BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)
CS F111 Computer Programming
LAB SESSION #1
(Basic of Unix; File and Directory Commands in Unix)

A computer is a device that takes data as input, does some processing, and then returns an output. The following block diagram (Figure 1) shows the essential components of a computer.

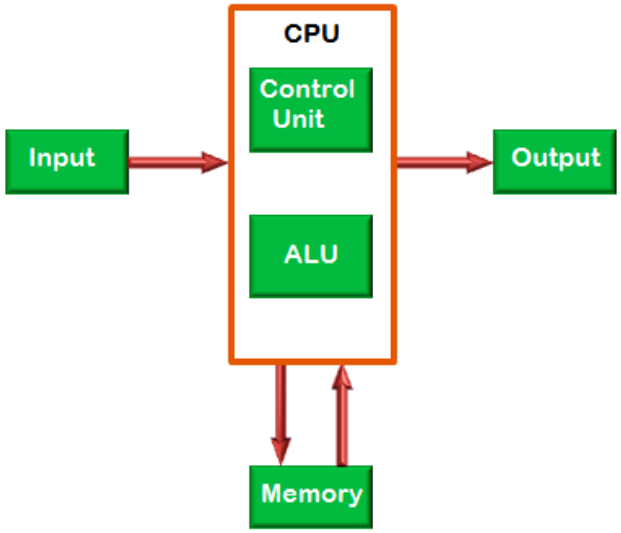

Figure 1: Components of a Computer

We can run programs on computers to process data and perform various tasks. In fact, all tasks that can be done on a computer require us to write programs to do. In other words, programs are means to perform tasks on computers.

Programs, when written and saved, reside on the **disk** (a kind of memory also known as secondary memory. It is not volatile). Figure 2 illustrates this. When we want to execute a program (to perform an intended task), it first gets loaded to primary memory or **RAM** (volatile memory but supports faster access than disk) and then executed on the **CPU**. Figure 2 illustrates this process.

Any computer needs someone to manage the hardware resources such as disk, RAM, CPU, Input/Output devices such as keyboard, mouse, printer, etc. There is a manager also needed to control and coordinate the execution of programs (or processes). This manager is the **Operating System** or the **OS**. *In essence, the operating system is a specialized program that controls and monitors the execution of all other programs that reside in the computer, including application programs (such as user-written programs, MS word, MS excel, Paintbrush, etc.) and other system software.* The primary functionalities of any operating system include:

- ***Memory management (RAM, DISK, ROM, etc.)***
- ***Processor management (Processor or CPU are the same things)***
- ***Device management (Keyboard, mouse, printer, Ethernet card, etc)***
- ***Security (Firewall)***
- ***Coordination between other software and users***
- ***etc.***

