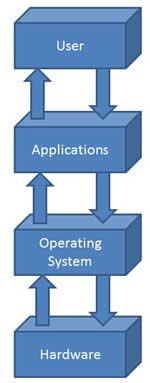
An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Operating System

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

**The Operating System is a program with the following features −**

* An operating system is a program that acts as an interface between the software and the computer hardware.
* It is an integrated set of specialized programs used to manage overall resources and operations of the computer.
* It is specialized software that controls and monitors the execution of all other programs that reside in the computer, including application programs and other system software.



**The objectives of the operating system are −**

* To make the computer system convenient to use in an efficient manner.
* To hide the details of the hardware resources from the users.
* To provide users a convenient interface to use the computer system.
* To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
* To manage the resources of a computer system.
* To keep track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
* To provide efficient and fair sharing of resources among users and programs.

**Two views of Operating systems**

**User View**

* The goal of the Operating System is to maximize the work and minimize the effort of the user.
* Most of the systems are designed to be operated by single user; however in some systems multiple users can share resources, memory. In these cases Operating System is designed to handle available resources among multiple users and CPU efficiently.
* Operating System must be designed by taking both usability and efficient resource utilization into view.
* In embedded systems (Automated systems) user view is not present.
* Operating System gives an effect to the user as if the processor is dealing only with the current task, but in background processor is dealing with several processes.

**System View**

* From the system point of view Operating System is a program involved with the hardware.
* Operating System is allocator, which allocate memory, resources among various processes. It controls the sharing of resources among programs.
* It prevents improper usage, error and handles deadlock conditions.
* It is a program that runs all the time in the system in the form of Kernel.
* It controls application programs that are not part of Kernel.

**Following are some of important functions of an operating System.**

* Memory Management
* Processor Management
* Device Management
* File Management
* Security
* Control over system performance
* Job accounting
* Error detecting aids
* Coordination between other software and users

**Memory Management**

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management:

* Keeps tracks of primary memory, i.e., what part of it are in use by whom, what parts are not in use.
* In multiprogramming, the OS decides which process will get memory when and how much.
* Allocates the memory when a process requests it to do so.
* De-allocates the memory when a process no longer needs it or has been terminated.

**Processor Management**

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called **process scheduling**. An Operating System does the following activities for processor management:

* Keeps tracks of processor and status of process. The program responsible for this task is known as **traffic controller**.
* Allocates the processor (CPU) to a process.
* De-allocates processor when a process is no longer required.

**Device Management**

An Operating System manages device communication via their respective drivers. It does the following activities for device management:

* Keeps tracks of all devices. The program responsible for this task is known as the I/O controller.
* Decides which process gets the device when and for how much time.
* Allocates the device in the most efficient way.
* De-allocates devices.

**File Management**

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An Operating System does the following activities for file management:

* Keeps track of information, location, uses, status etc. The collective facilities are often known as **file system**.
* Decides who gets the resources.
* Allocates the resources.
* De-allocates the resources.

**Other Important Activities**

Following are some of the important activities that an Operating System performs:

* **Security** -- By means of password and similar other techniques, it prevents unauthorized access to programs and data.
* **Control over system performance** -- Recording delays between request for a service and response from the system.
* **Job accounting** -- Keeping track of time and resources used by various jobs and users.
* **Error detecting aids** -- Production of dumps, traces, error messages, and other debugging and error detecting aids.
* **Coordination between other software and users** -- Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

**Types of Operating System**

**Batch Operating System**

The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows:

* Lack of interaction between the user and the job.
* CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
* Difficult to provide the desired priority.

**Time-sharing Operating Systems**

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

The main difference between Multyprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, the processor executes each user program in a short burst or quantum of computation. That is, if n users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most.

The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

**Advantages of Time-sharing operating systems are as follows:**

* Provides the advantage of quick response
* Avoids duplication of software
* Reduces CPU idle time

**Disadvantages of Time-sharing operating systems are as follows:**

* Problem of reliability
* Question of security and integrity of user programs and data
* Problem of data communication

**Distributed Operating System**

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as **loosely coupled systems** or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

The advantages of distributed systems are as follows:

* With resource sharing facility, a user at one site may be able to use the resources available at another.
* Speedup the exchange of data with one another via electronic mail.
* If one site fails in a distributed system, the remaining sites can potentially continue operating.
* Better service to the customers.
* Reduction of the load on the host computer.
* Reduction of delays in data processing.

**Network Operating System**

A Network Operating System runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.

Examples of network operating systems include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows:

* Centralized servers are highly stable.
* Security is server managed.
* Upgrades to new technologies and hardware can be easily integrated into the system.
* Remote access to servers is possible from different locations and types of systems. The disadvantages of network operating systems are as follows:
* High cost of buying and running a server.
* Dependency on a central location for most operations.
* Regular maintenance and updates are required.

