YOBE STATE UNIVERSITY

FACULTY OF SCIENCE

DEPARTMENTOF COMPUTER SCIENCE

LECTURES NOTE

COURSE CODE: CSC 2303

COURSE TITLE: OPERATING SYSTEM I

CREDIT UNITS: 3

SEMESTER: FIRST

COURSE LECTURER: YA'U NUHU

EMAIL: yaunuhu20@gmail.com

2023/2024 SESSION

OPERATING SYSTEM TIPS

i. Operating systems, the silent conductors of our digital lives, often go unnoticed until something goes wrong. But with a few handy tips and tricks, you can unlock their full potential and keep your computer running smoothly.

ii. Keyboard Shortcuts are your friends: Learn basic keyboard shortcuts for common actions (copy, paste, undo, etc.) to navigate your OS faster.

iii. Keep it updated: Regularly update your operating system and software to ensure security and performance.

iv. Back up your data: Regularly back up your important files to prevent data loss from crashes or hardware failures.

v. Explore and customize: Don't be afraid to tinker with your OS settings and personalize your experience. Most operating systems offer a variety of customization options.

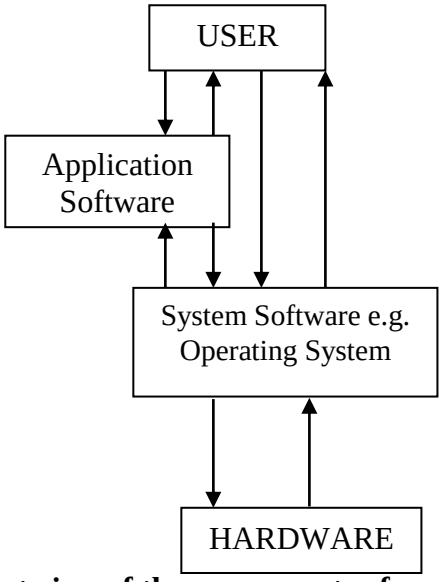
INTRODUCTION

This is the executive manager, the part of computing system that manages all of the hardware and software. It controls every file, every device, and every section of the main memory and every non-second of processing time. It controls who can use the system and how. It is an integrated set of programs that is used to manage the various resources and overall operation of a computer system.

Its prime objective is to improve the performance and efficiency of a computer system. Thus like a manager of a company an operating system is responsible for the smooth and efficient operation of the entire computer system. Examples of operating systems are; Windows, Dos, Linux, UNIX, e.t.c.

An operating system is a set of programs that enable the user to communicate with a computer system and also enable a computer's equipment to communicate with each other. An operating system acts as an interface between a user of the computer and the hardware. The operating system is used to bridge the gap between the hardware and the user.

Controlling the computer involves software at several levels. We will differentiate kernel services, library services, and application-level services, all of which are part of the operating system. Processes run Applications, which are linked together with libraries that perform standard services. The kernel supports the processes by providing a path to the peripheral devices. The kernel responds to service calls from the processes and interrupts from the devices. The core of the operating system is the kernel, a control program that functions in **privileged state** (an execution context that allows all hardware instructions to be executed), reacting to interrupts from external devices and to service requests and traps from processes. Generally, the kernel is a permanent resident of the computer. It creates and terminates processes and responds to their request for service.



Abstract view of the components of a computer system

The hardware provides the basic computing services. Application software/programs make use of these resources to solve the user problems. There may be many users using different application programs to solve different problems. The OS controls and coordinates the use of hardware among various application programs for different users. So we can view the OS as a Resource Manager and an extended machine or Virtual machine.

Operating system as a Resource Manager.

Modern computers consist of processors, memories, timers, disks, terminals, magnetic tape drives, network interfaces, printers and a wide variety of other devices. The job of OS is to provide an orderly, controlled allocation of the processors and memories, and input/output devices among the various programs competing for them. For example, programs trying to print their output simultaneously on the same printer.

In this view, the primary task is to keep track of who is using which resource, to grant resource requests, to account for usage, and to mediate conflicting requests from different programs and users. Bare hardware can not be easily used so programs are written for certain basic common operations. A layer of software is embedded on hardware to manage the system and present the user with an interface or extended machine.

Operating system as an extended machine

The architecture of most computers at the machine language level is primitive and difficult to program especially for I/O. For example, reading from a floppy disk requires knowledge of how data is organised in a floppy disk and address where the data to be written and how to switch on the floppy device motor, how to position the R/W head (Read/Write) to the particular address and many more details. The programmer doesn't want to get too intimately involved with all these details; instead what the programmer wants is a simple high level abstraction to deal with. In case of disks, typical abstraction would be that of disk contains a collection of named files. Each file can be opened for reading, writing, then read, or written and finally closed. The program that hides the truth about the hardware from the programmer and presents a nice simple view of named files that are read and written, is the "OPERATING SYSTEM". It also conceals a lot of unpleasant business concerning interrupts, timers, memory management and other low level features. In each abstraction presented to the user is simpler and easier to use than the underlying hardware.

In this view the function of the OS is to present the user with equivalent of an extended machine or virtual machine which is easier to program. Operating Systems are fundamentally event-driven systems - they wait for an event to happen, respond appropriately to the event and then wait for the next event.

Examples:

i. ***User hits a key*.** The keystroke is echoed on the screen. A user program issues a system call to read a file. The operating system figures out which disk blocks to bring in, and generates a request to the disk controller to read the disk blocks into memory. The disk controller

finishes reading in the disk block and generates an interrupt. The OS moves the read data into the user program and restarts the user program.

ii. ***A Mosaic or Netscape user asks for a URL to be retrieved*.** This eventually generates requests to the OS to send request packets out over the network to a remote WWW server. The OS sends the packets. The response packets come back from the WWW server, interrupting the processor. The OS figures out which process should get the packets, then routes the packets to that process. Time-slice timer goes off. The OS must save the state of the current process, choose another process to run, and then give the CPU to that process.

Objectives of Operating Systems

Modern Operating systems generally have following three major goals. Operating systems generally accomplish these goals by running processes in low privilege and providing service calls that invoke the operating system kernel in high-privilege state.

i. **To hide details of hardware by creating abstraction.**

An abstraction is software that hides lower level details and provides a set of higher-level functions. An operating system transforms the physical world of devices, instructions, memory, and time into virtual world that is the result of abstractions built by the operating system. There are several reasons for abstraction.

- **First,** the code needed to control peripheral devices is not standardized. Operating systems provide subroutines called device drivers that perform operations on behalf of programs for example, input/output operations.
- **Second,** the operating system introduces new functions as it abstracts the hardware. For instance, operating system introduces the file abstraction so that programs do not have to deal with disks.
- **Third,** the operating system transforms the computer hardware into multiple virtual computers, each belonging to a different program. Each program that is running is called a process. Each process views the hardware through the lens of abstraction.
- **Fourth,** the operating system can enforce security through abstraction.

ii. **To allocate resources to processes (Manage resources).**

An operating system controls how **processes** (the active agents) may access **resources** (passive entities).

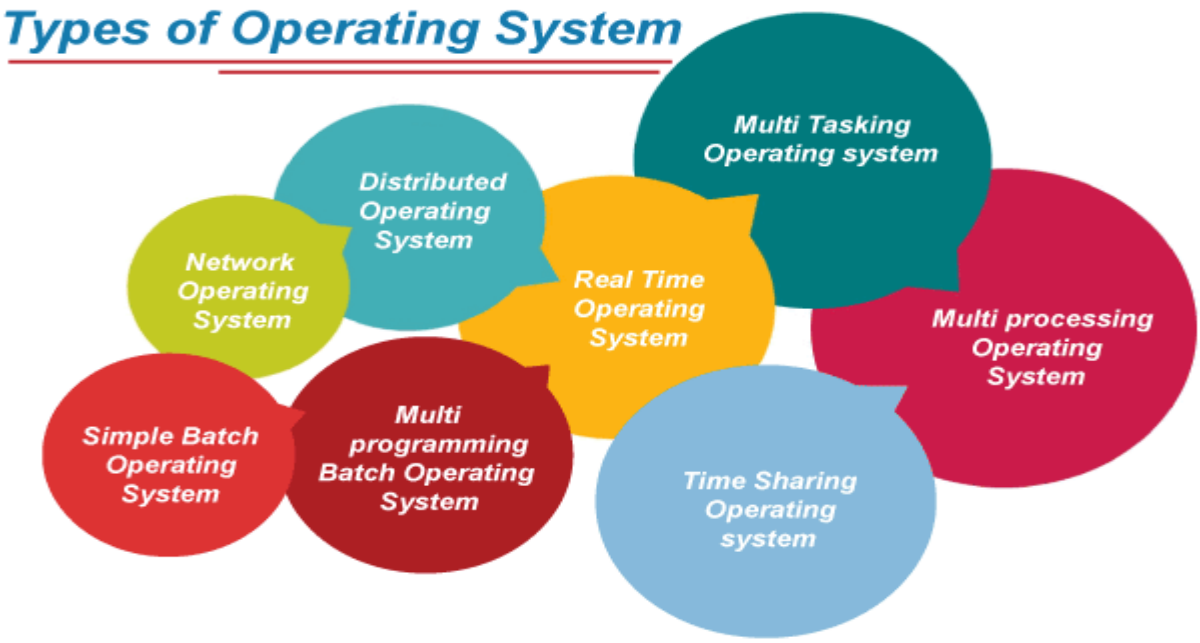
iii. **Provide a pleasant and effective user interface**

The user interacts with the operating systems through the user interface and usually interested in the "look and feel" of the operating system. The most important components of the user interface are the command interpreter, the file system, on-line help, and application integration. The recent trend has been toward increasingly integrated graphical user interfaces that encompass the activities of multiple processes on networks of computers.

