Operating System

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.

In technical terms, It is a software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.

Definition

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

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

Types of Operating System

Operating systems are there from the very first computer generation. Operating systems keep evolving over the period of time. Following are few of the important types of operating system which are most commonly used.

Batch operating system

The users of 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. Thus, the programmers left their programs with the operator. The operator then sorts programs into batches with similar requirements.

The problems with Batch Systems are following.

- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices is slower than 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 Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems 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 receives an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.

Distributed operating System

Distributed systems use multiple central processors to serve multiple real time application and multiple users. Data processing jobs are distributed among the processors accordingly to which one can perform each job most efficiently.

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.

Network operating System

Network Operating System runs on a server and and provides 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 are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Real Time operating System

Real time system is defines 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. Real time processing is always on line

whereas on line system need not be real time. The time taken by the system to respond to an input and display of required updated information is termed as response time. So in this method response time is very less as compared to the 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. Real-time operating system has well-defined, fixed time constraints otherwise system will fail.For example Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-applicance controllers, Air traffic control system etc.

Operating System – Services

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

- It provides programs, an environment to execute.
- It provides users, services to execute the programs in a convenient manner.

Following are few common services provided by operating systems.

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

Operating System – Properties

Following are few of very important tasks that Operating System handles

Batch processing

Batch processing is a technique in which Operating System collects one programs and data together in a batch before processing starts. Operating system does the following activities related to batch processing.

- OS defines a job which has predefined sequence of commands, programs and data as a single unit.
- OS keeps a number a jobs in memory and executes them without any manual information.
- Jobs are processed in the order of submission i.e first come first served fashion.
- When job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.

Multitasking

Multitasking refers to term where multiple jobs are executed by the CPU simultaneously by switching between them.Switches occur so frequently that the users may interact with each program while it is running.

Multiprogramming

When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Following figure shows the memory layout for a multiprogramming system.

Interactivity

Interactivity refers that a User is capable to interact with computer system. Operating system does the following activities related to interactivity.

- OS provides user an interface to interact with system.
- OS managers input devices to take inputs from the user. For example, keyboard.
- OS manages output devices to show outputs to the user. For example, Monitor.
- OS Response time needs to be short since the user submits and waits for the result.

Real Time System

Real time systems represents are usually dedicated, embedded systems. Operating system does the following activities related to real time system activity.

- In such systems, Operating Systems typically read from and react to sensor data.
- The Operating system must guarantee response to events within fixed periods of time to ensure correct performance.

Distributed Environment

Distributed environment refers to multiple independent CPUs or processors in a computer system. Operating system does the following activities related to distributed environment.

- OS Distributes computation logics among several physical processors.

- The processors do not share memory or a clock.
- Instead, each processor has its own local memory.
- OS manages the communications between the processors. They communicate with each other through various communication lines.

Spooling

Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices. Operating system does the following activites related to distributed environment.

- OS handles I/O device data spooling as devices have different data access rates.
- OS maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.
- OS maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashin. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task.

Operating System – Processes

Process

A process is a program in execution. The execution of a process must progress in a sequential fashion. Definition of process is following.

- A process is defined as an entity which represents the basic unit of work to be implemented in the system.

Components of process are following.

S.N.	Component & Description
1	**Object Program** Code to be executed.
2	**Data** Data to be used for executing the program.
3	**Resources** While executing the program, it may require some resources.
4	**Status** Verifies the status of the process execution.A process can run to completion only when

all requested resources have been allocated to the process. Two or more processes could be executing the same program, each using their own data and resources.

Program

A program by itself is not a process. It is a static entity made up of program statement while process is a dynamic entity. Program contains the instructions to be executed by processor.

A program takes a space at single place in main memory and continues to stay there. A program does not perform any action by itself.

Process States

As a process executes, it changes state. The state of a process is defined as the current activity of the process.

Process can have one of the following five states at a time.

S.N.	State & Description
1	**New** The process is being created.
2	**Ready** The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3	**Running** Process instructions are being executed (i.e. The process that is currently being executed).
4	**Waiting** The process is waiting for some event to occur (such as the completion of an I/O operation).
5	**Terminated** The process has finished execution.

Operating System – Process Scheduling

Definition

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing.

Scheduling Queues

Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. The operating system also maintains other queues such as device queue. Device queue is a queue for which multiple processes are waiting for a particular I/O device. Each device has its own device queue.