**Real-Time Operating System**

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the **response time**. So in this method, the response time is very less as compared to online processing.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

**Hard real-time systems**

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

**Soft real-time systems**

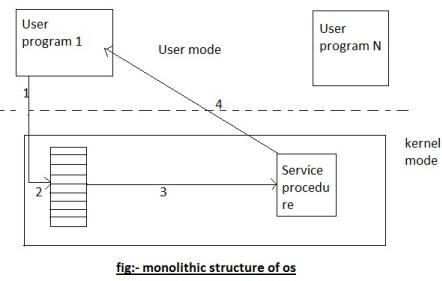
Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

**Operating system structure**:

An operating system might have many structures. According to the structure of the operating system; operating systems can be classified into many categories.

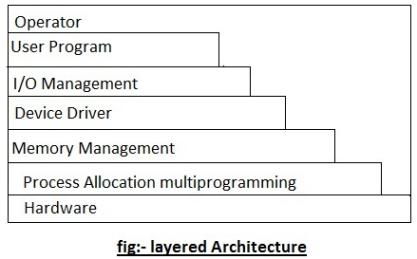
Some of the main structures used in operating systems are:

1. **Monolithic architecture of operating system**

  
It is the oldest architecture used for developing operating system. Operating system resides on kernel for anyone to execute. System call is involved i.e.  Switching from user mode to kernel mode and transfer control to operating system shown as event 1. Many CPU has two modes, kernel mode, for the operating system in which all instruction are allowed and user mode for user program in which I/O devices and certain other instruction are not allowed.  Two operating system then examines the parameter of the call to determine which system call is to be carried out shown in event 2. Next, the operating system index’s into a table that contains procedure that carries out system call. This operation is shown in events. Finally, it is called when the work has been completed and the system call is finished, control is given back to the user mode as shown in event 4.

2. **Layered Architecture of operating system**

The layered Architecture of operating system was developed in 60’s in this approach; the operating system is broken up into number of layers. The bottom layer (layer 0) is the hardware layer and the highest layer (layer n) is the user interface layer as shown in the figure.

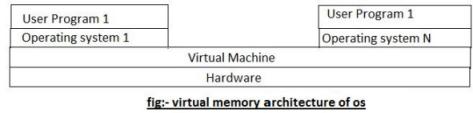


The layered are selected such that each user functions and services of only lower level layer. The first layer can be debugged without any concern for the rest of the system. It user basic hardware to implement this function once the first layer is debugged., it’s correct functioning can be assumed  while the second layer is debugged & soon . If an error is found during the debugged of particular layer, the layer must be on that layer, because the layer below it already debugged. Because of this design of the system is simplified when operating system is broken up into layer.

Os/2 operating system is example of layered architecture of operating system another example is earlier version of Windows NT.

The main disadvantage of this architecture is that it requires an appropriate definition of the various layers & a careful planning of the proper placement of the layer.

3. **Virtual memory architecture of operating system**



Virtual machine is an illusion of a real machine. It is created by a real machine operating system, which make a single real machine appears to be several real machine. The architecture of virtual machine is shown above.

The best example of virtual machine architecture is IBM 370 computer. In this system each user can choose a different operating system. Actually, virtual machine can run several operating systems at once, each of them on its virtual machine.

Its multiprogramming shares the resource of a single machine in different manner.  
The concepts of virtual machine are:-

a. **Control program (cp):-** cp creates the environment in which virtual machine can execute. It gives to each user facilities of real machine such as processor, storage I/0 devices.

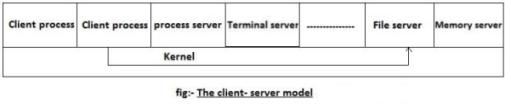
b. **conversation monitor system (cons):-** cons is a system application having features of developing program. It contains editor, language translator, and various application packages.

c.    **Remote spooling communication system (RSCS):-** provide virtual machine with the ability to transmit and receive file in distributed system.

d. IPCS (interactive problem control system):- it is used to fix the virtual machine software problems.

**4. client/server architecture of operating system**

A trend in modern operating system is to move maximum code into the higher level and remove as much as possible from operating system, minimizing the work of the kernel. The basic approach is to implement most of the operating system functions in user processes to request a service, such as request to read a particular file, user send a request to the server process, server checks the parameter and finds whether it is valid or not, after that server does the work and send back the answer  to client server model works on request- response technique i.e. Client  always send request to the side in order to perform the task, and on the other side, server gates complementing that request send back response. The figure below shows client server architecture.



