Unix Shell Programming
Unit 1

Ability Enhancement Course

AEC310

ABILITY ENHANCEMENT COURSE ASSESSMENT :

- The assessment of the Ability Enhancement Course shall comprise of Continuous Internal Evaluation (CIE) of 50 marks and a Semester End Examination (SEE) of 50 marks.
- The CIE will comprise two tests of 30 marks each having Part-A with MCQs and Part-B with descriptive Questions. Another 20 marks of CIE will be assessed through quizzes, Assignments, Presentations, etc.
- The SEE will comprise Part-A with MCQs and Part-B with descriptive Questions

Unit 1

- The architecture of UNIX
- Features of UNIX
- Introduction to UNIX file system
- vi editor
- File handling utilities and security by file permissions
- Basic UNIX commands (PATH, man, echo, printf, script, passwd, uname, who, date, stty, pwd, cd, mkdir, rmdir, ls, cp, mv, rm, cat, more, wc, lp, od, tar, gzip).

What is Unix? UNiplexed Information Computing System

- The Unix operating system is a set of programs that act as a link between the computer and the user.
- The computer programs that allocate the system resources and coordinate all the details of the computer's internals are called the operating system or the kernel.
- Users communicate with the kernel through a program known as the shell. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.
- Unix was originally developed in 1969 by a group of AT&T employees Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna at Bell Labs.
- There are various Unix variants available in the market. Solaris Unix, AIX, HP Unix and BSD are a few examples.
- Linux is also a flavor of Unix which is freely available.

Unix Architecture

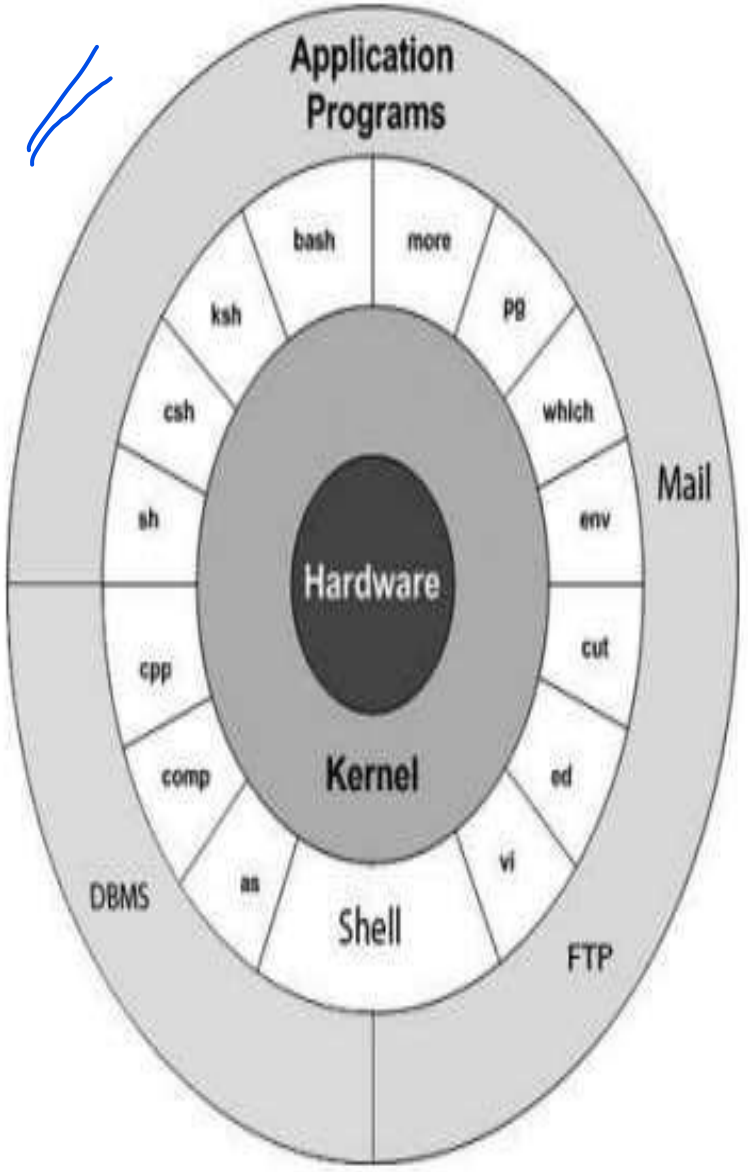
- **Kernel**: The kernel is the heart of the operating system. It interacts with the hardware and performs most of the tasks like memory management, task scheduling, and file management.

It contains two basic parts of the OS:

Process Control and Resource Management: It performs the following tasks

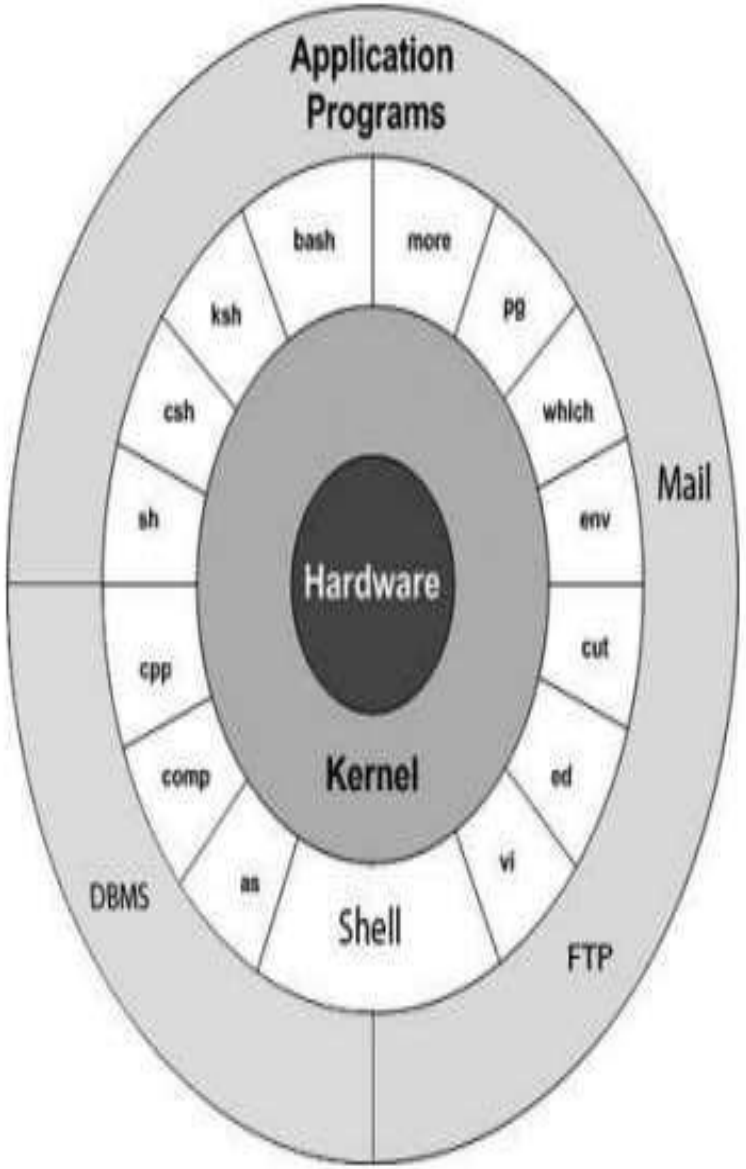
- Process Management
- File Management
- Device Management
- I/O Management
- Memory Management

- **Shell**: The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell, and Korn Shell are the most famous shells which are available with most of the Unix variants.