Types of Operating Systems (OS)

An operating system is a well-organized collection of programs that manages the computer hardware.

It is a type of system software that is responsible for the smooth functioning of the computer system.

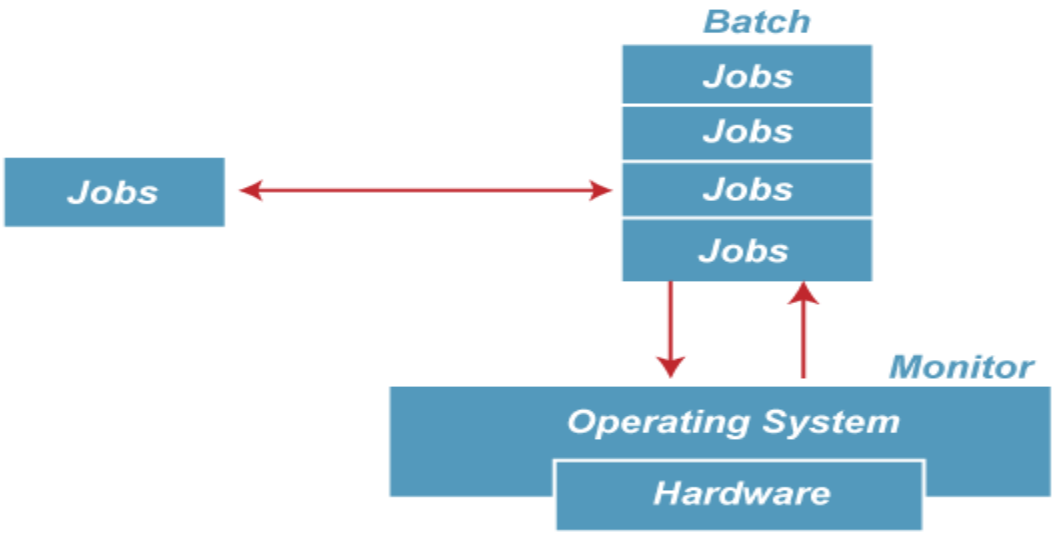


Batch Operating System

In the 1970s, Batch processing was very popular. In this technique, similar types of jobs were batched together and executed in time. People were used to having a single computer which was called a mainframe.

In Batch operating system, access is given to more than one person; they submit their respective jobs to the system for the execution.

The system put all of the jobs in a queue on the basis of first come first serve and then executes the jobs one by one. The users collect their respective output when all the jobs get executed.



The purpose of this operating system was mainly to transfer control from one job to another as soon as the job was completed. It contained a small set of programs called the resident monitor that always resided in one part of the main memory. The remaining part is used for servicing jobs.

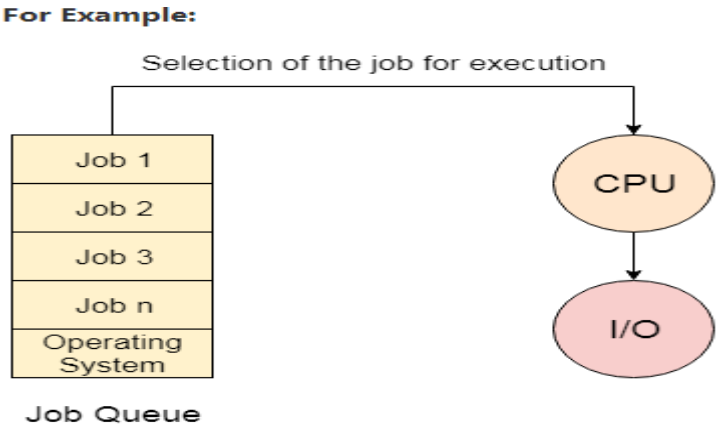
Advantages of Batch OS

The use of a resident monitor improves computer efficiency as it eliminates CPU time between two jobs.

Disadvantages of Batch OS

i. Starvation

Batch processing suffers from starvation.



There are five jobs J1, J2, J3, J4, and J5, present in the batch. If the execution time of J1 is very high, then the other four jobs will never be executed, or they will have to wait for a very long time. Hence the other processes get starved.

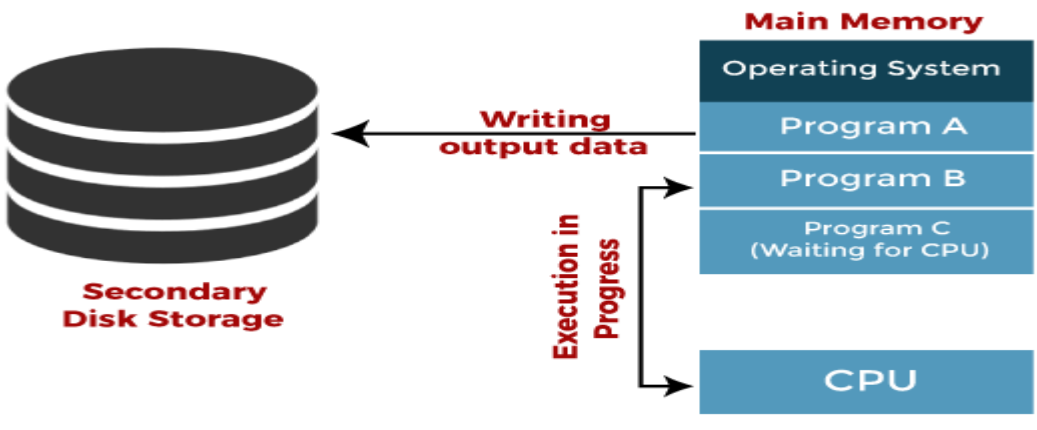
ii. Not Interactive

Batch Processing is not suitable for jobs that are dependent on the user's input. If a job requires the input of two numbers from the console, then it will never get it in the batch processing scenario since the user is not present at the time of execution.

Multiprogramming Operating System

Multiprogramming is an extension to batch processing where the CPU is always kept busy. Each process needs two types of system time: CPU time and IO time.

In a multiprogramming environment, when a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.



Jobs in multiprogramming system

Advantages of Multiprogramming OS

i. Throughout the system, it increased as the CPU always had one program to execute.

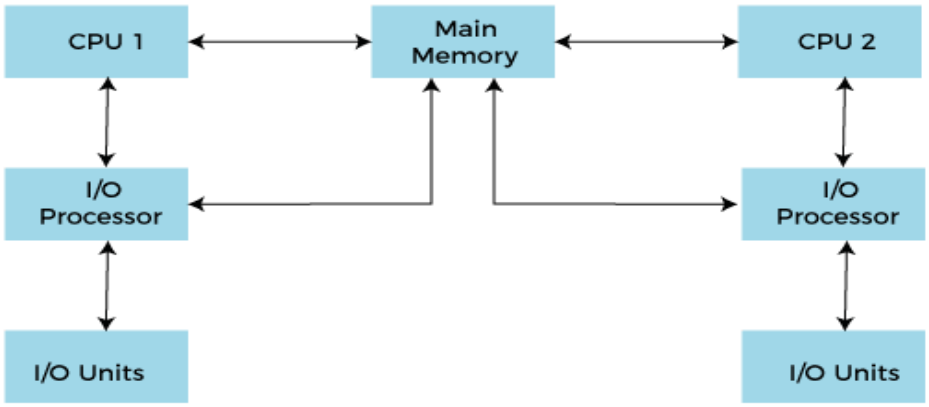
ii. Response time can also be reduced.

Disadvantages of Multiprogramming OS

Multiprogramming systems provide an environment in which various systems resources are used efficiently, but they do not provide any user interaction with the computer system.