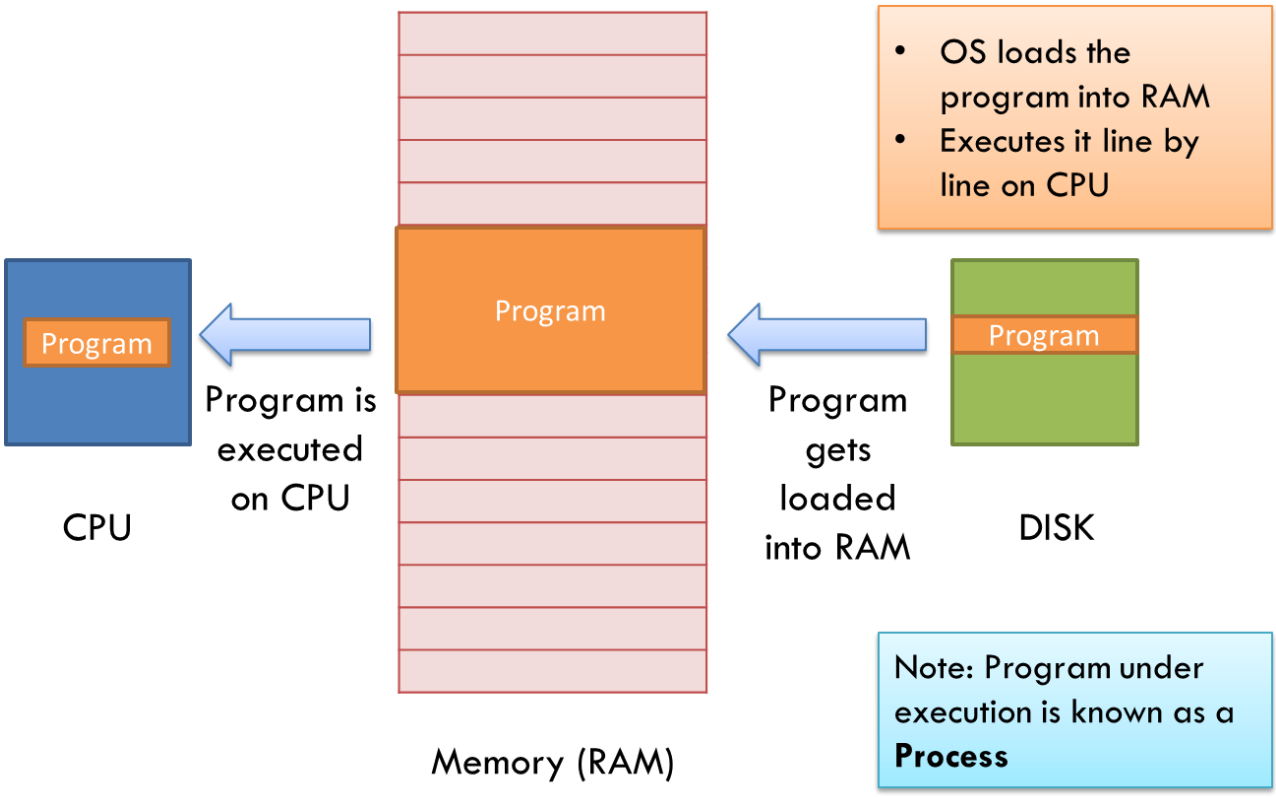

Figure 2: How Programs are Executed

A few examples of Operating Systems include – ***Windows, Mac OS X, Ubuntu Linux, RHEL, Fedora Linux, etc.***

Unix is a C-based operating system, originally developed in 1969 at AT&T Bell labs. It is a multi-user, multi-tasking operating system. A multi-user system allows more than one user to use the system simultaneously, and in a multi-tasking system, more than one task (or process) can be performed simultaneously. There are various Unix variants available in the market such as Solaris Unix, AIX, HP Unix, SCO Unix, BSD, etc. Older versions of UNIX were command-line interfaces without any graphics. Recent versions have Graphical interfaces with support for command-line interfaces as well.

Linux refers to a flavor of Unix that is freely available. A few variants of Linux include Ubuntu, CentOS, RHEL, etc. Linux variants generally support graphical interfaces along with a command-line interface.

For all our laboratory sessions of this course, we will be using the Ubuntu Linux as the Operating System on our computers. As explained before, Ubuntu is a variant of Linux and is available free of cost. It has a graphical interface and also supports a command-line interface through the **terminal**. In today's lab, we will be learning how to use the terminal and how to give Unix commands to the terminal to execute various tasks. We will also learn different types of Unix commands supported by the Ubuntu terminal. Generally, all variants of Linux/Unix support the same set of commands). Also, note that we will interchangeably use the word Unix and Linux. Before we study various Unix commands, let us first study Unix Architecture.

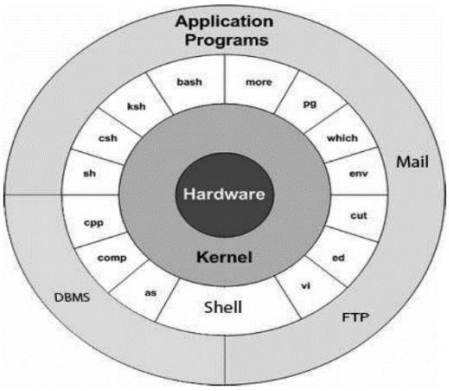

Figure 3: Unix Architecture

Figure 3 shows the architecture of a Unix operating system. It has the following four layers:

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.

Shell: The shell is the utility that processes your requests. When you type in a Unix 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.

Commands and Utilities: There are various Unix commands and utilities which you can make use of in your day-to-day activities. **cp**, **mv**, **cat**, **grep**, etc. are a few examples of commands and utilities. There are over 250 standard Unix commands plus numerous others provided through 3rd party software. All the Unix commands come along with various options. We will study a few commands in this week's and next week's lab sessions.

Files and Directories: All the data in 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*.

To perform any task in Unix, a user need to execute commands in the shell of the Unix. A user typically interacts with a Unix shell using a '**terminal**'. On an Ubuntu-installed computer, you can open a terminal by pressing **Ctrl+Alt+T**. Alternatively, you can also find the terminal in the menu of installed applications. The interface that you see is known as a **shell**. We can write commands on this shell to perform various tasks including writing and executing C programs (we'll see that in subsequent labs). When you open the terminal, you are automatically logged in into the shell and Unix places you in your **"home"** directory. This directory will vary for each user.