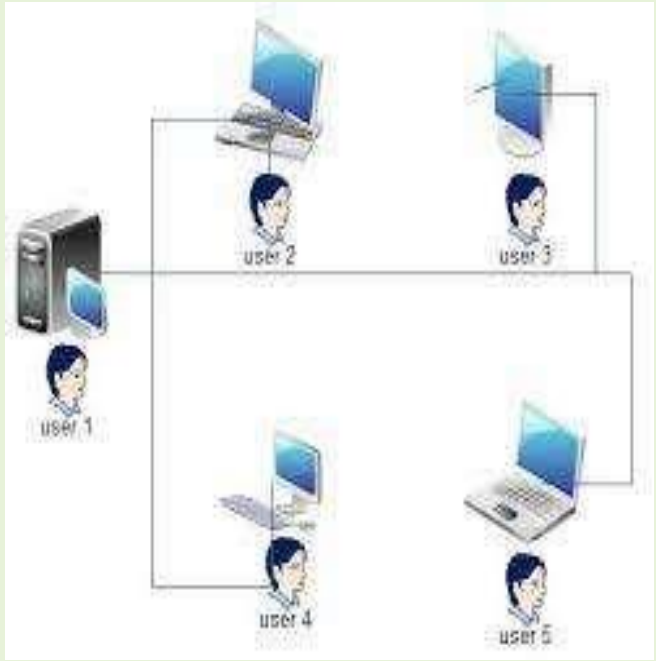
Unix Architecture

- **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 a 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.
- ***Utility: is a standard UNIX program that provides a support process for users. Three common utilities are Text editors, Search Programs, and sort programs.**



Features:

- Portable
- Multiuser
- Multitasking
- Networking
- Organized File System
- Device Independence
- Utilities
- Services



Introduction to UNIX File System:

- Filenames:
  - Length:14 -255 characters
  - Avoid special characters : > or < must not be used
  - Use Dividers: _ ,. ,-
  - Use Extensions
  - Never start a filename with a period. Filenames that start with a period are hidden files in UNIX.

Wildcards:

- **?**: Any Single character.
- […]: Single character from the set
- *: zero or more characters

Ex: c? c?t c??t ?a?

Matches: c1,ca,cat,c12t, bat, car etc….

EX: f[aoei]d f[a-d]t f[A-z][0-9] f[A-Za-z][0-9]

Matches: fad, fod, fed, fid, fat, fa3, f^2 etc….

Does not match: fud, faa….

* : every file

f*: all files which begins with f

*f : all files which ends in f

. every file whose name has a period.

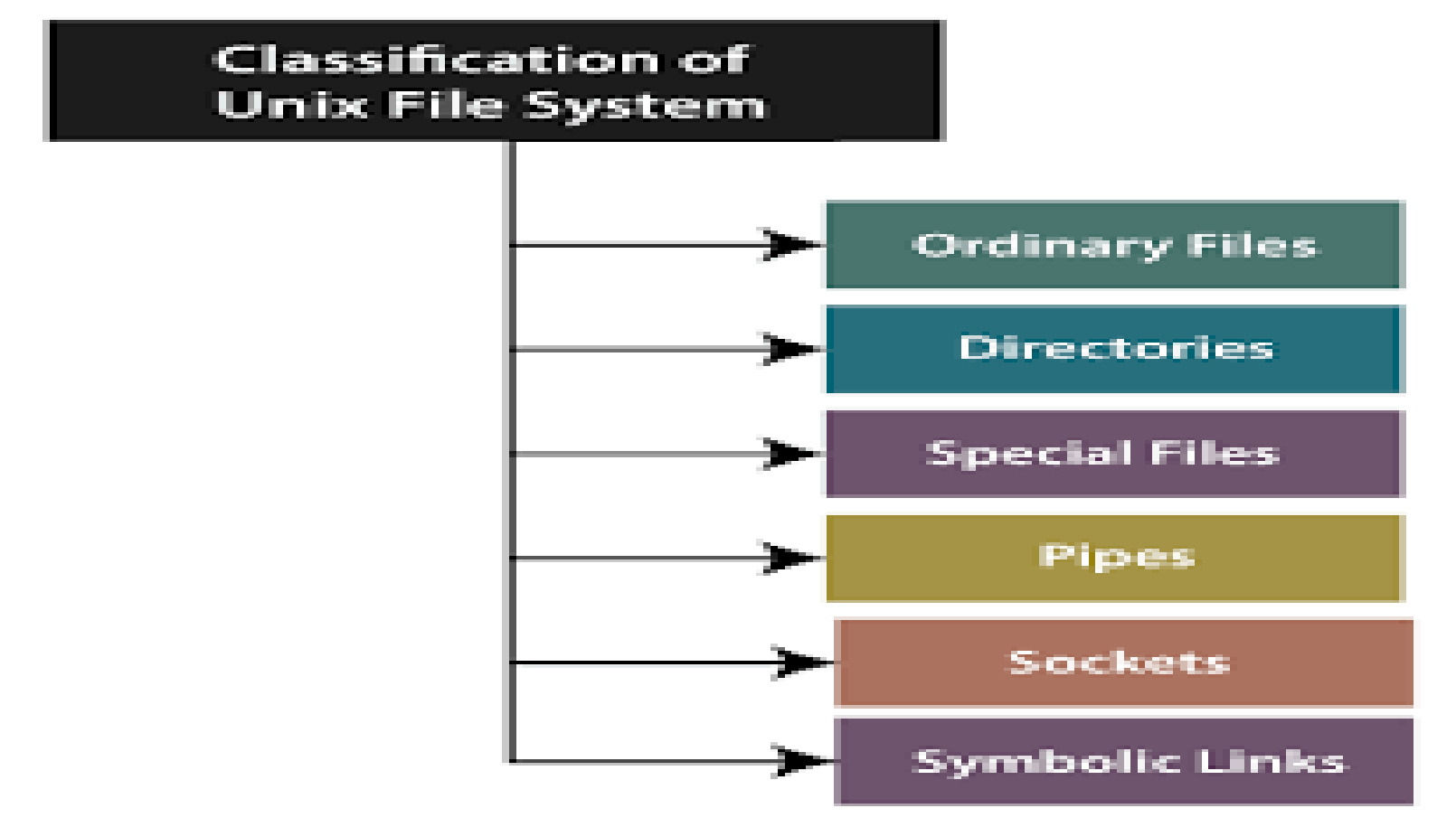
Wildcards with echo command:

$ echo f?t

f1t fat fbt fgt fwt

Wildcards are powerful tools for pattern matching in UNIX, allowing you to work with files and directories more efficiently.

