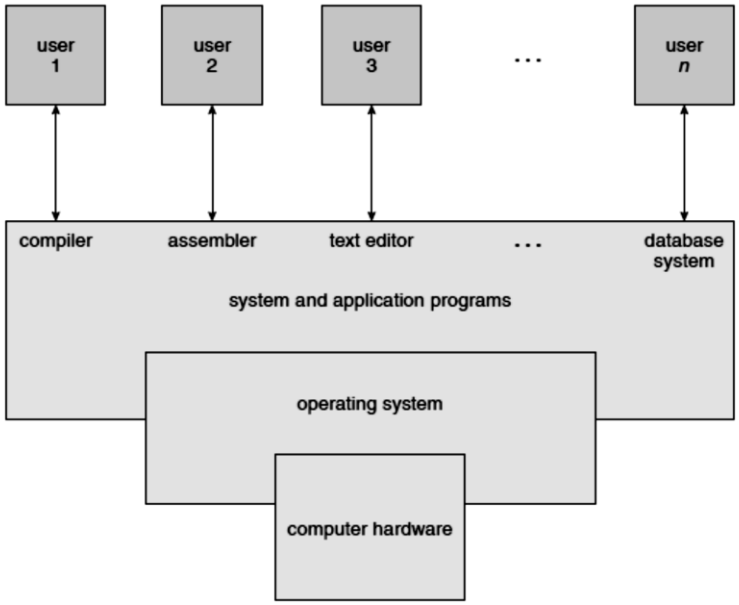
OPERATING SYSTEM

An operating system is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. An amazing aspect of operating systems is how varied they are in accomplishing these tasks. Mainframe operating systems are designed primarily to optimize utilization of hardware. Personal computer (PC) operating systems support complex games, business applications, and everything in between. Operating systems for handheld computers are designed to provide an environment in which a user can easily interface with the computer to execute programs. Thus, some operating systems are designed to be convenient, others to be efficient, and others some combination of the two.

A computer system can be divided roughly into four components- the hardwares, the operating systems, the application programs and the users.



The hardware—the central processing unit(CPU), the memory, and the input output devices—provides the basic computing resources for the system.

The application programs—such as word processors, spreadsheets, compilers, and Web browsers—define the ways in which these resources are used to solve user's computing problems.

The operating system controls the hardware and coordinates its use among the various application programs for the various users.

We can also view a computer system as consisting of hardware, software, and data. The operating system provides the means for proper use of these resources in the operation of the computer system.

An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an environment within which other programs can do useful work.

If we take an example where we have not taken the operating system into consideration then the computer hardware will be directly interacted by system or application programs on which the users give their commands to execute their work, which is quite impossible because system or application softwares requires a platform or we can say an environment to work because they are not meant to directly connect with hardwares.

Our operating system have two view points one from users and another from system.

USER VIEW

- The user's view of the computer varies according to the interface being used. Most computer users sit in front of a PC, consisting of a monitor, keyboard, mouse, and system unit. Such a system is designed for one user to monopolize its resources. The goal is to maximize the work that the user is performing. In this case, the operating system is designed mostly for ease of use, with some attention paid to performance and none paid to resources utilization —how various hardware and software resources are shared. Performance is, of course, important to the user; but such systems are optimized for the single-user experience rather than the requirements of multiple users.
- A user sits at a terminal connected to a mainframe or a minicomputer. Other users are accessing the same computer through other terminals. These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization— to assure that all available CPU time, memory, and I/O are used efficiently and that no individual user takes more than her fair share.
- users sit at workstations connected to networks of other workstations and servers. These users have dedicated resources at their side, but they also share resources such as networking and servers—file, compute, and print servers. Therefore, their operating system is designed to compromise between individual usability and resource utilization.

- many varieties of handheld computers have come into fashion. Most of these devices are standalone units for individual users. Some are connected to networks, either directly by wire or through wireless modems and networking. Because of power, speed, and interface limitations, they perform relatively few remote operations. Their operating systems are designed mostly for individual usability, but performance per unit of battery life is important as well.

SYSTEM VIEW

- From the computer's point of view, the operating system is the program most intimately involved with the hardware. In this context, we can view an operating system as a resource allocater.
  A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on.
  The operating system acts as the manager of these resources. Facing numerous and possibly conflicting requests for resources, the operating system must decide how to allocate them to specific programs and users so that it can operate the computer system efficiently and fairly. As we know, resource allocation is especially important where many users access the same mainframe or minicomputer.

