What is Unix ? Explain the structure of UNIX operating system and its components with the help of neat diagram.

2) List and explain the features of Unix and mention its brief history.

3) Discuss about the commands and types of commands in unix with suitable examples .

4) "Operating systems like UNIX provide services both for programs and users". Justify this statement with suitable examples.

5) Define Shell and kernel. Differentiate them.

6) Explain about the usage of multiple commands on the shell command line with example.

7) Give any five basic commands of UNIX with syntax and examples .

8) Explain briefly about command substitution with examples

9) Give short notes on a) Kernel b) shell c) Multitasking

10) Explain briefly about Using Unix (the Log Process) with the help of diagram.