File types


Classification of
Unix File System
Ordinary Files
Directories
Special Files
Pipes
Sockets
Symbolic Links

Listing Files ls

```
$ls

bin         hosts   lib      res.03
ch07        hw1     pub      test_results
ch07.bak    hw2     res.01   users
docs        hw3     res.02   work
```

- To list the files and directories stored in the current directory, use the following command:
- $ls
- The command ls supports the -l option which would help you to get more information about the listed files –
- $ls –l

ls -l

Permissions NoOfMemBlocks OwnerName GroupName SizeInBytes DateCreat/Edited File/dire

In the ls -l listing example, every file line begins with a d, -, or l. These characters indicate the type of file that's listed

```
total 1962188

drwxrwxr-x  2 amrood amrood      4096 Dec 25 09:59 uml
-rw-rw-r--  1 amrood amrood      5341 Dec 25 08:38 uml.jpg
drwxr-xr-x  2 amrood amrood      4096 Feb 15  2006 univ
drwxr-xr-x  2 root   root        4096 Dec  9  2007 urlspedia
-rw-r--r--  1 root   root      276480 Dec  9  2007 urlspedia.tar
drwxr-xr-x  8 root   root        4096 Nov 25  2007 usr
drwxr-xr-x  2    200    300      4096 Nov 25  2007 webthumb-1.01
-rwxr-xr-x  1 root   root        3192 Nov 25  2007 webthumb.php
-rw-rw-r--  1 amrood amrood     20480 Nov 25  2007 webthumb.tar
-rw-rw-r--  1 amrood amrood      5654 Aug  9  2007 yourfile.mid
-rw-rw-r--  1 amrood amrood    166255 Aug  9  2007 yourfile.swf
drwxr-xr-x 11 amrood amrood      4096 May 29  2007 zlib-1.2.3
$
```

-	Regular file, such as an ASCII text file, binary executable, or hard link
b	Block special file. Block input/output device file such as a physical hard drive
c	Character special file. Raw input/output device file such as a physical hard drive
d	Directory file that contains a listing of other files and directories
l	Symbolic link file. Links on any regular file
p	Named pipe. A mechanism for interprocess communications
s	Socket used for interprocess communication

First Column: Represents the file type and the permission given on the file.
Second Column: Represents the number of memory blocks taken by the file or directory.
Third Column: Represents the owner of the file. This is the Unix user who created this file.
Fourth Column: Represents the group of the owner. Every Unix user will have an associated group.
Fifth Column: Represents the file size in bytes.
Sixth Column: Represents the date and the time when this file was created or modified for the last time.
Seventh Column: Represents the file or the directory name.

Hidden Files ls -a

An invisible file is one, the first character of which is the dot or the period character (.). Unix programs (including the shell) use most of these files to store configuration information.

Some common examples of hidden files include the files –

.profile – The Bourne shell (sh) initialization script

.kshrc – The Korn shell (ksh) initialization script

.cshrc – The C shell (csh) initialization script

.rhosts – The remote shell configuration file

To list the invisible files, specify the -a option to ls –

```
$ ls -a

.       .profile    docs    lib     test_results
..      .rhosts     hosts   pub     users
.emacs  bin         hw1     res.01  work
.exrc   ch07        hw2     res.02
.kshrc  ch07.bak    hw3     res.03
```

Display Content of a File

- Display contents of the file:

cat file1

```
$ cat filename
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
```

- Display the contents with the line number

cat -b file1

```
$ cat -b filename
1   This is unix file....I created it for the first time.....
2   I'm going to save this content in this file.
```

- wc command to get a count of the total number of lines, words, and characters contained in a file.

wc file1

NoOfLines Words Chars

options
-l -w -b -m

```
$ wc filename
2  19 103 filename
$
```

Here is the detail of all the four columns –

- **First Column**: Represents the total number of lines in the file.
- **Second Column**: Represents the total number of words in the file.
- **Third Column**: Represents the total number of bytes in the file. This is the actual size of the file.
- **Fourth Column**: Represents the file name.

Copying Files cp o/gfile c/pfile

To make a copy of a file use the cp command. The basic syntax of the command is −

```
$ cp filename copyfile
$
```

Renaming Files moves or renames

To change the name of a file, use the mv command. Following is the basic syntax

mv oldfile newfile

```
$ mv old_file new_file
```

Deleting Files

```
$ rm filename
```

rm file1

```
$ rm filename1 filename2 filename3
```

rm file1 file2 file3

vi Editors

- An editor is a utility that facilitates the editing task- creation /Modification of text files.

- Editors are of 2 types:
  - Line Editors
  - Screen Editors

- Line Editors: changes are applied to a line or a group of lines. They are useful when you want to make global changes over a group of lines. Ex: if extra space is to be added at the start of the line, all the lines can be selected at once, instead of adding to each line.
- Screen Editors: Presents a whole screen of text at a time. Each line of the text in context with other lines can be viewed at once.

