OPERATING SYSTEM FUNCTIONS

computer System organization:

A modern general purpose computer system consists of one or more cpus and a number of device controllers connected through a common bus that provides access to shared memory

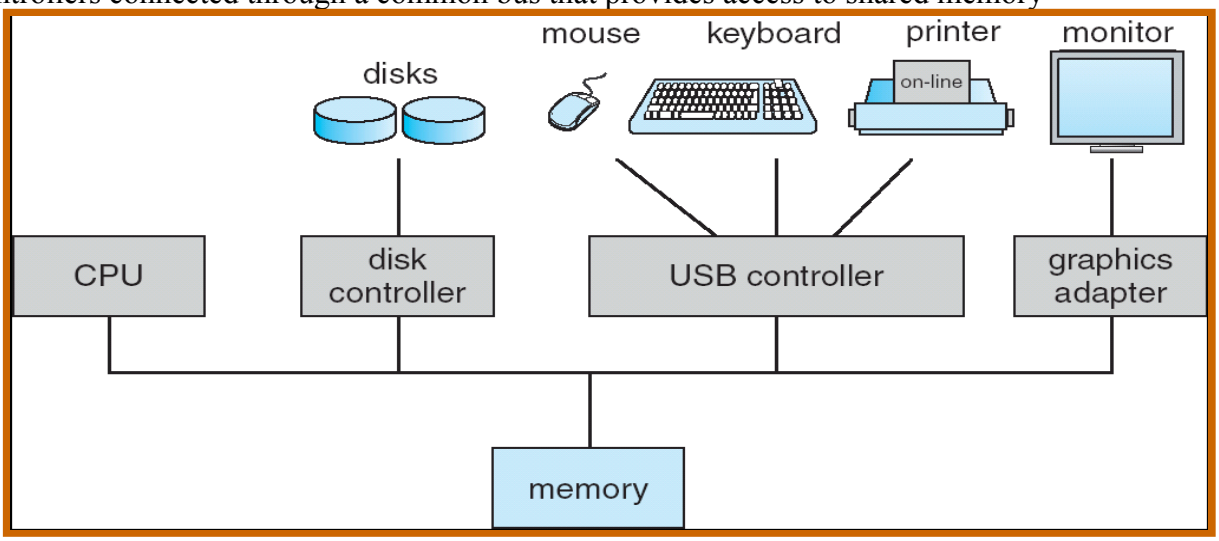
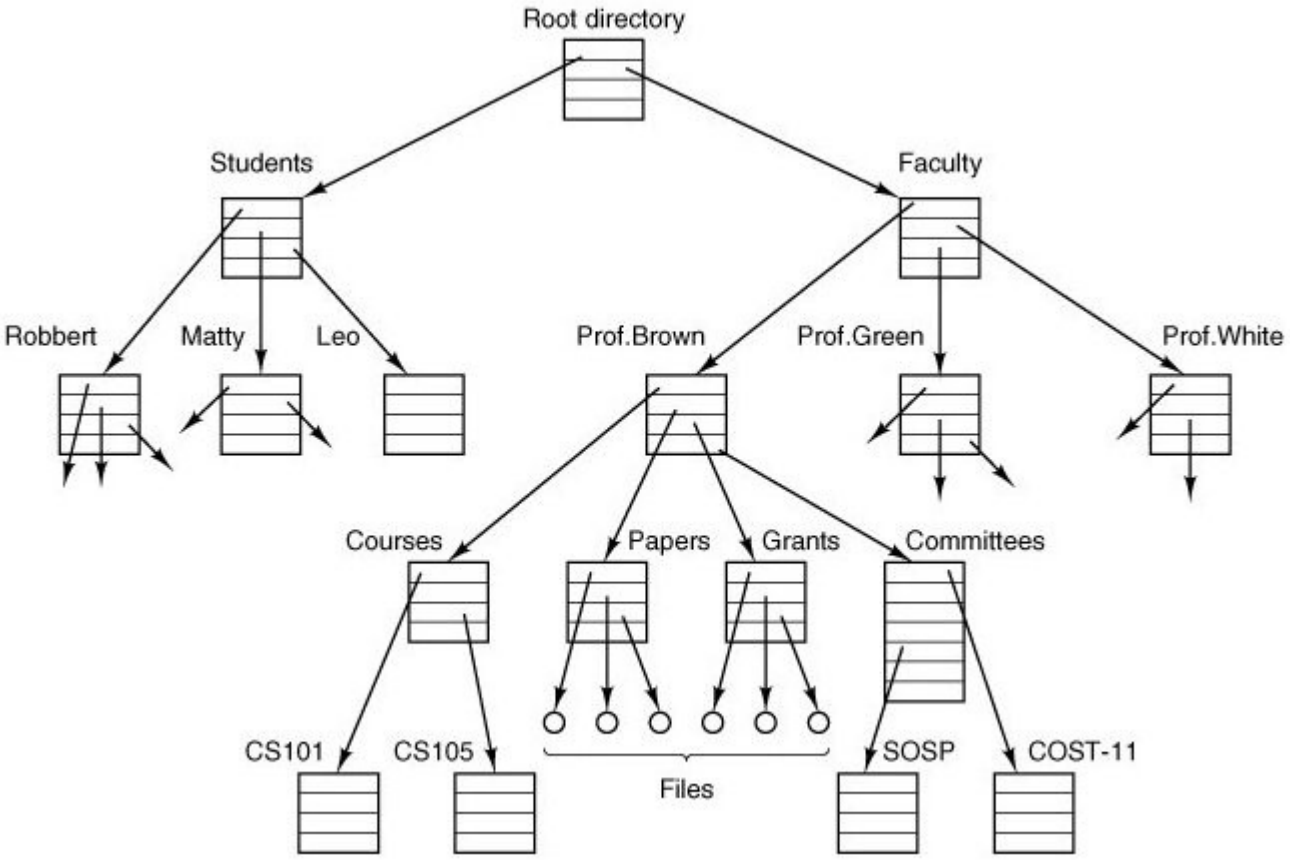