PROCESS MANAGEMENT

A program does nothing unless its instructions are executed by a CPU. A program in execution, as mentioned, is a process. A time-shared user program such as a compiler is a process. A word-processing program being run by an individual user on a PC is a process.
A system task, such as sending output to a printer, can also be a process (or at least part of one).
For now, we can consider a process to be a job or a time-shared program.
It is possible to provide system calls that allow processes to create sub processes to execute concurrently.

A process needs certain resources—including CPU time, memory, files, and I/O devices—to accomplish its task. These resources are either given to the process when it is created or allocated to it while it is running.
In addition to the various physical and logical resources that a process obtains when it is created, various initialization data (input) may be passed along. For example, consider a process whose function is to display the status of a file on the screen of a terminal. The process will be given as an input the name of the file and will execute the appropriate instructions and system calls to obtain and display on the terminal the desired information. When the process terminates, the operating system will reclaim any reusable resources.
A process is the unit of work in a system. Such a system consists of a collection of processes, some of which are operating-system processes (those that execute system code) and the rest of which are user processes (those that execute user code). All these processes can potentially execute concurrently— by multiplexing on a single CPU, for example. The operating system is responsible for the following activities in connection with process management:

- Scheduling processes and threads on the CPUs
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication

MEMORY MANAGEMENT

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes ranging in size from hundreds of thousand to billions. Main memory stores the quickly accessible data shared by the CPU & Input/output device. The central processor reads instruction from main memory during instruction fetch cycle and it both reads and writes data from main memory during the data fetch cycle. The main memory is generally the only large storage device that the CPU is able to address and access directly. For example, for the CPU to process data from disk. Those data must first be transferred to main memory by CPU generated input/output calls. Instruction must be in memory for the CPU to execute them.
The OS is responsible for the following activities in connection with memory management.

- Keeping track of which parts of memory are currently being used & by whom.
- Deciding which processes are to be loaded into memory when memory space becomes available.
- Allocating and deallocating memory space as needed.

FILE SYSTEMS

- A file is a collection of related information defined by its creator.
- In general file is sequence of bits, bytes, lines or records.

  Some examples of file types-
  Text file-
  Sequence of characters organised into lines.
  Source file-
  Sequence of sub routines or functions (which are executable or defined to work on specific task)
  Ex- [a] is source file and has two functions f1() and f2()
  So, when we call the source file “a” then it will automatically call the function f2().
  Object file-
  Collection of words organised into loader record blocks.

Whenever any file is executed it gets loaded or there are some loaders which gets called and there are certain aspects of the functions which needs to be initialized. Generally, those aspects or factors we initialize into this object file.

FILE ATTRIBUTES-

These are the parameters used to keep track of files in operating system.
A file's attributes vary from one operating system to another but typically consist of these:

- **Name**.- The symbolic file name is the only information kept in human readable form.
- **Identifier**.- This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.
- **Type**. -This information is needed for systems that support different types of files.
- **Location**.- This information is a pointer to a device and to the location of the file on that device.
- **Size**. -The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
- **Protection**.- Access-control information determines who can do reading, writing, executing, and so on.
- **Time and date-** record of creation, modification and deletion.
- **User ID-** user identification (used to give a unique ID to the creator of the file).

FILE OPERATIONS-

Creating a file- two steps are necessary to create a file.
Step 1- to find space for new file
Step 2- now make entry for new file
Writing a file- for writing a file we need to make system call both by specifying the name of the file and by writing information to the file.
The system must keep a *write* pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
Reading a file- system call by specifying the name of the file and where the next block of the file should be put in the memory.
Repositioning a file- repositioning a file doesn't involve any actual input and output operation. This operation is also known as file seek.
So, for searching a file we use repositioning (file seek).
Deleting a file- To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
Truncating a file- The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released.

FILE TYPES-

When we design a file system—indeed, an entire operating system—we always consider whether the operating system should recognize and support file types. If an operating system recognizes the type of a file, it can then operate on the file in reasonable ways.
For example, a common mistake occurs when a user tries to print the binary-object form of a program.

This attempt normally produces garbage; however, the attempt can succeed if the operating system has been told that the file is a binary-object program. A common technique for implementing file types is to include the type as part of the file name.
The name is split into two parts—a name and an extension, usually separated by a period character. In this way, the user and the operating system can tell from the name alone what the type of a file is.
For example, most operating systems allow users to specify a file name as a sequence of characters followed by a period and terminated by an extension of additional characters. File name examples include resume.doc, Server.java, and ReaderThread.c.

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information