Operating System

Unit 1

What is OS?

It is a collection of software that manages hardware resources and provides common services for programs.

When we start using a Computer System, it is the OS which acts as an interface between the hardware and us.

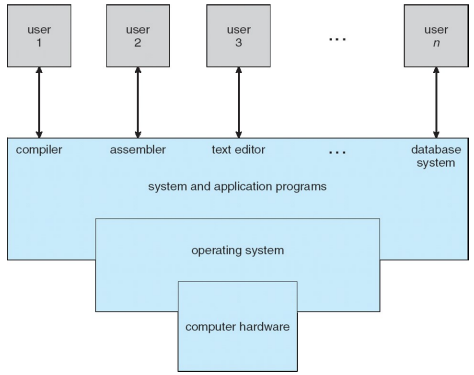
It is a low level **Software** which is categorized as a **System Software** and supports a computer's basic functions, such as memory management, tasks scheduling, controlling peripherals, etc.

An OS is an interface between a computer user and hardware. It is a 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.

Computer System

- **Computer Users** are the users who use the overall computer system.
- **Application Software** are the software which users use directly to perform different activities. These are simple and easy to use like Browsers, Word, Excel, different Editors, Games etc. These are usually written in high-level languages, such as Python, Java, and C++.
- **System Software** are the software which are more complex in nature and they are more near to hardware. These are written in low-level languages like assembly language and includes **OSs** (Microsoft Windows, macOS, and Linux), Compiler, Assembler, etc.
- **Computer Hardware** includes Monitor, Keyboard, CPU, Disks, Memory, etc.

Four Components of a Computer System



The Functions of OS

- Process Management
- I/O Device Management
- File Management
- Network Management
- Main Memory Management
- Secondary Storage Management
- Security Management
- Command Interpreter System
- Control over system performance
- Job Accounting
- Error Detection and Correction
- Coordination between other software and users

Memory Management

- It refers to management of Primary Memory or Main Memory. MM is a large array of words or bytes where each word or byte has its own address.
- MM provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the MM. An OS does the following activities for memory management −
  - Keeps tracks of MM, i.e., what part of it are in use by whom, what part 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 OS 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 to a process.
  - De-allocates processor when a process is no longer required.

Device Management

- An OS manages device communication via their respective drivers. It does the following activities for device management −
  - Keeps tracks of all devices. 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 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 OS 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

- **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.

Batch OS

- The users of a BOS do not interact with the computer directly. Each user prepares his job on an off-line device like punched 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 BOS are:

- 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.

Characteristics

- the CPU executes the jobs in the same sequence that they are sent to it by the operator, which implies that the task sent to the CPU first will be executed first. It's known as the **'first come, first serve'**
- The word job refers to the command or instruction that the user and the program should perform.
- A BOS runs a set of user-supplied instructions composed of distinct instructions and programs with several similarities.
- When a task is successfully executed, the OS releases the memory space held by that job.
- The user does not interface directly with the OS in a BOS; rather, all instructions are sent to the operator.
- The operator evaluates the user's instructions and creates a set of instructions having similar properties.

Advantages

- It isn't easy to forecast how long it will take to complete a job; only batch system processors know how long it will take to finish the job in line.
- This system can easily manage large jobs again and again.
- The batch process can be divided into several stages to increase processing speed.
- When a process is finished, the next job from the job spool is run without any user interaction.
- CPU utilization gets improved.

Disadvantages

- When a job fails once, it must be scheduled to be completed, and it may take a long time to complete the task.
- Computer operators must have full knowledge of batch systems.
- The batch system is quite difficult to debug.
- The computer system and the user have no direct interaction.
- If a job enters an infinite loop, other jobs must wait for an unknown period of time.

Multiprogramming OS

- A MOS may run many programs on a single processor computer. If one program must wait for an I/O transfer in a MOS, the other programs are ready to use the CPU. As a result, various jobs may share CPU time. The execution of their jobs is not defined to be at the same time period.
- When a program is being performed, it is a **"Task", "Process"**, and **"Job"**. Concurrent program executions improve system resource consumption and throughput as compared to serial and batch OS.
- The goal of multiprogramming is to manage the entire system's resources. The components of a MOS are the file system, command processor, transient area, and I/O control system. As a result, MOSs are designed to store different programs based on sub-segmenting parts of the transient area. The resource management routines are linked with the OS core functions.