Fig 1.4: A Modern Computer System

Files:

A major function of the operating system is to hide the peculiarities of the disks and other I/O devices and present the programmer with a nice, clean abstract model of device-independent files. System calls are obviously needed to create files, remove files, read files, and write files. Before a file can be read, it must be opened, and after it has been read it should be closed, so calls are provided to do these things.

To provide a place to keep files, most operating system has the concept of a directory as a way of grouping files together. A student, for example, might have one directory for each course he is taking (for the programs needed for that course), another directory for his electronic mail, and still another directory for his World Wide Web home page. System calls are then needed to create and remove directories. Calls are also provided to put an existing file into a directory, and to remove a file from a directory. Directory entries may be either files or other directories. This model also gives rise to a hierarchy of the file system as shown in fig.



Every file within the directory hierarchy can be specified by giving its path name from the top of the directory hierarchy, the root directory. Such absolute path names consist of the list of directories that must be traversed from the root directory to get to the file, with slashes separating the components. In Fig. 1-6, the path for file CS101 is /Faculty/Prof.Brown/Courses/CS101. The leading slash indicates that the path is absolute, that is, starting at the root directory. As an aside, in Windows, the backslash (\) character is used as the separator instead of the slash (/) character, so the file path given above would be written as \Faculty\Prof.Brown\Courses\CS101.

System Call:

In computing, a **system call** is how a program requests a service from an operating system's kernel. This may include hardware related services (e.g. accessing the hard disk), creating and executing new processes, and communicating with integral kernel services (like scheduling). System calls provide the interface between a process and the operating system.
On Unix, Unix-like and other POSIX-compatible operating systems, popular system calls are open, read, write, close, wait, execve, fork, exit, and kill. Many of today's operating systems have hundreds of system calls. For example, Linux has over 300 different calls.

System calls can be roughly grouped into five major categories:

1. Process Control.
    - load
    - execute
    - create process

- terminate process
- get/set process attributes
- wait for time, wait event, signal event
- allocate, free memory

2. File management.
    - create file, delete file
    - open, close
    - read, write, reposition
    - get/set file attributes
3. Device Management.
    - request device, release device
    - read, write, reposition
    - get/set device attributes
    - logically attach or detach devices
4. Information Maintenance.
    - get/set time or date
    - get/set system data
    - get/set process, file, or device attributes
5. Communication.
    - create, delete communication connection
    - send, receive messages
    - transfer status information
    - attach or detach remote devices

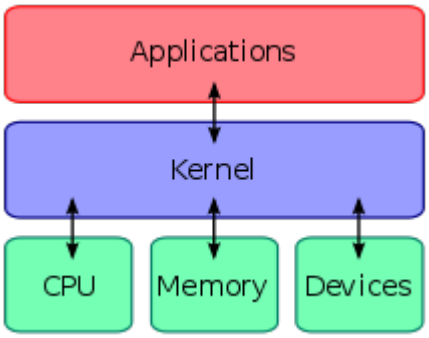
Shell:

A shell is a program that provides the traditional text only user interface for Linux and other Unix operating system. Its primary function is to read commands typed into a console or terminal window and then execute it. The term shell derives its name form the fact that it is an outer layer of the OS. A shell is an interface between the user and the internal part of the operating system.
A user is in shell(i.e interacting with the shell) as soon as the user has logged into the system. A shell is the most fundamental way that user can interact with the system and the shell hides the detail of the underlying system from the user.
Example:
Bourne Shell
Bash shell
Korn Shell
C shell

Kernel:

In computing, the **kernel** is the main component of most computer operating systems; it is a bridge between applications and the actual data processing done at the hardware level. The kernel's responsibilities include managing the system's resources (the communication between hardware and software components). Usually as a basic component of an operating system, a kernel can provide the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application

software must control to perform its function. It typically makes these facilities available to application processes through inter-process communication mechanisms and system calls.
Operating system tasks are done differently by different kernels, depending on their design and implementation. While monolithic kernels execute all the operating system code in the same address space to increase the performance of the system, microkernels run most of the operating system services in user space as servers, aiming to improve maintainability and modularity of the operating system. A range of possibilities exists between these two extremes.



Source : http://dayaramb.files.wordpress.com/2012/02/operating-system-pu.pdf