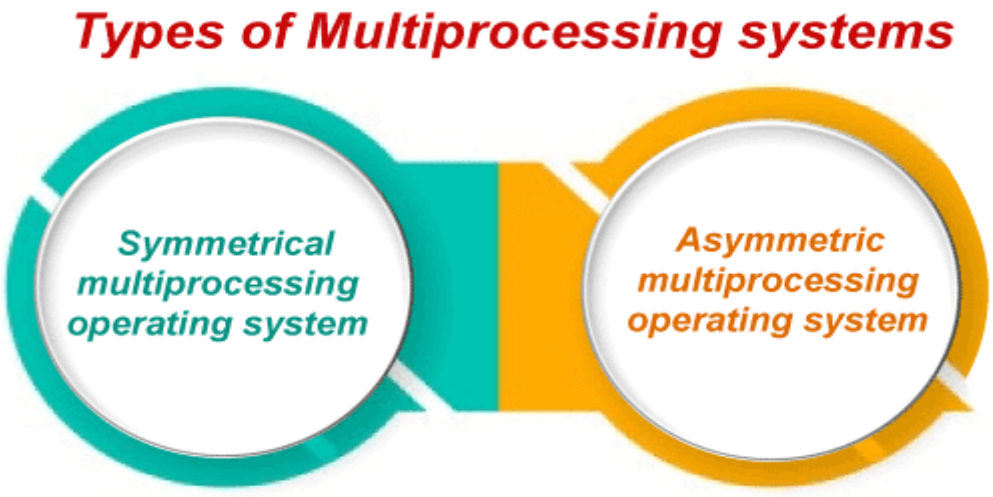
Multiprocessing Operating System

In Multiprocessing, Parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.



Working of Multiprocessor System

In Multiprocessing, Parallel computing is achieved. More than one processor present in the system can execute more than one process simultaneously, which will increase the throughput of the system.



Advantages of Multiprocessing operating system:

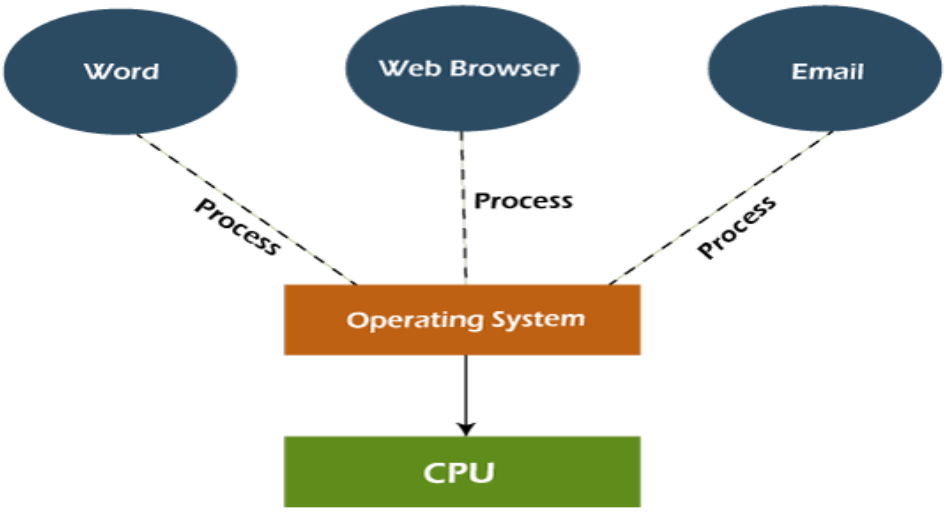
i. Increased reliability: Due to the multiprocessing system, processing tasks can be distributed among several processors. This increases reliability as if one processor fails, the task can be given to another processor for completion.

ii. Increased throughout: As several processors increase, more work can be done in less.

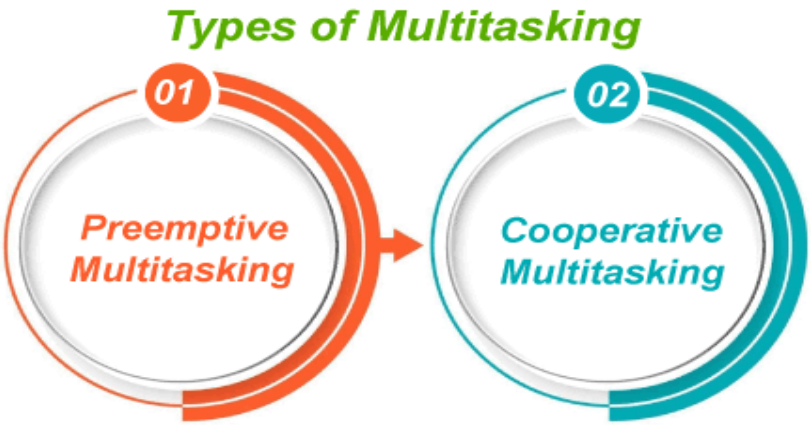
Disadvantages of Multiprocessing operating System

Multiprocessing operating system is more complex and sophisticated as it takes care of multiple CPUs simultaneously.

Multitasking Operating System



The multitasking operating system is a logical extension of a multiprogramming system that enables **multiple** programs simultaneously. It allows a user to perform more than one computer task at the same time.



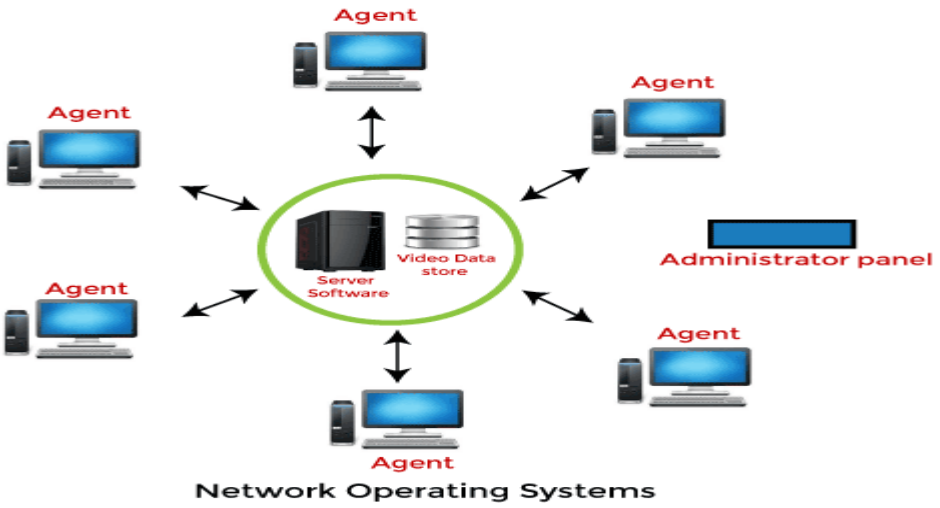
Advantages of Multitasking operating system

i. This operating system is more suited to supporting multiple users simultaneously.
ii. The multitasking operating systems have well-defined memory management.

Disadvantages of Multitasking operating system

The multiple processors are busier at the same time to complete any task in a multitasking environment, so the CPU generates more heat.

Network Operating System



An Operating system, which includes software and associated protocols to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.

- Just as a computer cannot operate without a computer operating system, a network of computers cannot operate without a network operating system.
- Without a network operating system, individual computers cannot share resources, and other users cannot make use of those resources

Depending on a network operating system's manufacturer, a desktop computer's networking software configuration can be either:

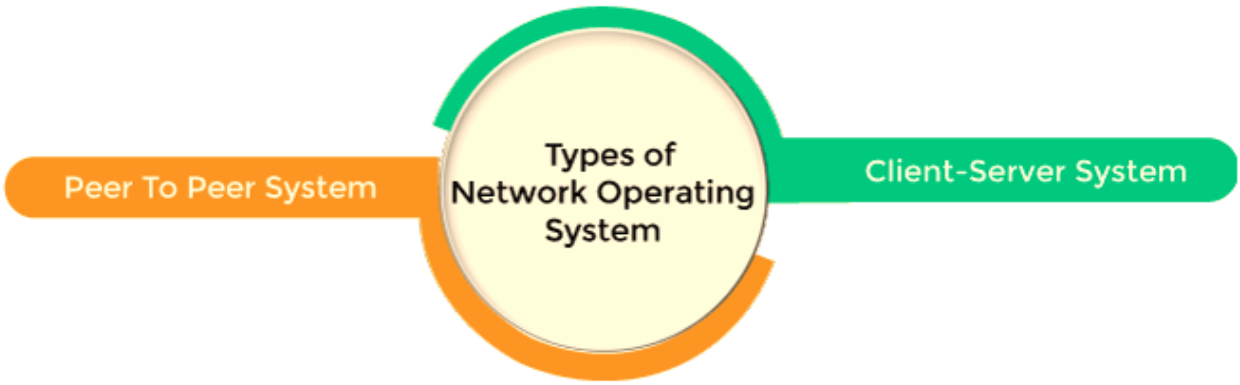
- **(A) Added to the computer's own operating system**, example: Novell's NetWare where the client computer's networking software is added on to its existing computer operating system. The desktop computer needs both operating systems in order to handle stand-alone and networking functions together.
- **(B) Integrated with it**, example: Windows 2000 Server/Windows 2000 Professional, Windows NT Server/Windows NT Workstation, Windows 98, Windows 95, and AppleTalk.