Two types of MOS

- A **multitasking** OS enables the execution of two or more programs at the same time. The OS accomplishes this by shifting each program into and out of memory one at a time. When a program is switched out of memory, it is temporarily saved on disk until it is required again.
- A **multiuser** OS allows many users to share processing time on a powerful central computer from different terminals. The OS accomplishes this by rapidly switching between terminals, each of which receives a limited amount of processor time on the central computer. The OS changes among terminals so quickly that each user seems to have continuous access to the central computer. If there are many users on a system like this, the time it takes the central computer to reply can become more obvious.

Advs of MOS

- It provides less response time.
- It may help to run various jobs in a single application simultaneously.
- It helps to optimize the total job throughput of the computer.
- Various users may use the multiprogramming system at once.
- Short-time jobs are done quickly in comparison to long-time jobs.
- It may help to improve turnaround time for short-time tasks.
- It helps in improving CPU utilization and never gets idle.
- The resources are utilized smartly.

Disadvs of MOS

- It is highly complicated and sophisticated.
- The CPU scheduling is required.
- Memory management is needed in the OS because all types of tasks are stored in the main memory.
- The harder task is to handle all processes and tasks.
- If it has a large number of jobs, then long-term jobs will require a long wait.

Multitasking

- Multitasking means working on multiple tasks simultaneously, such as using our computer while listening to music. Also, using a browser, search for something on the internet and create a word document that is our assignment. It appears that all of the tasks are taking place at the same time. It is not all of the tasks happening simultaneously; the processor moves between them at such a fast pace that we believe they are happening simultaneously.
- Multitasking is similar to multiprogramming in that the CPU is assigned to a process for a specified period of time, i.e., **'Time quantum or time slice'**, after which the CPU 'Context switches' to another process. It runs various programs at the same time.
- The PC requires a huge memory to execute multitasking (**RAM or ROM**). Its primary goal is to improve the timing of the CPU's response. Users can engage with the system during multitasking, for example, by typing a letter while the printing process is running.
- Multitasking is a highly complicated system. It is based on the time slice principle, which assigns a fixed amount of time to each activity to be completed. It is especially useful when a program requires a high level of parallelism. It provides a set amount of time for each program to run.

Multitasking

Advantages

- It provides logical parallelism.
- It provides a shorter response time.
- It provides CPU utilization.

Disadvantages

- It couldn't be executed on a slow-speed processor.
- It needs a large amount of storage memory to do the work.

Timesharing OS

- A TOS allows several users to use a computer from various locations simultaneously. An OS is a program that makes a connection between the user and the system h/w. The TOS is built on multiprogramming concepts, in which multiple jobs are completed simultaneously by constantly switching between them. Its switching is lightning quick, allowing users to interact with every program because it runs without sharing the system.
- TOS use an interactive computer to allow direct interaction between the user and the system. The term **'interactive'** refers to the user's direct instructions to the system or program via an i/p device. The results will be shown on the o/p devices by the system. The results are generated faster, and the response time must be shorter than one second.

- The TOS allows several users to share computer resources simultaneously. Each user takes less CPU time because each command or action in a time-shared system is short. The TOS use strategic CPU scheduling and multiprogramming to offer every user a small time-shared system. Every user interacts with at least one separate program in memory during execution, known as a **process.**

Advantages

- It helps to reduce the CPU idle time.
- It offers the benefits of a fast response.
- It avoids the duplication of software.
- Each job gets an equal opportunity.

Disadvantages

- Data communication happens in the time-sharing operating system.
- It has the problem of reliability.

Real-time OS

- A RTOS is a type of OS designed to serve real-time applications that process data as it arrives. It completes a task within a specific time. The logical result of computation and the time required to produce the result determine the correctness of the system output. It includes methods for real-time task scheduling. It is primarily used on embedded systems. It is highly useful for timing applications or activities that are performed within a particular time limit. It uses strict time limits to drive task execution in an external environment.
- RTOSs require accurate results and timely results, which means that the results must be produced within a certain time limit, or the system will fail. It is primarily used in control device applications like automobile-engine fuel injection systems, industrial control systems, weapon systems, medical imaging systems, etc.