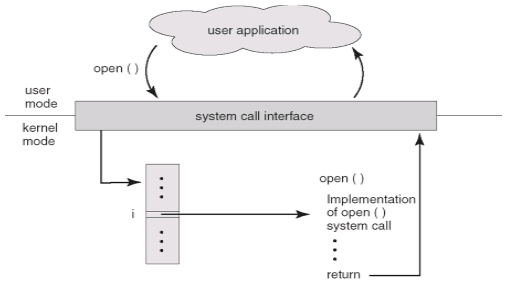
In this model, the main task of the kernel is to handle all the communication between the client and the server by splitting the operating system into number of ports, each of which only handle some specific task. I.e. file server, process server, terminal server and memory service.

Another advantage of the client-server model is it’s adaptability to user in distributed system. If the client communicates with the server by sending it the message, the client need not know whether it was send a ……. Is the network to a server on a remote machine? As in case of client, same thing happen and occurs in client side that is a request was send and a reply come back.

**System Call**

The System Call is the Request for running any Program and for performing any Operation on the System. When a user First Time Starts the System then the **System is in the user Mode and When he request For a Service then the User Mode will be Converted into the Kernel Mode** Which just Listen the Request of the user and Process the Request and Display the Results those are Produced after the Processing. When a user Request for Opening any Folder or When a Moves his Mouse his Mouse on the Screen, then this is called as the System call which he is using for performing any Operation.

* System calls provide an interface between the process and the operating system.
* System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do.
* In handling the trap, the operating system will enter in the kernel mode, where it has access to privileged instructions, and can perform the desired service on the behalf of user-level process.
* It is because of the critical nature of operations that the operating system itself does them every time they are needed.
* For example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system.



**Types of System calls**

* Process control
* File management
* Device management
* Information maintenance
* Communications

**1) Process Control:**

A running program needs to be able to stop execution either normally or abnormally.

When execution is stopped abnormally, often a dump of memory is taken and can be examined with a debugger.

Following are functions of process control:

* end, abort
* load, execute
* create process, terminate process
* get process attributes, set process attributes
* wait for time
* wait event, signal event
* allocate and free memory

**2) File management :**

We first need to be able to create and delete files. Either system call requires the name of the file and perhaps some of the file's attributes.

Once the file is created, we need to open it and to use it. We may also read, write, or reposition. Finally, we need to close the file, indicating that we are no longer using it.

We may need these same sets of operations for directories if we have a directory structure for organizing files in the file system.

In addition, for either files or directories, we need to be able to determine the values of various attributes and perhaps to reset them if necessary. File attributes include the file name, a file type, protection codes, accounting information, and so on

**Functions:**

* create file, delete file
* open, close file
* read, write, reposition
* get and set file attributes

**3) Device Management:**

* A process may need several resources to execute - main memory, disk drives, access to files, and so on. If the resources are available, they can be granted, and control can be returned to the user process. Otherwise, the process will have to wait until sufficient resources are available.
* The various resources controlled by the OS can be thought of as devices. Some of these devices are physical devices (for example, tapes), while others can be thought of as abstract or virtual devices (for example, files).
* Once the device has been requested (and allocated to us), we can read, write, and (possibly) reposition the device, just as we can with files.
* In fact, the similarity between I/O devices and files is so great that many OSs, including UNIX, merge the two into a combined file-device structure.
* A set of system calls is used on files and devices. Sometimes, 1/0 devices are identified by special file names, directory placement, or file attributes.

**Functions:**

* request device, release device
* read, write, reposition
* get device attributes, set device attributes
* logically attach or detach devices

**Information Maintenance**

* Many system calls exist simply for the purpose of transferring information between the user program and the OS. For example, most systems have a system call to return the current time and date.
* Other system calls may return information about the system, such as the number of current users, the version number of the OS, the amount of free memory or disk space, and so on.
* In addition, the OS keeps information about all its processes, and system calls are used to access this information. Generally, calls are also used to reset the process information.

**Functions:**

* get time or date, set time or date
* get system data, set system data
* get and set process, file, or device attributes

**Communication**

* There are two common models of interprocess communication: the message-passing model and the shared-memory model. In the message-passing model, the communicating processes exchange messages with one another to transfer information.
* In the shared-memory model, processes use shared memory creates and shared memory attaches system calls to create and gain access to regions of memory owned by other processes.
* Recall that, normally, the OS tries to prevent one process from accessing another process's memory. Shared memory requires that two or more processes agree to remove this restriction. They can then exchange information by reading and writing data in the shared areas.
* Message passing is useful for exchanging smaller amounts of data, because no conflicts need be avoided. It is also easier to implement than is shared memory for intercomputer communication.
* Shared memory allows maximum speed and convenience of communication, since it can be done at memory speeds when it takes place within a computer. Problems exist, however, in the areas of protection and synchronization between the processes sharing memory.