This figure shows the queuing diagram of process scheduling.

- Queue is represented by rectangular box.
- The circles represent the resources that serve the queues.
- The arrows indicate the process flow in the system.

Queues are of two types

- Ready queue
- Device queue

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.

- The process could issue an I/O request and then it would be placed in an I/O queue.
- The process could create new sub process and will wait for its termination.
- The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Two State Process Model

Two state process model refers to running and non-running states which are described below.

S.N.	State & Description
1	**Running** When new process is created by Operating System that process enters into the system as in the running state.
2	**Not Running**

Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system softwares which handles process scheduling in various ways.Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types

- Long Term Scheduler
- Short Term Scheduler
- Medium Term Scheduler

OS – Scheduling algorithms

We'll discuss four major scheduling algorithms here which are following

- First Come First Serve (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round Robin(RR) Scheduling
- Multilevel Queue Scheduling

Operating System – Multi-Threading

What is Thread?

A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control.Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Folowing figure shows the working of the single and multithreaded processes.

Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight taking lesser resources than a process.
1	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
1	In multiple processing environments each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
1	If one process is blocked then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, second thread in the same task can run.
1	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
1	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

- Thread minimize context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- Economy- It is more economical to create and context switch threads.
- Utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Thread

Threads are implemented in following two ways

- **User Level Threads** — User managed threads
- **Kernel Level Threads** — Operating System managed threads acting on kernel, an operating system core.

OS – Memory Management

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the limit register is 1209000, then the program can legally access all addresses from 300000 through 411999.

Instructions and data to memory addresses can be done in following ways

- **Compile time** — When it is known at compile time where the process will reside, compile time binding is used to generate the absolute code.
- **Load time** — When it is not known at compile time where the process will reside in memory, then the compiler generates re-locatable code.
- **Execution time** — If the process can be moved during its execution from one memory segment to another, then binding must be delayed to be done at run time

Dynamic Loading

In dynamic loading, a routine of a program is not loaded until it is called by the program. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. Other routines methods or modules are loaded on request. Dynamic loading makes better memory space utilization and unused routines are never loaded.

Dynamic Linking

Linking is the process of collecting and combining various modules of code and data into a executable file that can be loaded into memory and executed. Operating system can link system level libraries to a program. When it combines the libraries at load time, the linking is called static linking and when this linking is done at the time of execution, it is called as dynamic linking.

In static linking, libraries linked at compile time, so program code size becomes bigger whereas in dynamic linking libraries linked at execution time so program code size remains smaller.

Logical versus Physical Address Space

An address generated by the CPU is a logical address whereas address actually available on memory unit is a physical address. Logical address is also known a Virtual address.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

The run-time mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.

- The value in the base register is added to every address generated by a user process which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- The user program deals with virtual addresses; it never sees the real physical addresses.

Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory to a backing store , and then brought back into memory for continued execution.

Backing store is a usually a hard disk drive or any other secondary storage which fast in access and large enough to accommodate copies of all memory images for all users. It must be capable of providing direct access to these memory images.

Major time consuming part of swapping is transfer time. Total transfer time is directly proportional to the amount of memory swapped. Let us assume that the user process is of size 100KB and the backing store is a standard hard disk with transfer rate of 1 MB per second. The actual transfer of the 100K process to or from memory will take

100KB / 1000KB per second

= 1/10 second

= 100 milliseconds

Memory Allocation

Main memory usually has two partitions

- **Low Memory** — Operating system resides in this memory.
- **High Memory** — User processes then held in high memory.

Operating system uses the following memory allocation mechanism.

S.N.	Memory Allocation	Description
1	**Single-**	In this type of allocation, relocation-register scheme is used to protect

	partition allocation	user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.
2	**Multiple-partition allocation**	In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. When a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

Fragmentation

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes can not be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Fragmentation is of two types

S.N.	Fragmentation	Description
1	**External fragmentation**	Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it can not be used.
2	**Internal fragmentation**	Memory block assigned to process is bigger. Some portion of memory is left unused as it can not be used by another process.

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

Paging

External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.

Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.

Address generated by CPU is divided into

- **Page number (p)** — page number is used as an index into a page table which contains base address of each page in physical memory.

- **Page offset (d)** — page offset is combined with base address to define the physical memory address.