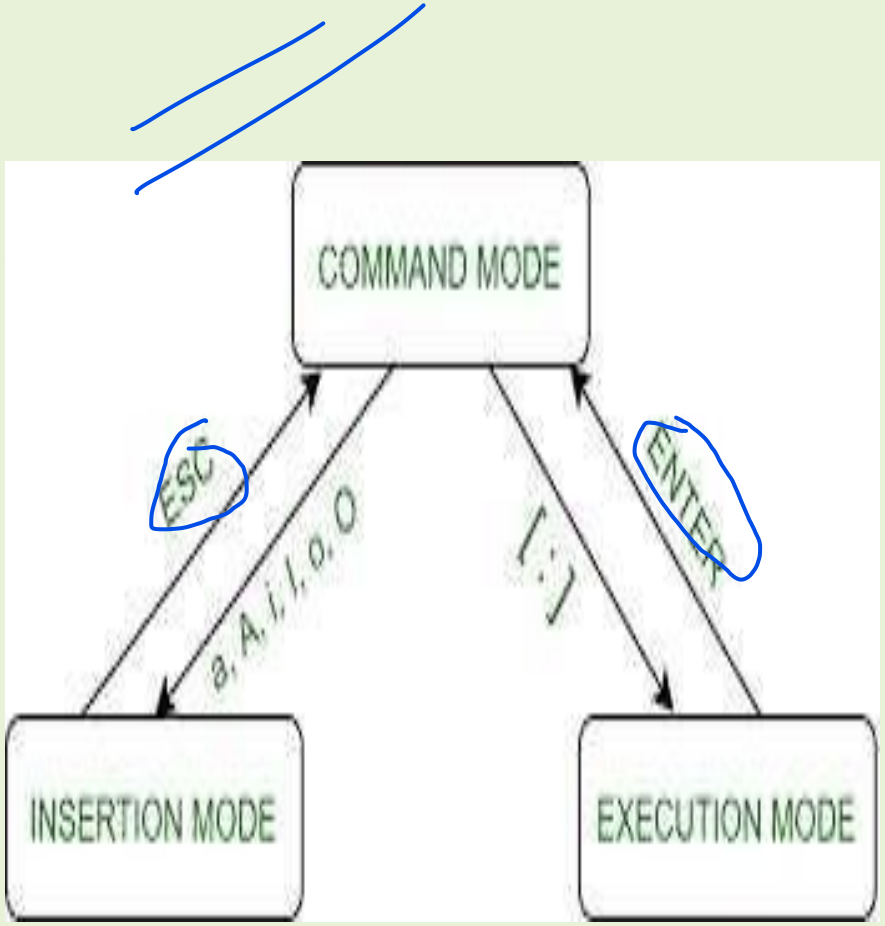
```
vi test.txt
```

Starting the vi Editor

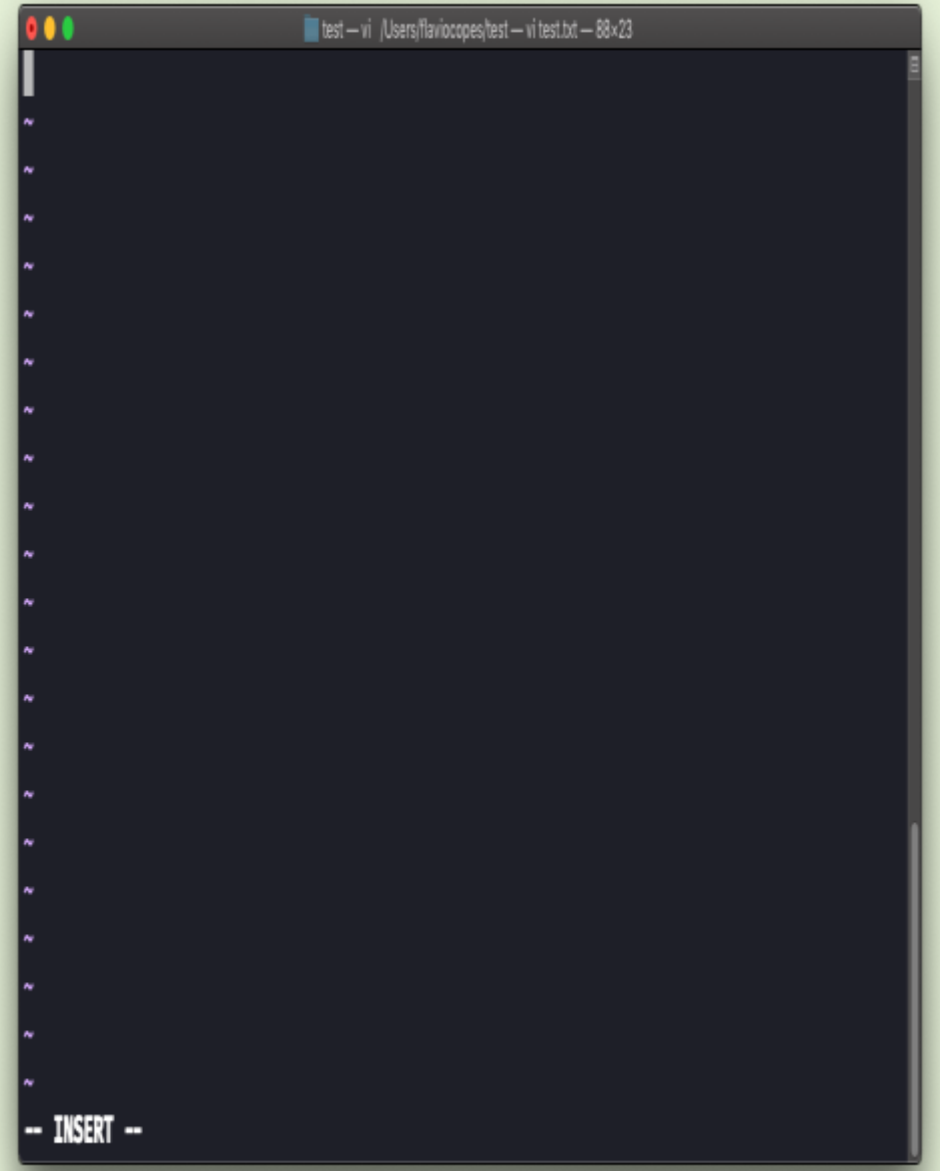
Command	Description
vi filename	Creates a new file if it already does not exist, otherwise opens existing file.
vi -R filename	Opens an existing file in read only mode.
view filename	Opens an existing file in read only mode.

2 Modes:

- **Command mode** – This mode enables you to perform administrative tasks such as saving files, executing commands, moving the cursor, cutting and pasting lines or words, and finding and replacing. In this mode, whatever you type is interpreted as a command.
- **Insert mode** – This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and finally, it is put in the file.



Execution Mode



You can move around the file with the arrow keys, or use `h` - `j` - `k` - `l` keys. `h-l` for left-right, `j-k` for down-up.

Once you are done editing you can press the `esc` key to exit insert mode, and go back to **command mode**.

```
    k          ^
h   j   l   <- | ->
```

Command	Description
k	Moves the cursor up one line.
j	Moves the cursor down one line.
h	Moves the cursor to the left one character position.
l	Moves the cursor to the right one character position.

Editing Files

inserting options
i I a A o O

- To edit the file, you need to be in the insert mode. There are many ways to enter insert mode from the command mode –

Command	Description
i	Inserts text before current cursor location.
I	Inserts text at beginning of current line.
a	Inserts text after current cursor location.
A	Inserts text at end of current line.
o	Creates a new line for text entry below cursor location.
O	Creates a new line for text entry above cursor location.

Deleting Characters

deleting options
x X dw d^ d$ D dd

Command	Description
x	Deletes the character under the cursor location.
X	Deletes the character before the cursor location.
dw	Deletes from the current cursor location to the next word.
d^	Deletes from current cursor position to the beginning of the line.
d$	Deletes from current cursor position to the end of the line.
D	Deletes from the cursor position to the end of the current line.
dd	Deletes the line the cursor is on.

Vi save and Exit Command

- :w - saves the contents of the buffer without quitting vi
- :w filename – Write contents of buffer to new file and continues.
- ZZ – saves the contents of the buffer and exit
- :wq - saves the contents of the buffer and exit
- :q - Exit from vi if only buffer is unchanged
- :q! – Exit from vi without saving.

save and exit

w w cpfile wq ZZ q q!

Category	Command	Description
Adding text	i	Inserts text before the current character.
	I	Inserts text at the beginning of the current line.
	a	Appends text after the current character.
	A	Adds text at the end of the current line.
	o	Opens an empty text line for new text after the current line.
	O	Opens an empty text line for new text before the current line.
Deleting text	x	Deletes the current character.
	dd	Deletes the current line.
Moving Cursor	h, <=, Backspace	Moves the cursor one character to the left.
	l, =>, Spacebar	Moves the cursor one character to the right.
	0	Moves the cursor to the beginning of the current line.
	$	Moves the cursor to the end of the current line.
	k, ⇑	Moves the cursor one line up.
	j, ⇓	Moves the cursor one line down.
	-	Moves the cursor to the beginning of the previous line.
	+, return	Moves the cursor to the beginning of the next line.
Join	J	Joins two consecutive lines.
Undo	u	Undoes only the last edit.
	U	Undoes all changes on the current line.
Scrolling	ctrl+y	Scrolls up one line.
	ctrl+e	Scrolls down one line.
	ctrl+u	Scrolls up half a screen.
	ctrl+d	Scrolls down half a screen.
	ctrl+b	Scrolls up whole screen.
	ctrl+f	Scrolls down whole screen.
Saving and Quitting	:w	Saves buffer contents to original file and continues.
	:w filename	Saves buffer contents to filename and continues.
	:wq , ZZ	Saves the contents of the buffer and exits.
	:q	Quits if contents have not been changed.
	:q!	Exits **vi** without saving.