Unix uses a hierarchical structure for organizing files and directories. This structure is often referred to as a directory tree. Figure 4 illustrates this.

Root: The node under which all directories and files are stored. A Unix tree has a single root node, the slash character (/).
Home: The directory in which you find yourself when you first opened the terminal is called your home directory.
Path: To access a resource in Unix you need to specify the path of that resource in the command. There are two different paths in Unix: Relative and Absolute. While relative path

refers to the path of a resource from a non-root directory, absolute path refers to the path specified relative to the root directory.

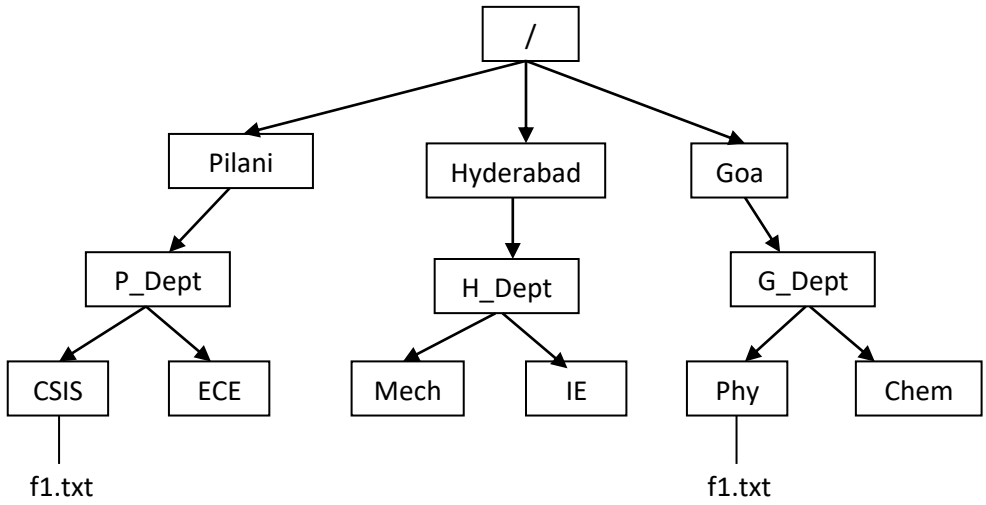

Figure 4: Hierarchical Structure of Unix File System

Consider the above directory structure. At some instance, let's say that you are working in the *Goa* directory and you want to display (using the *cat* command, which will be introduced a little later) the content of a file *f1.txt* stored in the *Phy* directory. You can display the file content by providing either the relative or the absolute path of *f1.txt* to the *cat* command:

Access using absolute path:

```
cat /Goa/G_Dept/Phy/f1.txt
```

here, the first '/' denotes the root directory.

Access using relative path:

```
cat G_Dept/Phy/f1.txt
```

here, you have specified the path of *f1.txt* relative to the *Goa* directory.

Now let us assume that you are in the *Goa* directory and you want to display (using the *cat* command) the content of a file *f1.txt* stored in the *CSIS* directory of the *Pilani* directory.
In this case, you have to access the file *f1.txt* by using its absolute path. This is because to reach CSIS, you have to pass through the root node. So, the command would be

```
cat /Pilani/P_Dept/CSIS/f1.txt
```

An alternate way to use the relative path is through the use of the dot (.) notation. In Unix, the dot denotes the current working directory, while two dots (..) denote the parent of the current working directory. So, if you are working in the *Goa* directory and you want to display the content of a file *f1.txt* stored in the *Phy* directory, then the command using the dot notation can be written as:

```
cat ./G_Dept/Phy/f1.txt
```

Now let us learn a few Unix commands and make our hands dirty. Given below are some Unix commands that we will be using in this course for file and directory management. You should try them out, as well as experiment with them, yourself on your **terminal**! As

mentioned before, you can open a terminal in ubuntu by using the command: **Ctrl+Alt+T**. By default, you are in your home directory.