- ***A network operating system (NOS)*** provides services to clients over a network.
- Operating systems in general (client OS and NOS) coordinate the interaction between the computer and the programs.
- It also controls the allocation and use of hardware resources such as:
  - Memory.
  - CPU time.
  - Disk space.
  - Peripheral devices.

Networking environment works as follows:

i. Servers provide resources to the network clients
ii. Client network OS makes these resources available to the client computer.

- The network and the client operating systems are coordinated so that all portions of the network function properly.



Advantages of Network Operating System

i. In this type of operating system, network traffic reduces due to the division between clients and the server.

ii. This type of system is less expensive to set up and maintain.

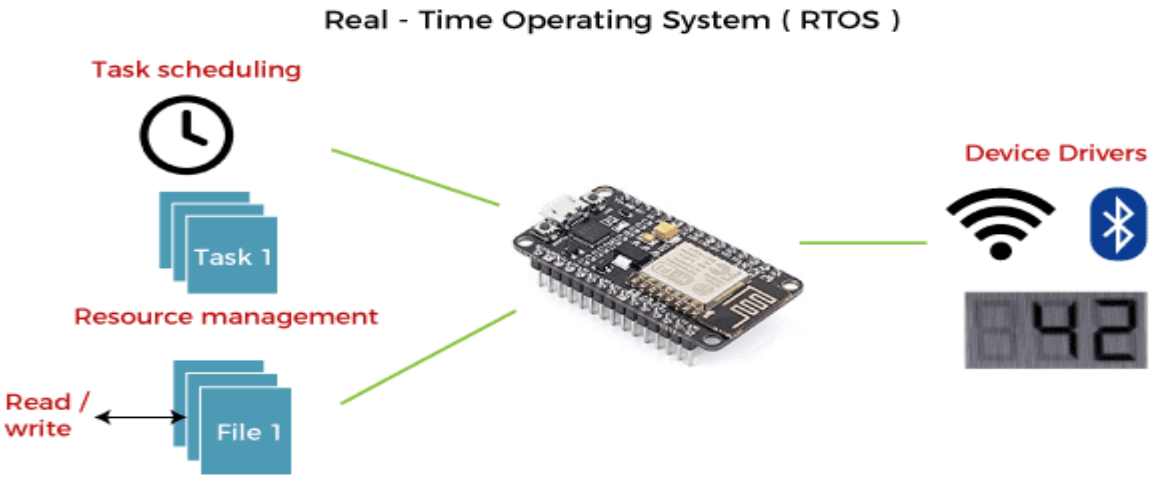
Disadvantages of Network Operating System

i. In this type of operating system, the failure of any node in a system affects the whole system.

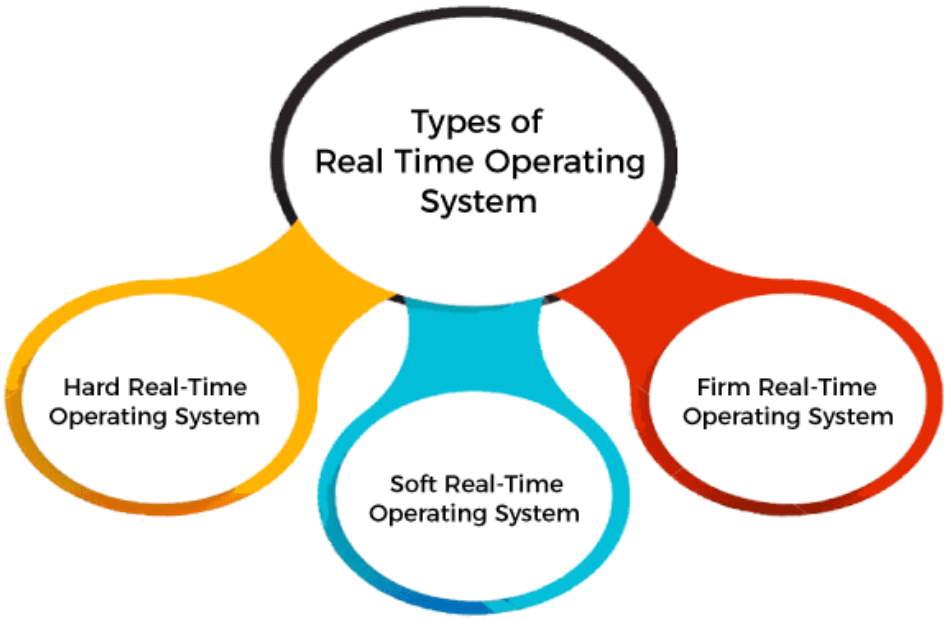
ii. Security and performance are important issues. So trained network administrators are required for network administration.

Real Time Operating System

In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed, otherwise, the huge loss will be there, or even if the result is produced, it will be completely useless.



The Application of a Real-Time system exists in the case of military applications, if you want to drop a missile, then the missile is supposed to be dropped with a certain precision.



Advantages of Real-time operating system:

i. Easy to layout, develop and execute real-time applications under the real-time operating system.

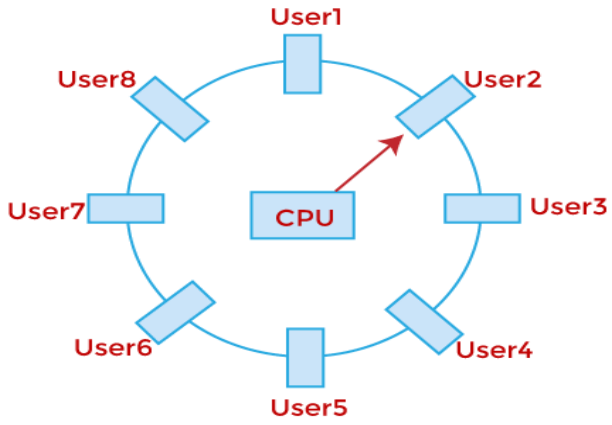
ii. In a Real-time operating system, the maximum utilization of devices and systems.

Disadvantages of Real-time operating system:

i. Real-time operating systems are very costly to develop.
ii. Real-time operating systems are very complex and can consume critical CPU cycles.

Time-Sharing Operating System

In the Time Sharing operating system, computer resources are allocated in a time-dependent fashion to several programs simultaneously. Thus it helps to provide a large number of user's direct access to the main computer. It is a logical extension of multiprogramming. In time-sharing, the CPU is switched among multiple programs given by different users on a scheduled basis.



Timesharing in case of 8 users

A time-sharing operating system allows many users to be served simultaneously, so sophisticated CPU scheduling schemes and Input/output management are required.

Time-sharing operating systems are very difficult and expensive to build.

Advantages of Time Sharing Operating System

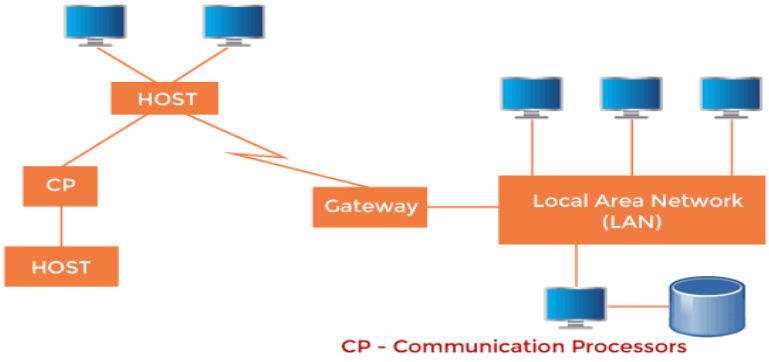
i. The time-sharing operating system provides effective utilization and sharing of resources.
ii. This system reduces CPU idle and response time.

Disadvantages of Time Sharing Operating System

i. Data transmission rates are very high in comparison to other methods.
ii. Security and integrity of user programs loaded in memory and data need to be maintained as many users access the system at the same time.

Distributed Operating System

The Distributed Operating system is not installed on a single machine, it is divided into parts, and these parts are loaded on different machines. A part of the distributed Operating system is installed on each machine to make their communication possible. Distributed Operating systems are much more complex, large, and sophisticated than Network operating systems because they also have to take care of varying networking protocols.



