|  |
| --- |
| IQRA NATIONAL UNIVERSITY |
| OPERATING SYSTEM |
| TYPES OF OPERATING SYSTEM |
|  |
|  |
|  |

|  |
| --- |
|  |

Contents

[Operating System 2](#_Toc442798489)

[Main functions of an operating system 2](#_Toc442798490)

[Process management 4](#_Toc442798491)

[Multiprogramming 5](#_Toc442798492)

[Multithreading 6](#_Toc442798493)

[Multiprocessing 6](#_Toc442798494)

# Operating System

Operating system is an integrated set of programs that controls the resources ( CPU, memory, I/O devices, e.t.c ) of a computer system and provide its users with an interface that is easier to use than the bare machine. According to this definition, the two primary objectives of an operation system are:

1. **Make a computer system easier to use**. A computer consists of one or more processors, main memory and many types of I/O devices such as disks, tapes, terminals, network interfaces, etc. writing programs for using these hardware resources correctly and efficiently is an extremely difficult job, requiring in-depth knowledge of functioning of these resources. Hence to make a computer system usable by a large number of users, it became clear several years ago that a computer systems need some mechanism to shield programmers and other users from the complexity of hardware resources.

An operating system hides details of hardware resources from programmers and other users. It provides a high-level interface to low-level hardware resources, making it easier for programmers and other users to use a computer system.

1. **Manage the resources of a computer system**. An operating system manages all the resources of a computer system. This involves performing such tasks as keeping track of who is using what resources, granting resource requests, accounting for resource usage, and mediating conflicting requests from different programs and users. Efficient and fair sharing of system resources among users and/or programs is a key goal of all operating systems.

# Main functions of an operating system

Most operating systems perform the functions given below. A separate module of operating system software performs each of these functions:

1. **Process management.** A process is a program in execution. During execution, a process needs certain resources such as CPU time, memory space, files, and I/O devices. At a particular instance of time, a computer system normally consists of a collection of processes. Process management module takes care of creation and deletion of processes, scheduling of system resources to different processes requesting them, and providing mechanisms for synchronization and communication among processes.
2. **Memory management.** To execute a program, it must be loaded in main memory ( at least partially), together with the data it accesses. To improve CPU utilization and to provide better response time to its users, a computer system normally keeps several programs in main memory. Memory management module takes care of allocation and de-allocation of memory space to programs in need of this resource.
3. **File management.** All computer systems store, retrieve and share information. Normally, a computer stores such information in units called files. Processes read information from files and create new files for storing newly generated information. File management module takes care of file-related activities such as organizations, storage, retrieval, naming, sharing and protection of files.
4. **Device management.** Normally, a computer system consists of several I/O devices such as printer, disk and tape. The device management module of an operating system controls all I/O devices. It keeps track of I/O requests from processes, issues commands to I/O devices, and ensures correct data transmission to/ from an I/O device. It also provides a simple and easy to use interface between the devices and rest of the system.
5. **Security.** Computer systems often store large amount of information, some of which are highly sensitive and valuable to their users. Users can trust a computer system and rely on it only if its various resources and information stored in it are protected against destruction and unauthorized access. Security module protects the resources and information of a computer system against destruction and unauthorized access. It also ensures that when the system executes several disjoint processes simultaneously, one process does not interfere with others or with the operating system itself.
6. **Command interpretation.** For using various system resources, a user communicates with the operating system via a set of commands provided by it. The operating system also provides a simple language know as command language or job control language (JCL). Using which a user can put several commands together from the command set to describe the resource requirements of a job.

**Measuring System Performance**

The efficiency and performance of an operation system is usually measured by the following parameters:

**Throughput**:

Throughput is the amount of work that a system is able to do per unit time. We measure it as the number of jobs completed by the system per unit time. Throughput is usually measured in processes/ hour. It should be noted that throughput of a system does not depend on its job processing efficiency only, but also on the nature of jobs processed. For long processes, throughput of a system may be one process/hour; whereas for short processes it may be 100 processes/ hour for the same system.

**Turnaround time**

Turnaround time is the interval between the time of submission of a job to the system for processing and the time of completion of the job. Although, higher throughput is desirable from the point of view of overall system performance, individual users are more interested in better turnaround time for their jobs.

**Response time**

Response time is the interval between the time of submission of a job to the system for processing and the time of first response from the system for the job. In any computer system it is desirable to maximize throughput and minimize turnaround time and response time.