Advantages

- A RTOS often takes less time to shift from one task to another. Tasks are typically switched in 3 microseconds or less. This type of expedited task management ensures that key processes are performed on time.
- An RTOS is a system that is available 24/7 because it produces maximum results. As a result, it is suited for applications that must run at all times. Apart from that, an RTOS system can support different MCU systems.
- RTOSs, particularly those based on hard RTOS, are completely error-free. It ensures a more effective way of handling the errors. Furthermore, operating systems experience with jitter, an issue in which the number of errors between loops is measured. A correctly programmed RTOS can be optimized so that it suffers fewer jitters.
- An RTOS ensures that the system consumes more resources while keeping all devices active. As a result, a system using RTOS experiences very little downtime. And also hosting companies to exhibit maximum results while using RTOS.
- A RTOS focuses on one application at a time. This application will often be the one that is already executing. All others in a queue will be held in a holding pattern. As a result, critical tasks may be performed on time and within the specified deadline to achieve the exact results needed.

Disadvantages:

- A RTOS constantly experiences signal interruptions. As a result, the needed drivers must be loaded on the computer in order to get consistent speed. With the help of drivers, an RTOS will be able to respond quickly whenever an interruption occurs.
- An RTOS focuses on only one application at a time. It is used to maintain accuracy and reduce errors. All other low-priority applications need to be on waiting.
- Although a RTOS can focus on specific applications, it is not the same as multitasking. They are only designed to run some of the tasks.
- Program crashes may often be experienced while using a RTOS. Unlike a regular OS, an RTOS may not efficiently separate memory domains. As a result, processes would have a problem addressing them.
- Complex algorithms are behind an RTOS interface. These algorithms will be difficult to write for a typical user. Only a professional developer will be able to write and understand them.

System Software

- **Assembler:** it is a program that converts the assembly language into machine code.
- The output of an assembler is called an object file, which contains a combination of machine instructions as well as the data required to place these instructions in memory.
- **Assembly Language** is a low-level programming language in which there is a very strong correspondence between the instructions in the language and the hardware's machine code instructions.
- **Machine Code** are machine language instructions, which are used to control a CPU. Each instruction causes the CPU to perform a very specific task, such as a load, a store, a jump, or an ALU operation on one or more units of data in the CPU's registers or memory.

- **Compiler :** a program that translates source code from a HLL to a LL computer understandable language (e.g. assembly language, object code, or machine code) to create an executable program
- It is more intelligent than interpreter because it goes through the entire code at once
- It can tell the possible errors and limits and ranges.
- But this makes it’s operating time a little slower
- It is platform-dependent
- It help to detect error and get displayed after reading the entire code by compiler.
- “Compilers turns the HLL to binary language or machine code at once”.

- **Interpreter:** a program like compiler that converts assembly language into machine Code
- But an interpreter goes through one line of code at a time and executes it and then goes on to the next line of the code and then the next and keeps going on until there is an error in the line or the code has completed.
- It is 5 to 25 times faster than a compiler but it stops at the line where error occurs and then again if the next line has an error too.
- Where as a compiler gives all the errors in the code at once.
- Also, a compiler saves the machine codes for future use permanently but an interpreter doesn't, but an interpreter occupies less memory.

- **Linker**
- For a code to run we need to include a header file or a file saved from the library which are pre-defined if they are not included in the beginning of the program then after execution the compiler will generate errors, and the code will not work.
- Linker is a program that holds one or more object files which is created by compiler, combines them into one executable file. Linking is implemented at both time, load time and compile time. Compile time is when HLL is turns to machine code and load time is when the code is loaded into the memory by loader.

Linker is of two types:

- **1. Dynamic Linker:-**
- It is implemented during run time.
- It requires less memory.
- There are many chances of error and failure.
- Linking stored the program in virtual memory to save RAM, so we have need to share library.

- **2. Static Linker:-**
- It is implemented during compilation of source program.
- It requires more memory.
- Linking is implemented before execution in static linking.
- It is faster and portable.
- In static linking there are less chances to error and no chances to failure.

- **Loader :** a program that loads the machine codes of a program into the system memory.
- It is responsible for loading the program. It is the bare beginning of the execution of a program.
- Loading a program involves reading the contents of an executable file into memory. Only after the program is loaded on OS starts the program by passing control to the loaded program code. All the OS that supports loading have loader and many have loaders permanently in their memory.