A Typical View of a Distributed System

Advantages of Distributed Operating System

i. The distributed operating system provides sharing of resources.

ii. This type of system is fault-tolerant.

Disadvantages of Distributed Operating System

Protocol overhead can dominate computation cost.

Functions of the operating system

Hide the fact that resources are being shared. This gives the concept of the OS as a virtual machine.

I. Bridge the gap between the machine and the user levels.

II. Memory management

- Checks the validity of each request for memory space and if legal, it allocates a portion which isn't in use.
- In multi-user environment, it sets up a table to keep track of who is using which section of the memory.
- Preserves the space in the main memory that is occupied by the Operating System itself.

III. Processor management

- It handles the jobs as they enter into the system. This is handled by the job scheduler, which accepts or rejects the jobs.
- It then manages the processes within those jobs. This is handled by the Process scheduler and decides which process gets the CPU and for how long.

IV. Device management

This monitors every device channel and control units. Its job is to choose the most efficient way to allocate all of the system devices like keyboard, printer, disk, and modem.

V. File management

- This keeps track of every file in the system, including utility programs like compilers, interpreters, and assemblers, data files and application programs.
- Enforces protection policies; access rights, access control. It enforces access restrictions on each file depending on how the file was declared.

- The file manager also allocates the resources by opening the file and then de-allocates it by closing the file. It helps in backing up of files and recovery mechanisms during competition for information.

VI. Synchronisation and communication

Maintenance of internal clock and log off system usage for all users.

Operating System Components

Even though, not all systems have the same structure many modern operating systems share the same goal of supporting the following types of system components.

Process Management

The operating system manages many kinds of activities ranging from user programs to system programs like printer spooler, name servers, file server etc. Each of these activities is encapsulated in a process. A process includes the complete execution context (code, data, PC, registers, OS resources in use etc.)

It is important to note that a process is not a program. A process is only ONE instant of a program in execution. There are many processes can be running the same program. The five major activities of an operating system in regard to process management are

- Creation and deletion of user and system processes.
- Suspension and resumption of processes.
- A mechanism for process synchronization.
- A mechanism for process communication.
- A mechanism for deadlock handling.

Main-Memory Management

Primary-Memory or Main-Memory is a large array of words or bytes. Each word or byte has its own address. Main-memory provides storage that can be access directly by the CPU. That is to say for a program to be executed, it must in the main memory.

The purpose of memory management is to optimize use of random access memory (RAM). The OS has the responsibility to allocate data and programs to an area of the RAM while they are being processed; to monitor carefully the contents of these items (data and /or programs) in memory; and to clear these items from memory when they are no longer required by CPU. Some operating system use virtual memory (VM) to optimize RAM. With VM, the OS allocates a portion of a storage medium usually the hard disk to function as an additional RAM.

The area of the hard disk used is called swap file. Programs working on the foreground have to be on memory, once a user leaves a program to work on another one, the previous program and /or data moved to the swap file (i.e. if there are too many programs already on the RAM).

Hence the operating system exchanges data and programs (swap) between memory (RAM) and the hard disk. The amount of data and/or program exchanged at a given time is called a page. Thus, the technique of swapping items between memory and storage is called paging. When an OS spends much of its time paging, instead of executing application software, it is said to be thrashing. In it worth nothing that when a user is working on application, has stopped responding and the LED for that computers hard disk is blinking, the OS is probably trashing. To stop thrashing, one should quit

the program that stopped responding. If trashing occurs frequently on your computer, you may need to increase the size of your RAM

- The major activities of an operating in regard to memory-management are:
- Keep track of which part of memory are currently being used and by whom.
- Decide which process is loaded into memory when memory space becomes available.
- Allocate and deallocate memory space as needed.

File Management

A file is a collected of related information defined by its creator. Computer can store files on the disk (secondary storage), which provide long term storage. Some examples of storage media are magnetic tape, magnetic disk and optical disk. Each of these media has its own properties like speed, capacity, and data transfer rate and access methods. A file system normally organized into directories to ease their use. These directories may contain files and other directions. The five main major activities of an operating system in regard to file management are:

- The creation and deletion of files.
- The creation and deletion of directions.
- The support of primitives for manipulating files and directions.
- The mapping of files onto secondary storage.
- The back up of files on stable storage media.

I/O System Management

I/O subsystem hides the peculiarities of specific hardware devices from the user. Only the device driver knows the peculiarities(individualities) of the specific device to which it is assigned.

Secondary-Storage Management

Generally speaking, systems have several levels of storage, including primary storage, secondary storage and cache storage. Instructions and data must be placed in primary storage or cache to be referenced by a running program. Because main memory is too small to accommodate all data and programs, and its data are lost when power is lost, the computer system must provide secondary storage to back up main memory. Secondary storage consists of tapes, disks, and other media designed to hold information that will eventually be accessed in primary storage (primary, secondary, cache) is ordinarily divided into bytes or words consisting of a fixed number of bytes. Each location in storage has an address; the set of all addresses available to a program is called an address space. The three major activities of an operating system in regard to secondary storage management are:

- Managing the free space available on the secondary-storage device.
- Allocation of storage space when new files have to be written.
- Scheduling the requests for memory access.

Networking

A distributed system is a collection of processors that do not share memory, peripheral devices, or a clock. The processors communicate with one another through communication lines called network. The communication-network design must consider routing and connection strategies, and the problems of contention and security.

i. Protection System

If a computer system has multiple users and allows the concurrent execution of multiple processes, then the various processes must be protected from one another's activities. Protection refers to mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system.

ii. Command Interpreter System

A command interpreter is an interface of the operating system with the user. The user gives commands with are executed by operating system (usually by turning them into system calls). The main function of a command interpreter is to get and execute the next user specified command. Command-Interpreter is usually not part of the kernel, since multiple command interpreters (shell, in UNIX terminology) may be support by an operating system, and they do not really need to run in kernel mode.

There are two main advantages to separating the command interpreter from the kernel.

- If we want to change the way the command interpreter looks, i.e., I want to change the interface of command interpreter, I am able to do that if the command interpreter is separate from the kernel. I cannot change the code of the kernel so I cannot modify the interface.
- If the command interpreter is a part of the kernel it is possible for a malicious process to gain access to certain part of the kernel that it showed not have to avoid this ugly scenario it is advantageous to have the command interpreter separate from kernel.

Operating Systems Services

The following are the five services provided by operating systems to the convenience of the users.

Program Execution

The purpose of a computer system is to allow the user to execute programs. So the operating system provides an environment where the user can conveniently run programs. The user does not have to worry about the memory allocation or multitasking or anything. These things are taken care of by the operating systems.

Running a program involves the allocating and deallocating memory and CPU scheduling in case of multi-processing. These functions cannot be given to the user-level programs. So user-level programs cannot help the user to run programs independently without the help from operating systems.

I/O Operations

Each program requires an input and produces output. This involves the use of I/O. The operating system hides the user`s details of underlying hardware for the I/O. All the user sees is that the I/O has been performed without any details. So the operating system by providing I/O makes it convenient for the users to run programs. For efficiency and protection, users cannot control I/O so this service cannot be provided by user-level programs.

File System Manipulation

The output of a program may need to be written into new files or input taken from some files, the operating system provides this service. The user does not have to worry about secondary storage management. User gives a command for reading or writing to a file and sees his/her task accomplished. Thus operating system makes it easier for the users to accomplish their tasks.

This service involves secondary storage management. The speed of I/O that depends on secondary storage management is critical to the speed of many programs and hence I think it is best relegated to the operating systems to manage it than giving individual users the control of it. It is not difficult for the user-level programs to provide these services but for above mentioned reasons it is best if this service s left with operating system.

Communications

There are instances where processes need to communicate with each other to exchange information. It may be between processes running on the same computer or running on the different computers. By providing this service the operating system relieves the user of the worry of passing messages between processes. In case where the messages need to be passed to processes on the other computers through a network it can be done by the user programs. The user program may be customized to the specifics of the hardware through which the message transits and provides the service interface to the operating system.

Error Detection

An error is one part of the system may cause malfunctioning of the complete system. To avoid such a situation, the operating system constantly monitors the system to detect errors. This relieves the user of the worry of errors propagating to various part of the system and causing malfunctioning.

This service cannot be allowed to be handled by user programs because it involves monitoring and in cases altering area of memory or deallocation of memory for a faulty process. Or may be relinquishing the CPU of a process that goes into an infinite loop. These tasks are too critical to be

handed over to the user programs. A user program if given these privileges can interfere with the correct (normal) operation of the operating systems.