Basic UNIX Commands

PATH

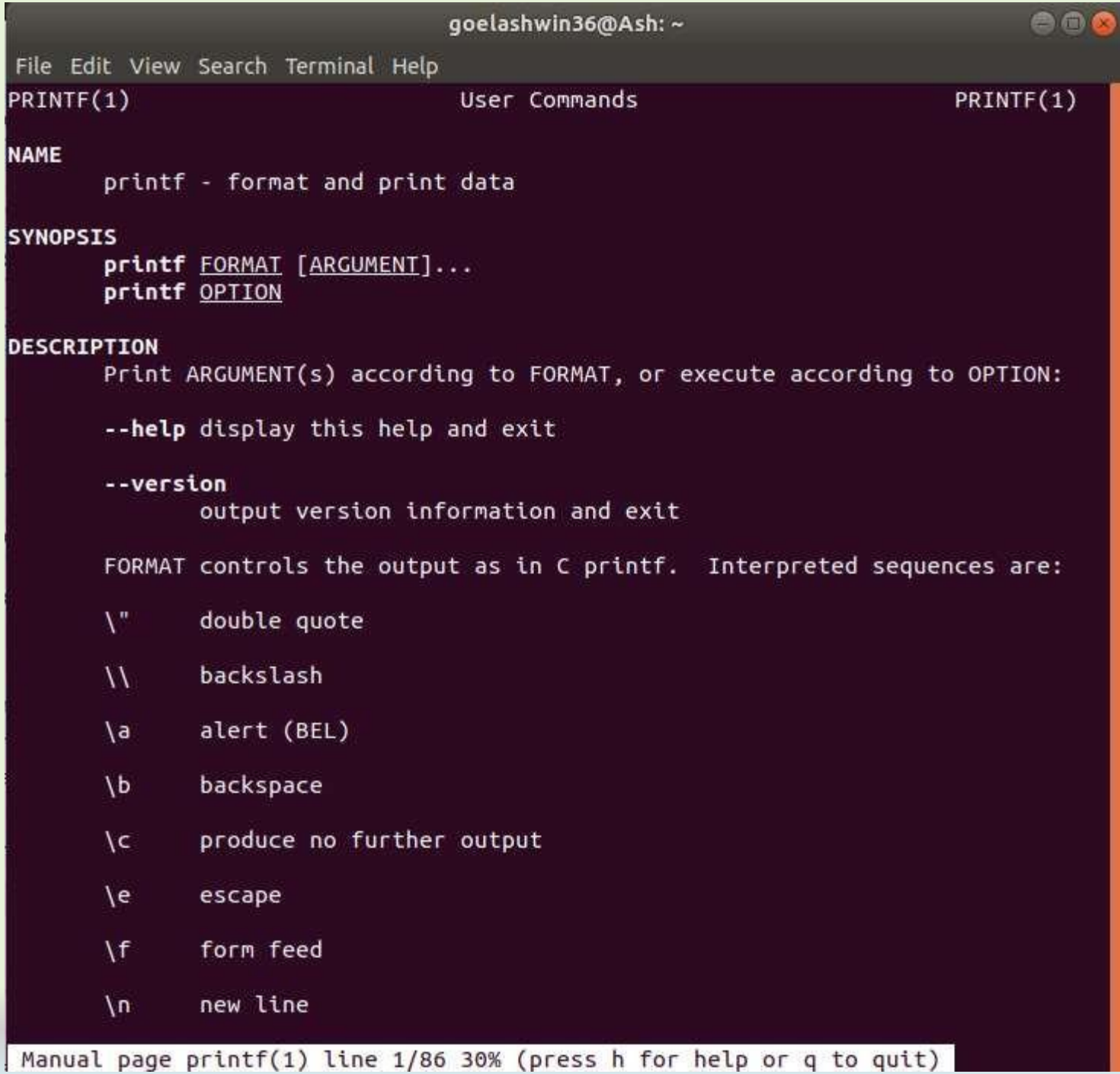
- PATH is an environment variable used by the shell to determine where to look for executable programs.

- PATH is interpreted as a list of directory names separated by a colon. When any command is typed on the command line, the SHELL looks up into each PATH directory in the list in turn until the executable file with the command name is encountered

- $PATH

man

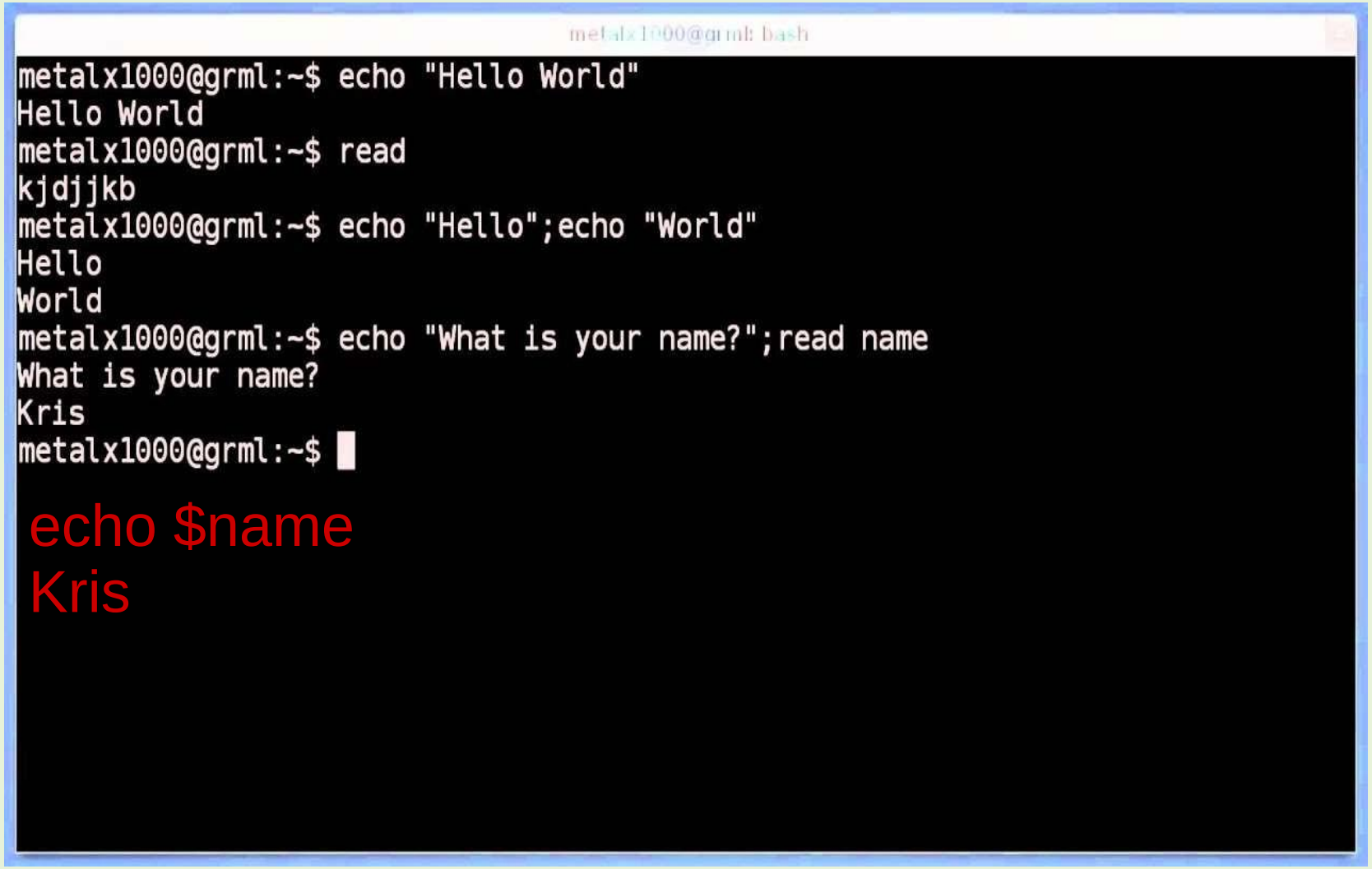
- man command in Linux is **used to display the user manual of any command that we can run on the terminal**