# Process management

In early computer systems, the process of executing a job was as follows:

1. A programmer first wrote a program on paper.
2. The programmer or a data entry operator then punched the program and its data on cards or paper tape.
3. The programmer then submitted the deck of cards or the paper tape containing the program and data at the reception counter of a computer center.
4. An operator then loaded the cards deck or paper tape manually into the system from card reader or paper tape reader. The operator also loaded any other software resource (such as language compiler), or carried out required settings of hardware devices for execution of job. Before loading the job, the operator used front panel switches of the system to clear any data remaining in main memory from previous job.
5. The operator then carried out required settings of the appropriate switches in the front panel to run the job.
6. Finally, the operator printed and submitted the result of execution of the job at the reception counter for the programmer to collect it later.

Every job went through the same process. The method was known as manual loading mechanism because the operator loaded the jobs one after another in the system manually. Notice that in this method, job-to-job transition was not automatic. Hence, a computer remained idle while an operator loaded and unloaded jobs and prepared the system for a new job. This caused enormous wastage of valuable computer time. To reduce this idle time, researchers later devised a mechanism (called batch processing) for automatic job-to-job transition. Batch processing ensured that after completing execution of a job, the system transferred control back to the operating system automatically. The operating system then performed housekeeping jobs (such as clearing any data remaining in memory from previous job), which were needed to load and run the next job. Hence, in a batch processing system, the process of executing jobs was as follows:

1. Programmers prepared their programs and data on decks of cards or paper tapes and submitted them at the reception counter of a computer center.
2. An operator collected all the submitted programs periodically, batched them together and then loaded them all at one time into input device of the system.
3. The operator then gave a command to the system to start executing the batched jobs.
4. The system then loaded the jobs automatically from the input device and executed them one-by-one without any operator intervention.
5. After the system completed execution of all jobs in the submitted batch, the operator separated and kept the printed output of each job at the reception counter of programmers to collect them later.

# Multiprogramming

Both manual loading and batch processing systems load and process one job at a time. After such a system loads a job, the job remains in main memory until its execution is over and the system loads the next job only after completion of the current job. The following figure shows that in such a situation the job that is currently loaded in the system is the sole occupant of user’s area of main memory and has CPU available exclusively for its execution.



Figure shows a uniprogramming system, which processes only one job at a time, and all system resources are available exclusively for the job until its completion.

 A job does not need CPU for entire duration of its processing because in addition to computing ( for which CPU is needed), it often needs to perform I/O operations ( such as reading/ writing of data from/ to tape/disk, waiting for data input from keyboard and printing of results) during the course of its processing. In fact, depending on CPU utilization during the course of processing jobs are of two types:

1. CPU-bound jobs. These jobs need little Input/output operations.
2. I/O bound jobs. Mostly perform input/output operations.

Multiprogramming is interleaved execution of two or more different and independent programs by a computer. In multiprogramming two or more user programs reside simultaneously in main memory and carry out their interleaved execution. With multiple user programs residing simultaneously in the user program area of main memory, whenever a user program that was executing goes to perform I/O operations, the operating system allocates CPU to another user program in main memory that is ready to use CPU, instead of allowing CPU to remain idle. CPU switches from one program to another almost instantaneously. Hence, in multiprogramming several user programs share CPU time to keep it busy.



The figure shows a typical scenario of jobs in a multiprogramming system. At the time instance shown, there are three user jobs residing in memory. Out of which job A is performing I/O operation, job B is executing and job C is waiting for CPU to become free.

# Multithreading

Threads are a popular way to improve application performance. In traditional operating systems, the basic unit of CPU utilization is a process. Each process has its own program counter, its own register states, its own stack and its own address space. On the other hand, in operating systems with threads facility, the basic unit of CPU utilization is a thread. In these operating systems, a process consists of an address space and one or more threads of control. Each thread of a process has its own program counter, its own register states, and its own stack. However, all the threads of a process share the same address space. Hence, they also share the same global variables. In addition, all threads of a process also share the same set of operating system resources, such as open files, signals, accounting information etc. due to sharing of address space; there is no protection between the threads of a process. However, protection between multiple threads of a process is not necessary because a single user always owns a process. Multiple processes need protection against each other because different processes may belong to different users.

# Multiprocessing

Up to this point, we have considered uniprocessor systems. However, we have already seen that the use of I/O processors improves the efficiency of a computer system by making concurrent input, processing, and output operations possible. CPU performs arithmetic and logical operations, while I/O processors carry out I/O operations concurrently.

 Designer carried further the idea of using I/O processors to improve system performance by designing systems with multiple CPUs. Such systems are called multiprocessing systems because they use multiple processors and can execute multiple processes concurrently. These systems use multiple CPUs to process either instructions from different and independent programs or different instructions from the same program simultaneously. The following figure shows basic organization of a typical multiprocessing system.



Multiprocessing systems are of two types

* Tightly-coupled
* Loosely-coupled

In tightly-coupled systems, all processors share a single system-wide primary memory. On the other hand , in loosely-coupled systems, the processors do not share memory and each processor has its own local memory.