1. **pwd:** Displays the path of the present working directory

2. **ls:** Lists the content of a directory. Can be executed with various options like –a, -l, etc.
   - ls // displays content of the current directory
   - ls –a // displays the contents of the current directory along with hidden files
   - ls –l // displays the contents of the current directory along with various other details such as the owner of this file, the read/write permissions of the file, the date and time stamp when the file was last updated, etc.
   - ls –al // same as ls –l but now displays details for hidden files as well

3. **man:** Displays the user manual of any command that we can run on the terminal.
   - man ls // displays the manual of ls command along with all its options
   - man cat // displays the manual of cat command
   - etc.

4. **cd:** changes the current directory of the shell
   - cd / // Takes you to the root directory
   - cd .. // Takes you to the parent of the current directory
   - cd ../.. // Takes you to the parent of the parent of the current directory
   - cd - // Takes you to the previous directory you were working in
   - cd ~ // Takes you to your *home* directory

5. **mkdir:** Making a new directory
   - mkdir n_dir //creates a directory named n_dir
   - mkdir –p // create nested directories.
     - e.g. mkdir –p m_dir/hello/world //creates a world directory inside a hello directory which itself is inside m_dir directory

6. **rmdir:** Removing a directory
   - rmdir dir //removes directory dir, provided it is empty
   - rm –r dir //deletes a directory recursively along with its content.

7. **cat:** Displays the content of a file
   - cat file1 // Displays the content of the file1
   - cat > file1 // Opens file1 for writing into it, if file1 already exists. Otherwise, creates file1 and opens it for writing into it.
   - cat file1 file2 //Displays the content of file1 followed by the content of file2

8. **cp:** Copying the content of a file
   - cp file1 file2 // copies the content of file1 in file2. Overwrites file2's content
   - cp $file_1$ $file_2$ …. $file_n$ dest // copies all files ($file_1$ to $file_n$) into the dest directory
   - cp file1 ~ // copies file1 into your home directory
   - cp *.txt dest // copies all files that have their file names with ".txt" as the suffix into dest directory

9. **mv:** Used to move one or more files or directories from one place to another and also for renaming a file or a directory
   - *mv f1.txt f2.txt* // rename f1.txt to f2.txt
   - *mv f1.txt dir/* //move f1.txt to the directory *dir*

- *mv f1.txt dir/new/name/* // move f1.txt to the directory *dir/new/name/*

10. **wc:** Counting words or lines
    - *wc -l file1.txt* //counts the number of lines in file1.txt
    - *wc -w file1.txt* //counts the number of words in file1.txt
    - *wc -c file1.txt* //counts the number of bytes in file1.txt
    - *wc -m file1.txt* //counts the number of characters in file1.txt

11. **head:** print the top N number of data of the given input.
    - *head file1.txt* //Prints the first 10 lines of file1.txt
    - *head –n 5 file1.txt* or *head -5 file1.txt* //Prints the first 5 lines of file1.txt
    - *head -c 2 file1.txt* //Prints the first 2 bytes of file1.txt

12. **tail:** print the last N number of data of the given input.
    - *tail file1.txt* //Prints the last 10 lines of file1.txt
    - *tail –n 5 file1.txt or tail -5 file1.txt* //Prints the last 5 lines of file1.txt
    - *tail -c 2 file1.txt* //Prints the last 2 bytes of file1.txt

13. **echo:** used to display lines of text/string that are passed as an argument
    - *echo* "Hello world" //will display hello world on the terminal

Few other utilities/commands:

- ***date:*** Displays the current date and time, day, month name, day of the month, the time zone name, and the year.
- ***whoami:*** Displays the user name of the current user
- ***who:*** Displays information about all users currently logged in.
- ***clear:*** Clears the screen
- ***cal:*** Displays the calendar of a specific month or a whole year.
- ***hostnamectl:*** Displays OS name and version
- ***uname –r:*** Displays kernel version

Exercise:

Open terminal in your ubuntu. You will be at your home screen. Create the directory structure given in the figure on page 1. Then first create a new file f2.txt in the ECE directory. Store the names of 10 students (any random name you can use) and then copy it's content into f4.txt of the Mech directory. It should be noted that you do not have any f4.txt in the *Mech* directory and you should not create one using the cat command. After copying the content of f2.txt in f4.txt, print the names of the students starting from the 4th line in the file f4.txt, and remove the ECE directory from the tree.

Review Questions:

1. What is the effect of typing cd ~ on your terminal? What is your working directory after executing this command?
2. What is the difference between mkdir and mkdir with –p flag?
3. How does the rm command differ from rmdir command? What is the use of rm –r?
4. Given files f1 and f2, how can you create a new file f3 with the contents of f1 followed by contents of f2? (note: this requires some exploration beyond the contents of this lab sheet)