echo

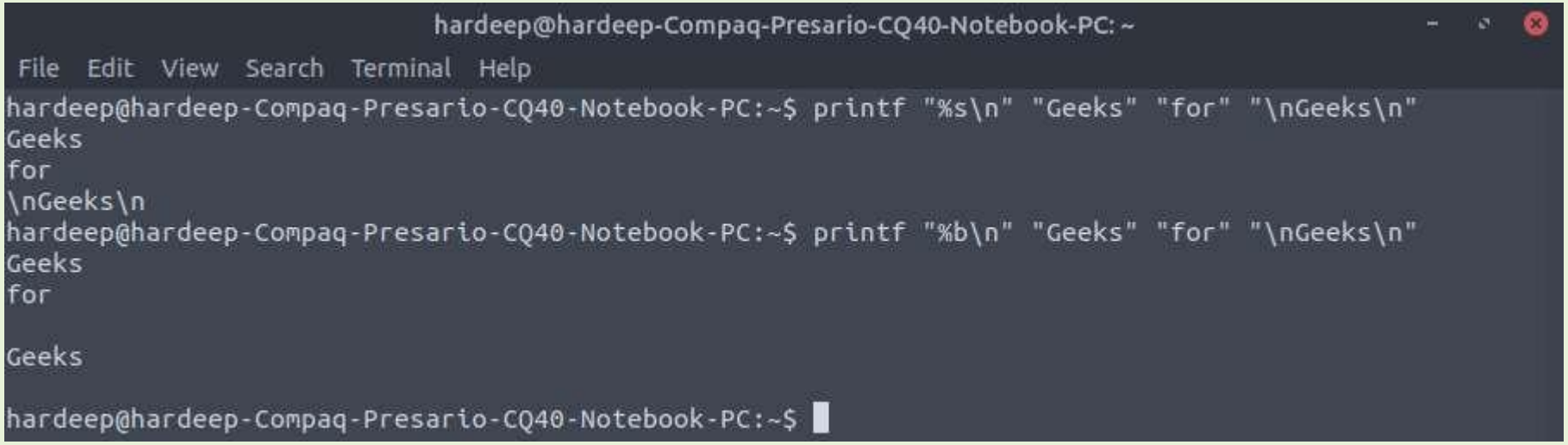
- Echo is a Unix/Linux command tool used for **displaying lines of text or string which are passed as arguments on the command line.** This is one of the basic command in linux and most commonly used in shell scripts.

echo - printf
read - scanf



printf

- In Unix and Unix-like operating systems, printf is a shell builtin that formats and prints data
- *%s* specifier: It is basically a string specifier for string output.
- *%b specifier*: It is same as string specifier but it allows us to interpret escape sequences with an argument.
- *%d specifier:* It is an integer specifier for showing the integral values
- *%f specifier:* It is used for output of floating point values.



script record all the term act / typescript

- **script** command in Linux is used to make typescript or record all the terminal activities.
- After executing the ***script* command** it starts recording everything printed on the screen including the inputs and outputs until exit.
- By default, all the terminal information is saved in the file ***typescript***, if no argument is given. The *script* is mostly used when we want to capture the output of a command or a set of commands while installing a program or the logs generated on the terminal while compiling open source codes, etc. *script* command uses *two files* i.e. one for the **terminal output** and other for the **timing information**.

passwd

- The **passwd** command changes passwords for user accounts. A normal user may only change the password for their own account, while the superuser may change the password for any account. passwd also changes the account or associated password validity period.
- Options
  - passwd -S <username>
  - The -S option displays the status of user account password settings.

  - For example:
  - # passwd -S evans
  - evans PS 2020-09-07 0 99999 7 -1 (Password set, SHA512 crypt.)
  - The output above shows the account **evans** was created on 7th September 2020 and has a password set with SHA512 encryption.

- **passwd -u <username>**

This option will unlock the password. This option works for an account that already has the password locked.

$passwd -u user2

- **passwd -l <username>**

$passwd -l user1

- **passwd -d <username>**

This is a quick way to delete a password for an account.

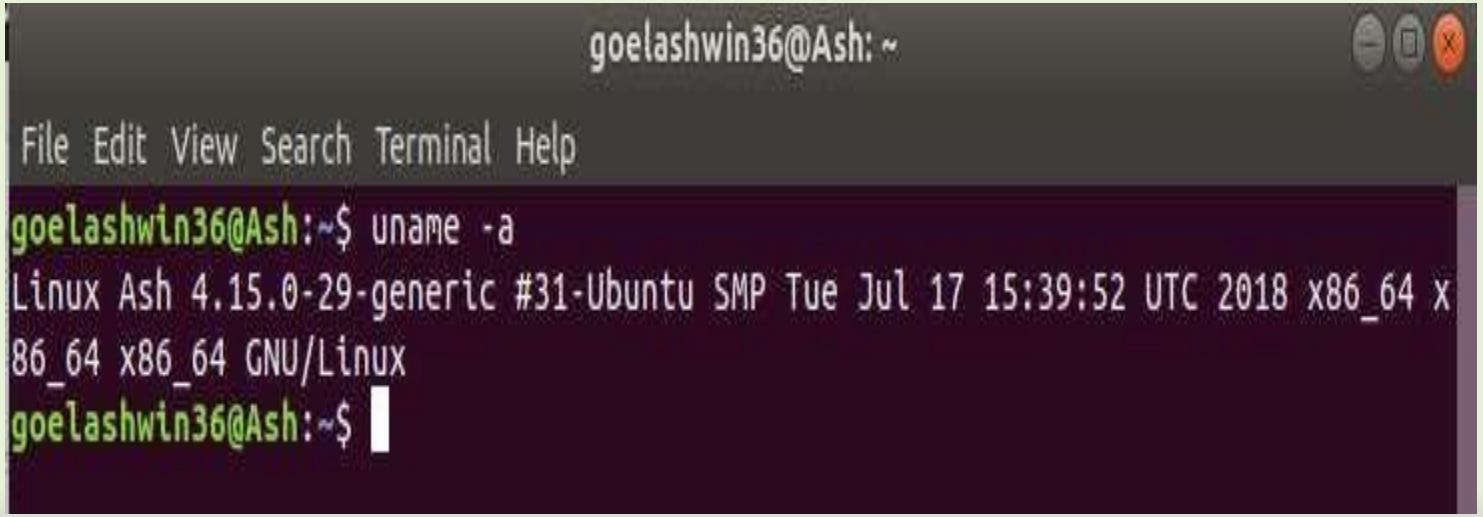
- **passwd -e <username>**

This is a quick way to expire a password for an account. The user will be forced to change the password during the next login attempt.

uname uname -a

- The command '*uname*' displays the information about the system.
  - uname [OPTION]

1. -a option: It prints all the system information in the following order: *Kernel name, network node hostname, kernel release date, kernel version, machine hardware name, hardware platform, operating system*



who

- The standard Unix command who displays a list of users who are currently logged into the computer.