Following figure show the paging table architecture.

Segmentation

Segmentation is a technique to break memory into logical pieces where each piece represents a group of related information. For example ,data segments or code segment for each process, data segment for operating system and so on. Segmentation can be implemented using or without using paging.

Unlike paging, segment are having varying sizes and thus eliminates internal fragmentation. External fragmentation still exists but to lesser extent.

Address generated by CPU is divided into

- **Segment number (s)** — segment number is used as an index into a segment table which contains base address of each segment in physical memory and a limit of segment.
- **Segment offset (o)** — segment offset is first checked against limit and then is combined with base address to define the physical memory address.

Operating System – Virtual Memory

Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.

This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occured in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.

- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Operating System – I/O Hardware

Overview

Computers operate on many kinds of devices. General types include *storage devices* (disks, tapes), *transmission devices* (network cards, modems), and *human-interface devices* (screen, keyboard, mouse). Other devices are more specialized. A device communicates with a computer system by sending signals over a cable or even through the air.

The device communicates with the machine via a connection point termed a *port* (for example, a serial port). If one or more devices use a common set of wires, the connection is called a *bus*.In other terms, a bus is a set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.

Daisy chain

When device A has a cable that plugs into device B, and device B has a cable that plugs into device C, and device C plugs into a port on the computer, this arrangement is called a **daisy chain**. It usually operates as a bus.

Controller

A controller is a collection of electronics that can operate a port, a bus, or a device. A serial-port controller is an example of a simple device controller. This is a single chip in the computer that controls the signals on the wires of a serial port.

The SCSI bus controller is often implemented as a separate circuit board (a host adapter) that plugs into the computer. It contains a processor, microcode, and some private memory to enable it to process the SCSI protocol messages. Some devices have their own built-in controllers.

I/O port

An I/O port typically consists of four registers, called the **status** , **control**, **data-in**, and **data-out** registers.

S.N.	Register & Description
1	**Status Register** The status register contains bits that can be read by the host. These bits indicate states such as whether the current command has completed, whether a byte is available to be read from the data-in register, and whether there has been a device error.
2	**Control register** The control register can be written by the host to start a command or to change the mode of a device. For instance, a certain bit in the control register of a serial port chooses between full-duplex and half-duplex communication, another enables parity checking, a third bit sets the word length to 7 or 8 bits, and other bits select one of the speeds supported by the serial port.
3	**Data-in register** The data-in register is read by the host to get input.
4	**Data-out register** The data out register is written by the host to send output.

Polling

Polling is a process by which a host waits for controller response.It is a looping process, reading the status register over and over until the busy bit of status register becomes clear. The controller uses/sets the busy bit when it is busy working on a command, and clears the busy bit when it is ready to accept the next command. The host signals its wish via the command-ready bit in the command register. The host sets the command-ready bit when a command is available for the controller to execute.

In the following example, the host writes output through a port, coordinating with the controller by handshaking

- The host repeatedly reads the busy bit until that bit becomes clear.
- The host sets the write bit in the command register and writes a byte into the data-out register.
- The host sets the command-ready bit.
- When the controller notices that the command-ready bit is set, it sets the busy bit.
- The controller reads the command register and sees the write command.
- It reads the data-out register to get the byte, and does the I/O to the device.
- The controller clears the command-ready bit, clears the error bit in the status register to indicate that the device I/O succeeded, and clears the busy bit to indicate that it is finished.

I/O devices

I/O Devices can be categorized into following category.

S.N.	Category & Description
1	**Human readable** Human Readable devices are suitable for communicating with the computer user. Examples are printers, video display terminals, keyboard etc.
2	**Machine readable** Machine Readable devices are suitable for communicating with electronic equipment. Examples are disk and tape drives, sensors, controllers and actuators.
2	**Communication** Communication devices are suitable for communicating with remote devices. Examples are digital line drivers and modems.

Following are the differences between I/O Devices

S.N.	Criteria & Description
1	**Data rate** There may be differences of several orders of magnitude between the data transfer rates.
2	**Application** Different devices have different use in the system.
3	**Complexity of Control** A disk is much more complex whereas printer requires simple control interface.
4	**Unit of transfer** Data may be transferred as a stream of bytes or characters or in larger blocks.
5	**Data representation** Different data encoding schemes are used for different devices.
6	**Error Conditions** The nature of errors differs widely from one device to another.