**Functions:**

* create, delete communication connection
* send, receive messages
* transfer status information
* Attach and Detach remote devices

**shell**

In [computing](https://en.wikipedia.org/wiki/Computing" \o "Computing), a **shell** is a [user interface](https://en.wikipedia.org/wiki/User_interface" \o "User interface) for access to an [operating system](https://en.wikipedia.org/wiki/Operating_system" \o "Operating system)'s services. In general, operating system shells use either a [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface" \o "Command-line interface) (CLI) or [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface" \o "Graphical user interface) (GUI), depending on a computer's role and particular operation. It is named a shell because it is the outermost layer around the operating system [kernel](https://en.wikipedia.org/wiki/Kernel_(operating_system)" \o "Kernel (operating system)).

CLI shells allow some operations to be performed faster, especially when a proper GUI has not been or cannot be created; however, they require the user to be familiar with commands and their calling [syntax](https://en.wikipedia.org/wiki/Syntax" \o "Syntax), and to understand concepts about the shell-specific scripting language (for example [bash script](https://en.wikipedia.org/wiki/Bash_script" \o "Bash script)), which may prove difficult for those with little computer experience. CLIs are also easier to be operated via [refreshable braille display](https://en.wikipedia.org/wiki/Refreshable_braille_display" \o "Refreshable braille display) and provide certain advantages to [screen readers](https://en.wikipedia.org/wiki/Screen_reader" \o "Screen reader).

Graphical shells place a low burden on beginning computer users, and they are characterized as being simple and easy to use. With the widespread adoption of programs with GUIs, the use of graphical shells has gained greater adoption. Since graphical shells come with certain disadvantages, most GUI-enabled operating systems also provide additional CLI shells.

* A **shell** is a command interpreter, i.e. the program that either process the command you enter in your terminal emulator (interactive mode) or process shell scripts (text files containing commands) (batch mode). In early UNIX times, it used to be the unique way for users to interact with their machines. Nowadays, graphical environments are replacing the shell for most casual users.
* A **kernel** is a low level program interfacing with the hardware (CPU, RAM, disks, network ...) on top of which applications are running. It is the lowest level program running on computers although with virtualization you can have multiple kernels running on top of virtual machines which themselves run on top of another operating system.

**Open Source Software**

In general, open source refers to any program whose source code is made available for use or modification as users or other developers see fit. Open source software is usually developed as a public collaboration and made freely available.

Open-source software (OSS) is computer software distributed with its source code available for modification. The software usually includes a license for programmers to change the software in any way they choose. They can fix bugs, improve functions, or adapt the software to suit their own needs. [The Open Source Initiative](https://opensource.org/" \t "_blank) (OSI) is a leading authority on OSS; their definition of open-source software complies with rules under 10 criteria.

These include:

* Software redistribution
* Source code availability
* Distribution of licenses
* License properties
* Anti-discrimination

Examples of computer open-source operating systems include Linux, FreeBSD and OpenSolaris. Closed-source operating systems include Microsoft Windows, Solaris Unix and OS X. Older closed-source operating systems include OS/2, BeOS and the original Mac OS, which was replaced by OS X. On mobile and tablet systems, closed-source operating systems include Windows Phone, iOS and the Symbian OS that is used by BlackBerry. Android is based on the open-source Linux OS, though it has many proprietary, closed-source extensions. The Linux-based Firefox OS is an example of a fully open-source mobile OS, though it has yet to gain significant popularity.

**SERVICES**

An Operating System provides services to both the users and to the programs.

It provides programs an environment to execute.

It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system −

* Program execution
* I/O operations
* File System manipulation
* Communication
* Error Detection
* Resource Allocation
* Protection

**Program execution**

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management −

* Loads a program into memory.
* Executes the program.
* Handles program's execution.
* Provides a mechanism for process synchronization.
* Provides a mechanism for process communication.
* Provides a mechanism for deadlock handling.

**I/O Operation**

An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

* An Operating System manages the communication between user and device drivers.
* I/O operation means read or write operation with any file or any specific I/O device.
* Operating system provides the access to the required I/O device when required.

**File system manipulation**

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

* Program needs to read a file or write a file.
* The operating system gives the permission to the program for operation on file.
* Permission varies from read-only, read-write, denied and so on.
* Operating System provides an interface to the user to create/delete files.
* Operating System provides an interface to the user to create/delete directories.
* Operating System provides an interface to create the backup of file system.

**Communication**

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

* Two processes often require data to be transferred between them
* Both the processes can be on one computer or on different computers, but are connected through a computer network.
* Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

**Error handling**

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

* The OS constantly checks for possible errors.
* The OS takes an appropriate action to ensure correct and consistent computing.
* Resource Management
* In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management −
* The OS manages all kinds of resources using schedulers.
* CPU scheduling algorithms are used for better utilization of CPU.

**Protection**

Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection −

* The OS ensures that all access to system resources is controlled.
* The OS ensures that external I/O devices are protected from invalid access attempts.
* The OS provides authentication features for each user by means of passwords.