  - Who -m -H (**Command to display the hostname and user associated with the input/output devices**)
  - who -a (**To display all details of currently logged in users**)
  - Who -all
  - who -p -h (**all active processes that are spawned by the NIT process)**
  - who -d -H (complete list of all dead processes)

date

- **date** command 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. You must be the super-user (root) to change the date and time.

Options with Examples:

1. **$date:** current date and time, including the abbreviated day name, abbreviated month name, day of the month, the time separated by colons, the time zone name, and the year.

   Tue Oct 10 22:55:01 PDT 2017

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

3. **Using –date option for displaying past dates:**

   $date --date="2 year ago"

tty and stty

- tty command writes the name of your terminal to standard output
- **stty** displays or changes the characteristics of the terminal.

Options

-a, --all	Print all current settings in human-readable form.
-g, --save	Print all current settings in a **stty**-readable form.
-F, --file=*DEVICE*	Open and use the specified *DEVICE* instead of stdin.
--help	Display a help message, and exit.
--version	Output version information, and exit.

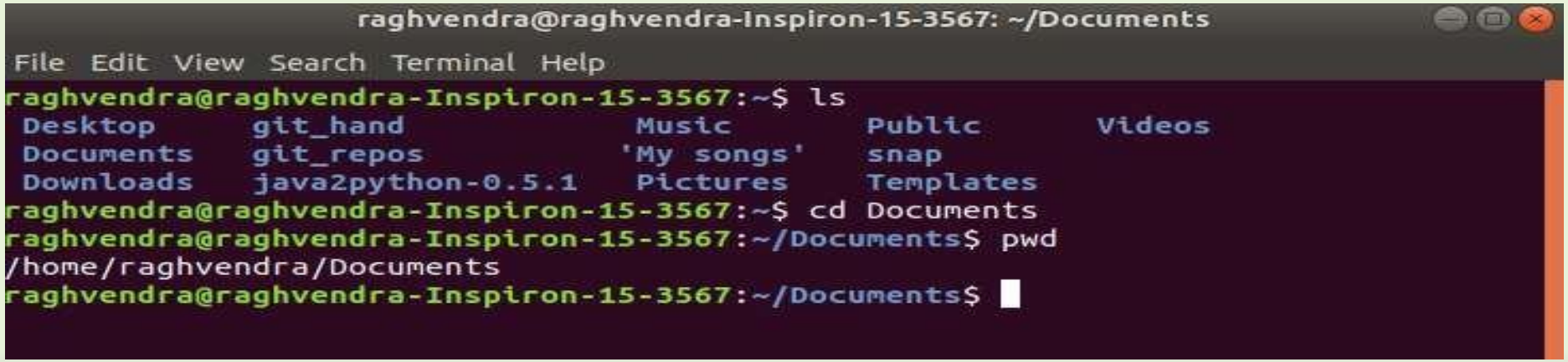
pwd print working directory

- The pwd command **writes to standard output the full path name of your current directory (from the root directory)**. All directories are separated by a / (slash). The root directory is represented by the first /, and the last directory name is your current directory.

- $pwd
  - (Print Working Directory)

cd cd / ~ ..

- cd command known as **change directory command**.
- **cd /:** this command is used to change directory to the root directory, The root directory is the first directory in your file system hierarchy.
- **cd ~ :** this command is used to change directory to the home directory.
- **cd .. :** this command is used to move to the parent directory of current directory, or the directory one level up from the current directory. ".." represents parent directory.



mkdir

mkdir
mkdir -v dir1 dir2 dir3 ....

- **mkdir** command allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have enough permissions to create a directory in the parent directory, or he/she may receive a “permission denied” error.

more more opt no.Lines repPattn lineNo. file1

-d -f -p -c -s -u

- more command is **used to view the text files in the command prompt, displaying one screen at a time in case the file is large** (For example log files). The more command also allows the user do scroll up and down through the page.
- ***more [-options] [-num] [+/pattern] [+linenum] [file_name]***
- ***[-options]****: any option that you want to use in order to change the way the file is displayed. Choose any one from the followings: (-d, -l, -f, -p, -c, -s, -u)*
- ***[-num]****: type the number of lines that you want to display per screen.*
- ***[+/pattern]****: replace the pattern with any string that you want to find in the text file.*
- ***[+linenum]****: use the line number from where you want to start displaying the text content.*
- ***[file_name]****: name of the file containing the text that you want to display on the screen.*

Options:

- **-d :** Use this command in order to help the user to navigate. It displays "[Press space to continue, 'q' to quit.]" and displays "[Press 'h' for instructions.]" when wrong key is pressed.
- **-f :** This option does not wrap the long lines and displays them as such.
- **-p :** This option clears the screen and then displays the text.
- **-c :** This command is used to display the pages on the same area by overlapping the previously displayed text.
- **-s :** This option squeezes multiple blank lines into one single blank line.
- **-u :** This option omits the underlines.

rmdir: Removes the directory from the system. The directory must be empty before you can remove it.
Use the ls -al command to check whether the directory is empty.

ls: Used to list files and directories. The contents of your current working directory.

cp: This command is used to copy files or groups of files or directories. It creates an exact image of a file on a disk.

mv: Moves files and directories from one directory to another or renames a file or directory.

moves and renames

rm: Use the rm command to remove files you no longer need. The rm command removes the entries for a specified file, group of files, or certain select files from a list within a directory.

cat: To print the content of a file onto the standard output. concatenation of files which combines multiple files into a single file.

```
cat f1      - writes  to std o/p
cat f1 f2   - adds f2 to f1
```

wc

wc
-l -w -b -m

- wc stands for word count. As the name implies, it is mainly used for counting purpose. It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.
- By default it displays four-columnar output.
- First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.
  - Syntax: wc [OPTION]... [FILE]...
- **-l:** This option prints the **number of lines** present in a file. With this option wc command displays two-columnar output, 1st column shows number of lines present in a file and 2nd itself represent the file name.
- **-w:** This option prints the **number of words** present in a file. With this option wc command displays two-columnar output, 1st column shows number of words present in a file and 2nd is the file name.
- **-c:** This option displays **count of bytes** present in a file. With this option it display two-columnar output, 1st column shows number of bytes present in a file and 2nd is the file name.
- **-m:** Using **-m** option ‘wc’ command displays **count of characters** from a file.

- **lp:** submits files for printing or alters a pending job..