Direct Memory Access (DMA)

Many computers avoid burdening the main CPU with programmed I/O by offloading some of this work to a special purpose processor. This type of processor is called, a Direct Memory Access(DMA) controller. A special control unit is used to transfer block of data directly between an external device and the main memory, without intervention by the processor. This approach is called Direct Memory Access(DMA).

DMA can be used with either polling or interrupt software. DMA is particularly useful on devices like disks, where many bytes of information can be transferred in single I/O operations. When used with an interrupt, the CPU is notified only after the entire block of data has been transferred. For each byte or

word transferred, it must provide the memory address and all the bus signals controlling the data transfer. Interaction with a device controller is managed through a device driver.

Handshaking is a process between the DMA controller and the device controller. It is performed via wires using terms DMA request and DMA acknowledge.

Step	Description
1	Device driver is instructed to transfer disk data to a buffer address X.
2	Device driver then instruct disk controller to transfer data to buffer.
3	Disk controller starts DMA transfer.
4	Disk controller sends each byte to DMA controller.
5	DMA controller transfers bytes to buffer, increases the memory address, decreases the counter C until C becomes zero.
6	When C becomes zero, DMA interrupts CPU to signal transfer completion.

Device Controllers

A computer system contains a many types of I/O devices and their respective controllers

- network card
- graphics adapter
- disk controller
- DVD-ROM controller
- serial port
- USB
- sound card

Operating System – I/O Softwares

Interrupts

The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt, performs the necessary processing and executes a interrupt instruction to return the CPU to its execution state.

The basic mechanism of interrurpt enables the CPU to respond to an asynchronous event, such as when a device controller become ready for service. Most CPUs have two interrupt request lines.

- **non-maskable interrupt** – Such kind of interrupts are reserved for events like unrecoverable memory errors.
- **maskable interrupt** – Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

The interrupt mechanism accepts an address – a number that selects a specific interrupt handling routine/function from a small set.In most architectures, this address is an offset stored in a table called the interrupt vector table. This vector contains the memory addresses of specialized interrupt handlers.

Application I/O Interface

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O interfaces with respected to devices.

- **Character-stream / block -** A character-stream device transfers bytes in one by one fashion, whereas a block device transfers a complete unit of bytes.
- **Sequential / random-access** – A sequential device transfers data in a fixed order determined by the device, random-access device can be instructed to seek position to any of the available data storage locations.
- **Synchronous / asynchronous** – A synchronous device performs data transfers with known response time where as an asynchronous device shows irregular or unpredictable response time.
- **Sharable / dedicated** – A sharable device can be used concurrently by several processes or threads but a dedicated device cannot be used.
- **Speed of operation** – Device speeds may range from a few bytes per second to a few gigabytes per second.
- **Read-write, read only, or write only** – Some devices perform both input and output, but others support only one data direction that is read only.

Clocks

Clocks are also called timers. The clock software takes the form of a device driver though a clock is neither a blocking device nor a character based device. The clock software is the clock driver. The exact

function of the clock driver may vary depending on operating system. Generally, the functions of the clock driver include the following.

S.N.	Task	Description
1	Maintaining the time of the day	The clock driver implements the time of day or the real time clock function.It requires incrementing a counter at each clock tick.
2	Preventing processes from running too long	As a process is started, the scheduler initializes the quantum counter in clock ticks for the process. The clock driver decrements the quantum counter by 1, at every clock interrupt. When the counter gets to zero , clock driver calls the scheduler to set up another process. Thus clock driver helps in preventing processes from running longer than time slice allowed.
3	Accounting for CPU usage	Another function performed by clock driver is doing CPU accounting. CPU accounting implies telling how long the process has run.
4	Providing watchdog timers for parts of the system itself	Watchdog timers are the timers set by certain parts of the system. For example, to use a floppy disk, the system must turn on the motor and then wait about 500msec for it to comes up to speed.

Kernel I/O Subsystem

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following are some of the services provided.

- **Scheduling** – Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.
- **Buffering** – Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a device with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

- **Caching** – Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
- **Spooling and Device Reservation** A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.
- **Error Handling** An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Device driver

Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviours on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver are following.

- To accept request from the device independent software above it.
- To see to it that the request is executed.

Source: tutorialpoint