Memory Management Systems/Schemes

i. Contiguous, Real Memory Management System
   - Single Contiguous Memory Management System.
   - Fixed Partitioned Memory Management System
   - Variable Partitioned Memory Management System

ii. Non contiguous Real Memory Management System
   - Paged Memory Management System.
   - Segmented Memory Management System.
   - Combined Memory Management System.

iii. Non Contiguous Virtual Memory Management System.

Over view

- **Contiguous, Real Memory Management System**

  Is a scheme that expects the program to be loaded in contiguous memory locations i.e. contiguous blocks.

- **Non-Contiguous Memory Management System.**

  Is a scheme which allows the program to be divided into different chunks/partitions and to be loaded at different positions of the memory.

  In **paging**, the partitions are of the same size where as in **segmentation** the memory partitions can be of different sizes.

- **Real memory management system**

  This is where the full process or job is expected to be loaded in the memory before execution.

- **Virtual memory management system**

  Is a system that can start executing a process even with only part of the process image is loaded into memory.

Contagious Memory Allocation

- **Single contiguous memory management scheme**

It is also referred to as a single user contiguous scheme. The scheme works as follows;

- All ready processes are held on the disk as executable images.
- At anytime one of them runs in the main memory
- When this process is blocked, it is swapped out from the main memory to the disk.
- The next highest priority process is swapped in the main memory from the disk and it starts running.
- Thus there is only one process in the main memory even if conceptually, it is a multi-programming system.

Each program to be processed is loaded in its entirety into memory and allocated as much contiguous space in memory as needed. If the program is too large for the available memory, no execution could

be effected. More-over early computers had very little memory. The solution either size of memory to be increased or program modified.

Notice that the amount of work done by the operating system's memory manager is minimal, 0nly two hardware items are needed a **register** to store the **"base address"** and an **"accumulator".** Once the program is entirely loaded into memory, it remains there until execution is complete, either through normal termination or by intervention of the OS.

The major **problems** with this type of memory allocation scheme are:

i. Wastage of memory space in case of a small job.
ii. It could not support multiprogramming since it was unable to handle more than one job at a time.
iii. Inefficient because of the number of jobs it could execute at a time thus the CPU was idle most of the time.

But there were some **advantages** such as;

- Very first Access Time
- Very little time complexity thus no much overhead due to limited allocation and de-allocation thus activity was limited.

Fixed petitioned memory management scheme

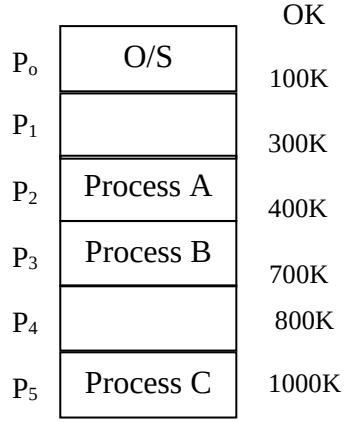
This was the first attempt to allow multi-programming, by creating "fixed partitions" (static partitions) within the main memory – one partition for each job.

Once a partition was assigned to a job, no other job could be allowed to enter its boundaries, either accidentally or internationally.

This partition allocation scheme demanded for protection of data from partition intrusion to other partition. This partition scheme is more flexible than the single user scheme because it allows several programs to be in memory at the same time. However, it required the entire program to be stored contagiously and in memory from the beginning to the end of its execution, else reject job.

In order to allocate memory space to jobs, the operating system's memory manager must keep a memory table with each memory partition size, its address, its access restrictions and its current

This table is also called **Partition Description Table** (PDT).



Partition	**Partition**		
1D	Starting address	Size	Status
0	OK	100K	Busy
1	100K	200K	Free
2	300K	100K	Busy
3	400k	300K	Busy
4	700K	100K	Free
5	800K	200K	Busy

NB: Size is bound to change depending on the program.

A small job occupying a bigger block lacking some memory space (internal fragmentation) then the external fragmentation which is the vice versa.

When a job terminates the status of its memory partition is changed from busy to free so that an incoming job can be assigned to that partition.

With this partition scheme, large jobs may have a longer **"turnaround time'** due to waiting a larger time for free partitions of sufficient size. On the other hand, if the partition sizes are too big, memory is wasted, due to unoccupied space.

Limitations

- Internal fragmentation – if a process of less size than the partition size is allocated the memory space could be wasted.
- External fragmentation – free partitions whose total size could accommodate a process waiting but the process can't be allowed in and this wastes memory space.

Advantages

- Support of multi-users.

Variable partitions memory management system

❖ Dynamic partitions.

With dynamic partitions, available memory is still kept in contiguous blocks but jobs are given only as much memory as they request. Here memory is not wasted within the partitions, but it doesn't eliminate entirely the problem.

The problem is that whereas this scheme fully utilises memory for the first jobs, new jobs that are not of the same size, don't fully occupy the memory, many free memory blocks, thus wasting memory.

In dynamic partitions, the partitions are not defined at the time of system generation.

❖ Best-fit versus first fit allocation

In this case partitions are allocated on a first-fit or best-fit basis. For both schemes the memory manager organises the memory list of the free and used partitions (free/busy).

The best-fit allocation method keeps the free/busy lists in order by size, smallest to largest. Processing time is long.

Best-fit scans the *whole* list, and chooses the smallest free area that is big enough.

The intuition is that this will be more efficient and only waste a little bit each time. However, this is wrong: first-fit is usually better, because best-fit tends to create multiple small and unusable fragments

The first-fit method keeps the free/busy lists organised by memory locations, low-order memory to high-order memory. Each has some advantage over the other.

First-fit-scans the list from its beginning and chooses the first free area that is big enough.

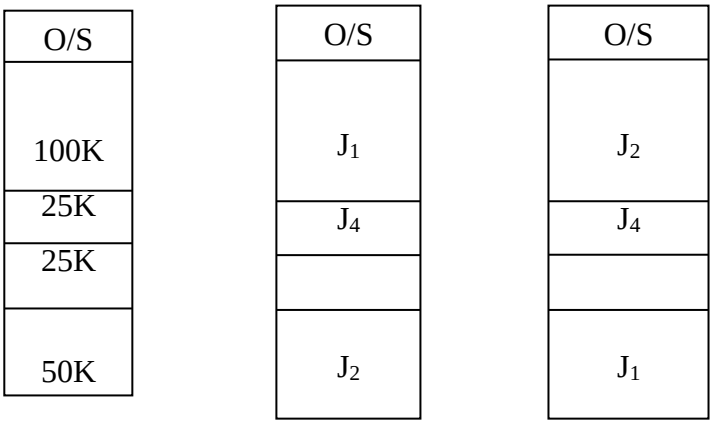
Best-fit usually makes the best use of memory space. First-fit is faster in making the allocation.

Next-fit is a variant of first-fit. The problem is that first-fit tends to create more fragmentation near the beginning of the list, which causes it to take more time to reach reasonably large free areas. Next-fit solves this by starting the next search from the point where it made the previous allocation. Only when the end of the list is reached does it start again from the beginning.

Example of **Best-fit versus First-fit**

Job/process/program

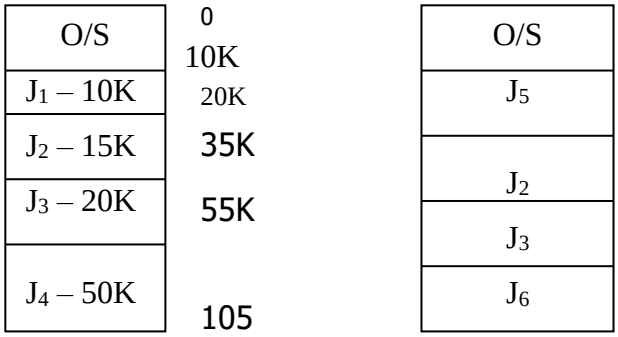
J_1 – 30k
J_2 – 50K
J_3 – 35K
J_4 – 25K



External fragment is available because the job was not able to be allocated thus there is space in both the first-fit and best-fit since one job is left out. Eg. J_3 is left out in First-fit and J_3 is also left out in Best-fit.
Then J_3 can come in when J_2 or J_1 has been completed.

Note: External fragmentation is when a job is not taken over yet there is enough space that is wasted that would have occupied it.

Example 2
Job list
J_1 – 10K
J_2 – 15K
J_3 – 20K
J_4 – 50K
J_5 – 5K
J_6 – 30K
J_7 – 10K
J_8 – 30K



Assuming J1 and J4 have ended, then there is de-allocation

Assuming J3 has ended then J7 will occupy it.

Note:

There is no external fragmentation in the non-contiguous memory allocation. Then with internal fragmentation is example being looked at as below;

Assume 1 word is allocated in the last page program size

One solution to the problem of fragmentation is compaction: relocate the various segments so as to accumulate all the free space in one place. This will hopefully be large enough for additional

allocations. However, this suffers from considerable overhead. A better solution is to use paging rather than contiguous allocation.