Tag	Description
--	Marks the end of options; use this to print a file whose name begins with a dash (-).
-E	Forces encryption when connecting to the server.
-U username	Specifies the username to use when connecting to the server.
-C	This option is provided for backwards-compatibility only. On systems that support it, this option forces the print file to be copied to the spool directory before printing. In CUPS, print files are always sent to the scheduler via IPP which has the same effect.
-d destination	Prints files to the destination printer.
-h hostname[:port]	Chooses an alternate server.
-i job-id	Specifies an existing job to modify.
-m	Sends an email when the job is completed.
-n copies	Sets the number of copies to print from 1 to 100.
-o "name=value [name=value ...]"	Sets one or more job options.
-q priority	Sets the job priority from 1 (lowest) to 100 (highest). The default priority is 50.
-s	Do not report the resulting job IDs.
-t "name"	Sets the job name.
-H hh:mm-H hold-H immediate-H restart-H resume	Specifies when the job should be printed. A value of immediate will print the file immediately, a value of hold will hold the job indefinitely, and a UTC time value (HH:MM) will hold the job until the specified UTC (not local) time. Use a value of resume with the -i option to resume a held job. Use a value of restart with the -i option to restart a completed job.
-P page-list	Specifies which pages to print in the document. The list can contain a list of numbers and ranges (#-#) separated by commas (e.g. 1,3-5,16). The page numbers refer to the output pages and not the document's original pages - options like "number-up" can affect the numbering of the pages.

od

od
converting to octal/decimal/ASCII
useful in reading files not in human readable format
Ex: .exe files

- od command in Linux is used to convert the content of input in different formats with the octal format as the default format. This command is especially useful when debugging Linux scripts for unwanted changes or characters. If multiple files are specified, the od command concatenates them in the listed order to form the input. It can display output in a variety of other formats, including hexadecimal, decimal, and ASCII. It is useful for visualizing data that is not in a human-readable format, like a program's executable code.
- Syntax: od [OPTION]... [FILE]...
- **-b Option:** It displays the contents of input in octal format.
- **-c Option:** It displays the contents of input in character format.
- **-An Option:** It displays the contents of input in character format but with no offset
- **-A Option:** It displays input contents in different formats by concatenating some special character with -A.

tar tar opt archFile file/DirToBeArch

- The Linux ‘tar’ stands for tape archive, which is used to create Archive and extract the Archive files. tar command in Linux is one of the important commands which provides archiving functionality in Linux. We can use the Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.
- Syntax:
- tar [options] [archive-file] [file or directory to be archived]
- Options:
- -c: Creates Archive
- -x: Extract the archive
- -f: creates archive with the given filename
- -t: displays or lists files in the archived file
- -u: archives and adds to an existing archive file
- -v: Displays Verbose Information
- -A: Concatenates the archive files
- -z: zip, tells tar command that creates tar file using gzip
- -j: filter archive tar file using tbzip
- -W: Verify an archive file
- -r: update or add file or directory in an already existing .tar file

gzip gzip compresses files

- gzip command compresses files. Each single file is compressed into a single file. The compressed file consists of a GNU zip header and deflated data. If given a file as an argument, gzip compresses the file, adds a “.gz” suffix, and deletes the original file. With no arguments, gzip compresses the standard input and writes the compressed file to standard output.

File Permissions

Every file in Unix has the following attributes –

- Owner permissions – The owner's permissions determine what actions the owner of the file can perform on the file.
- Group permissions – The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- Other (world) permissions – The permissions for others indicate what action all other users can perform on the file.

The Permission Indicators

- While using ls -l command, it displays various information related to file permission as follows :

```
$ls -l /home/amrood
-rwxr-xr--  1 amrood  users 1024  Nov 2 00:10  myfile
drwxr-xr--- 1 amrood  users 1024  Nov 2 00:10  mydir
```

- The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) –
- The first three characters (2-4) represent the permissions for the file's owner. For example, -rwxr-xr-- represents that the owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, -rwxr-xr-- represents that the group has read (r) and execute (x) permission, but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, -rwxr-xr-- represents that there is read (r) only permission.

File Access Modes

- The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the read, write, and execute permissions, which have been described below –
- Read Grants the capability to read, i.e., view the contents of the file.
- Write Grants the capability to modify, or remove the content of the file.
- Execute Users with execute permissions can run a file as a program.

Directory Access Modes

- Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned:
- Read Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.
- Write Access means that the user can add or delete files from the directory.
- Execute Executing a directory doesn't really make sense, so think of this as a traverse permission. A user must have execute access to the bin directory in order to execute the ls or the cd command.

user/group/others

Changing Permissions

chmod u/g/o +/-/= (perms)

- To change the file or the directory permissions, you use the chmod (change mode) command. There are two ways to use chmod — the symbolic mode and the absolute mode.
- Using chmod in: The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table. Symbolic Mode

Symbolic Mode

chmod Operator	Description
+	Adds the designated permission(s) to a file or directory.
-	Removes the designated permission(s) from a file or directory.
=	Sets the designated permission(s).

```
$ls -l testfile

-rwxrwxr--  1 amrood   users 1024  Nov 2 00:10  testfile
```

Then each example chmod command from the preceding table is run on the testfile, followed by ls –l, so you can see the permission changes –

```
$chmod o+wx testfile
$ls -l testfile
-rwxrwxrwx  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod u-x testfile
$ls -l testfile
-rw-rwxrwx  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod g=rx testfile
$ls -l testfile
-rw-r-xrwx  1 amrood   users 1024  Nov 2 00:10  testfile
```

```
$chmod o+wx,u-x,g=rx testfile
$ls -l testfile
-rw-r-xrwx  1 amrood   users 1024  Nov 2 00:10  testfile
```

Using chmod with Absolute mode

chmod uNo.gNo,oNo. filename

- The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file. Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set

0-No
1-Exe
2-Write
4-Read
Others
Combination

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

Here's an example using the testfile. Running **ls -1** on the testfile shows that the file's permissions are as follows –

```
$ls -l testfile
-rwxrwxr--  1 amrood   users 1024  Nov 2 00:10  testfile
```

Then each example **chmod** command from the preceding table is run on the testfile, followed by **ls –l**, so you can see the permission changes –

```
$ chmod 755 testfile
$ls -l testfile
-rwxr-xr-x  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod 743 testfile
```

```
$ls -l testfile
-rwxr---wx  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod 043 testfile
$ls -l testfile
----r---wx  1 amrood   users 1024  Nov 2 00:10  testfile
```

Changing Owners and Groups

While creating an account on Unix, it assigns a owner ID and a group ID to each user. All the permissions mentioned above are also assigned based on the Owner and the Groups.

Two commands are available to change the owner and the group of files –

chown – The chown command stands for "change owner" and is used to change the owner of a file.

chgrp – The chgrp command stands for "change group" and is used to change the group of a file

```
$ chown user filelist
```

```
$ chgrp group filelist
```

Write the commands for :

- Rahul wants to keep documentation of the work he does on the command prompt. script
- Delete the password of the system. passwd -d
- Seema wants to know the version of the kernel of the system she is working. stty --version
- Ram wants to know the users currently logged in who -a
- Change the input terminal stty
- Count the number of lines in the file wc -l
- Print the file and send the message when the job is done. lp
- Compress the file gzip