Process Management

A Program does nothing unless its instructions are executed by a CPU. A program in execution is called a process. In order to accomplish its task, process needs the computer resources.

There may exist more than one process in the system which may require the same resource at the same time. Therefore, the operating system has to manage all the processes and the resources in a convenient and efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

In single-user systems the processor is busy only when the user is executing a job – at all other times it is idle. Processor management in this environment is simple. However, when there are many users with many jobs on the system (this is known as a multiprogramming environment), the processor must be allocated to each job in a fair and efficient manner, and this can be a complex task.

A processor which is also known as the Central Processing Unit (CPU) is the part of a machine which does the calculations and executes the program for example a simple mathematical calculation is a process.

A job, or program in an operating systems environment, is a unit of work that's submitted by the user.

Multiprogramming requires that the processor be allocated to each job or to each process for a period of time and de-allocated at an appropriate moment.

A single processor can be shared by several jobs, or several processes, but if, and only if, the operating system has a scheduling policy as well as a scheduling algorithm to determine when to stop working on one job and proceed to another.

The operating system is responsible for the following activities in connection with Process Management

i. Scheduling processes and threads on the CPUs.
ii. Creating and deleting both user and system processes.
iii. Suspending and resuming processes.
iv. Providing mechanisms for process synchronization.
v. Providing mechanisms for process communication.

Attributes of a process

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them. This is also called context of the process. Attributes which are stored in the PCB are described below.

Process Control Blocks (PCB)

PCB gives key information about a process when the operating system switches the attention of processor among processes. It uses the save areas in the PCB to hold information that needs to restart each process when the process next gets the CPU. The PCB is the entity that defines a process of the operating system.

Process – ID
Process state
Process priority
Register Save Area for PC, IR, SP …
Pointers to process's memory
Pointers to other resources
List of open files
Accounting information
Other information if required
Pointer to other PCBs

1. **Process priority**

Some processes are urgently required (set either higher or lower priority). Priority can be set externally by the user or system manager or internally by the OS depending on various parameters or by a combination of both.

2. **Pointers to the process's memory**

This gives direct or indirect addresses or pointers to the locations where the process image resides in the memory. E.g. in paging system, it could point towards the Page Map Table (PMT) which in turn point to the physical memory (indirect).

3. **Pointer to other resources**

This gives pointers to other data structures maintained for that process.

4. **Accounting Information**

This gives account of the usage of resources such as CPU time, connect time, disk I/O used especially in a data centre environment or cost centre environment where different users are to be charged for their system usage.

5. **List of open files**

This can be used by the O/S to close all open files not closed by a process on termination.

6. **Pointer to other PCB**

This gives the address of the next PCB (e.g PCB number) within a specific category i.e the O/S maintains a list of ready processes.

7. **Register Save Area**

This is needed to save all the CPU Registers at the context switch

Job Scheduling Versus Process Scheduling

The processor manager is a composite of two sub managers:

i. Job scheduler which is in charge of job scheduling.

ii. Process scheduler which is in charge of process scheduling.

A **Job scheduler** is also called the high-level scheduler, which is only concerned with selecting jobs from a queue of incoming jobs and placing them in the process queue. The job scheduler's goal is to put the jobs in a sequence that will use all of the system's resources as fully as possible. It keeps most of the components of the computer systems busy most of the time.

Process scheduler

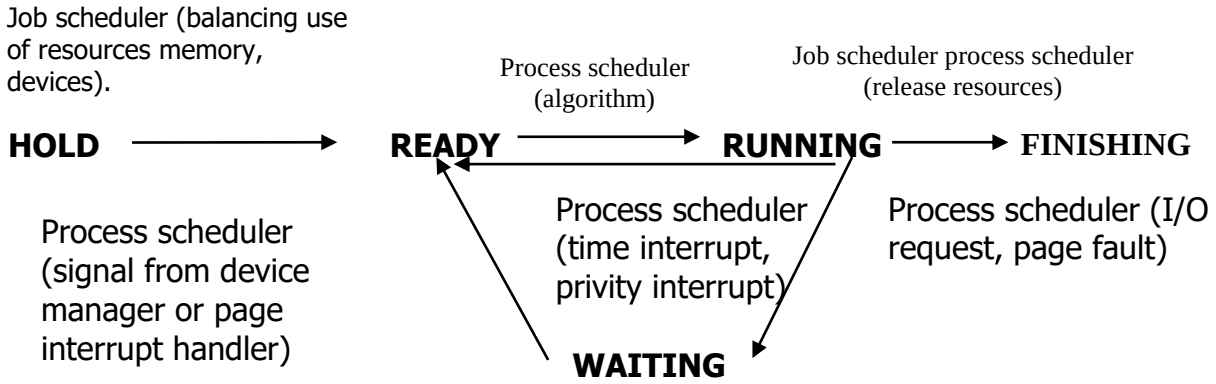
Once a job has been placed on the READY queue by the job scheduler, it's the process scheduler that takes over. It determines which jobs will get the CPU, when, and for how long, it also decides when processing should be interrupted, determines which queue the job should be moved to during its execution, and recognises when a job has concluded and should be terminated.

The process scheduler is the low-level scheduler that assigns the CPU to execute the processes of those jobs **I/O-bound** jobs (such as printing a series of documents) have many brief CPU cycles and long I/O cycles, whereas **CPU-bound** jobs (such s finding the first 300 prime numbers) have long CPU cycles and shorter I/O cycles.

In a single-user environment, there is no distinction made between job and process scheduling because only one job is active in the system at any given time. So the CPU and all other resources are dedicated to that job until it is completed.

Job and Process Status

As a job moves through the system its always in one of five statuses (or at least three). There are HOLD, READY, RUNNING, FINISHED AND WAITING.



i. The transition from HOLD to READY the job scheduler according to some predefined policy, like availability of enough main memory and any request devices

ii. The transition from READY to RUNNING is handled by the process scheduler according o some predefined algorithm ie. FCFS, SJN, priority scheduling, SRT, or round robin.

iii. The transition from RUNNING to READY is handled by the process scheduler according to some predefined time limit, or other criterion, i.e. a priority interrupt.

iv. The transition from WAITING to READY is handled by the process scheduler and is initiated by a signal from the I/O device manager that the I/O request has been satisfied and the job can continue.

v. Eventually, the transition from RUNNING to FINISHED is initiated by the process scheduler or the job scheduler.

A **process** is the basic unit for CPU scheduling in an operating system. It is an execution stream in the context of a particular process state

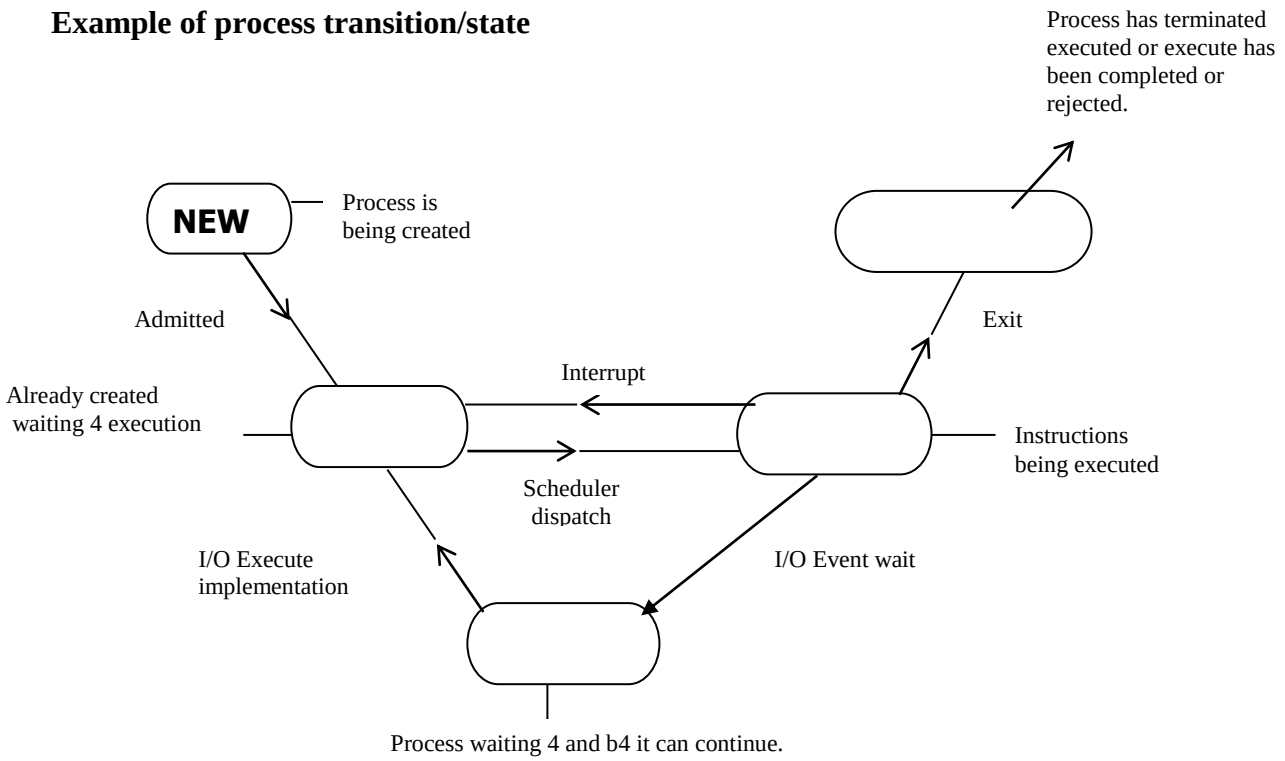
A process may be independent of other processes in the system or may interact with some other processes. An execution stream is a sequence of instructions. A process is an execution stream in the context of a particular process state. An execution stream is a sequence of instructions. Process state determines the effect of the instructions. Process is a key OS abstraction that users see - the environment you interact with when you use a computer is built up out of processes.

Differences between a program and a process.

Program	**process**
It doesn't compete for computing resources like the CPU or memory.	A process does
It can exist on paper or reside on a disk; it may be Compiled or tested, but it still doesn't compete for CPU, RAM and other resources.	It doesn't do any of those done by the program.

Process Transition/State

Example of process transition/state



Two concepts: uniprogramming and multiprogramming.

Uniprogramming: only one process at a time. Typical example is DOS. Problem: users often wish to perform more than one activity at a time (load a remote file while editing a program, for example), and uniprogramming does not allow this. So DOS and other uniprogrammed systems put in things like memory-resident programs that invoked asynchronously, but still have separation problems. One key problem with DOS is that there is no memory protection - one program may write the memory of another program, causing weird bugs.

Multiprogramming: multiple processes at a time. Typical of UNIX plus all currently envisioned new operating systems. It allows system to separate out activities clearly. However, Multiprogramming introduces the resource sharing problem - which processes get to use the physical resources of the machine and when? One crucial resource: CPU. Standard solution is to use preemptive multitasking - OS runs one process for a while, then takes the CPU away from that process and lets another process run. The OS must save and restore process state. Key issue is fairness and must ensure that all processes get their fair share of the CPU.

Process Scheduling

In a multiprogramming environment there are usually more jobs to be executed than could possibly be run at one time. The following criteria should be followed when developing an operating system in order to efficiently mange the processor:

i. Maximise throughput i.e. as many jobs as possible in a given time.
ii. Minimise response time
iii. Minimise turnaround time by moving entire jobs in and out of the system quickly.
iv. Minimise waiting time by moving jobs out of the READY queue as quickly as possible.
v. Maximise CPU efficiency by keeping the CPU busy 100% of the time.

The operating system must choose one of the processes in the list of ready-to-launch processes whenever the CPU gets idle. A temporary (CPU) scheduler does the selection. The Scheduler choose one of the ready-to-start memory processes to get the CPU. Before, going to the Types of CPU Scheduling Algorithms, we are going to learn about the Basic Terminologies which are to be followed and used in the CPU Scheduling Algorithms.

Process ID

The Process ID is the first Thing is to be written while solving the problem. The Process ID acts like the name of the process. It is usually represented with numbers or P letter with numbers

Example:

```
0, 1, 2, 3, . . . . . .
P0, P1, P2, P3 . . . . . .
```

Usually, we start the naming of the process from zero. We can also start the numbering from number 1 also. It is our interest

Arrival Time

The time which is required for the Process to enter the ready queue or the time when the Process is ready to be executed by the CPU. This Arrival Time can be represented as AT in short form. The Arrival Times is always positive or also zero.

Burst Time

The Time Slot which the Process requires to complete the Process is known as the Burst Time. The Burst Time can be represented as BT in short form. The Burst Times of a process is always greater than zero.

Completion Time

The Total Time required by the CPU to complete the process is known as Completion Time. The Completion Time can be represented as CT in short form. The Completion will always be greater than zero.

Turn Around Time

The time taken by the CPU since the Process has been ready to execute or since the process is in Ready Queue is known as Turn Around Time. The Turn Around Time can be calculated with the help of Completion Time and Arrival Time. The Turn Around Time can be represented as TAT in short form.

The Turn Around Time is the difference of Completion Time and Arrival Time.

Formula

```
TAT = CT - AT
```

Here, CT is Completion Time and AT is Arrival Time.

Waiting Time

The time the Process has been waiting to complete its process since the assignment of process for completion is known as Waiting Time. The Waiting Time can be represented as WT in short form.

The Waiting Time can be calculated with the help of Turn Around Time and Burst Time.

The Waiting Time is the difference between Turn Around Time and Burst Time

Formula

```
WT = TAT - BT
```

Ready Queue

The Queue where all the processes are stored until the execution of the previous process. This ready queue is very important because there would be confusion in CPU when two same kinds of processes are being executed at the same time.

Then, in these kinds of conditions the ready queue comes into place and then, the its duty is fulfilled.

Gantt Chart

It is the place where all the already executed processes are stored. This is very useful for calculating Waiting Time, Completion Time, Turn Around Time.

Modes in CPU Scheduling Algorithms

There are two modes in CPU Scheduling Algorithms. They are:

1. Pre-emptive Approach
2. Non Pre-emptive Approach

A scheduling policy that interrupts the processing of a job and transfers the CPU to another job is called a **pre-emptive scheduling policy** and it is widely used in time-sharing environments.

A **non-pre-emptive scheduling policy** functions without external interrupts. Once a job captures the processor and begins execution, it remains in the RUNNING state uninterrupted until it is finished or issues on I/O request (natural wait).

The CPU Scheduling is the process by which a process is executed by the using the resources of the CPU. The process also can wait due to the absence or unavailability of the resources. These processes make the complete use of Central Processing Unit.

The process scheduler relies on a process scheduling algorithm, based on a specific policy, to allocate the CPU and move jobs through the system.

Types of CPU Scheduling Algorithms

i. First Come First Serve
ii. Shortest Job First
iii. Priority Scheduling
iv. Round Robin Scheduling

1. First Come First Served (FCFS)

Other names of this algorithm are:

- First-In-First-Out (FIFO)
- Run-to-Completion
- Run-Until-Done

Perhaps, First-Come-First-Served algorithm is the simplest scheduling algorithm is the simplest scheduling algorithm. Processes are dispatched according to their arrival time on the ready queue. Being a non-preemptive discipline, once a process has a CPU, it runs to completion. The FCFS scheduling is fair in the formal sense or human sense of fairness but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait.

This is a non-pre-emptive scheduling algorithm that handles jobs according to their arrival time. The earlier they arrive; the sooner they are served. This algorithm uses a FIFO type of queue. This algorithm is fine for most batch systems, but it is unacceptable for interactive systems because interactive users expect quick response times.

With FCFS, a new job is linked to the READY queue and it is removed from the front of the queue when the processor becomes available. In a FCFS system, there are no WAIT queues. Each job is run to completion.

Characteristics of FCFS (First Come First Serve):

i. First Come First Serve can follow or can be executed in Pre-emptive Approach or Non Pre-emptive Approach
ii. The Process which enters the Ready Queue is executed First. So, we say that FCFS follows First in First Out Approach.
iii. First Come First Come First Serve is only executed when the Arrival Time (AT) is greater than or equal to the Time which is at present.

Advantages

i. Very easy to perform by the CPU
ii. Follows FIFO Queue Approach

Disadvantages

i. First Come First Serve is not very efficient.
ii. First Come First Serve suffers because of Convoy Effect.

Examples

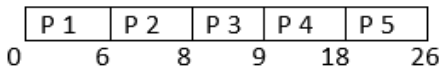
Process ID	Process Name	Arrival Time	Burst Time
P 1	A	0	6
P 2	B	2	2
P 3	C	3	1
P 4	D	4	9
P 5	E	5	8

Now, we are going to apply FCFS (First Come First Serve) CPU Scheduling Algorithm for the above problem.

In FCFS, there is an exception that Arrival Times are not removed from the Completion Time. Here, in First Come First Serve the basic formulas do not work. Only the basic formulas work.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P 1	0	6	6	6	0
P 2	2	2	8	8	6
P 3	3	1	9	9	8
P4	4	9	18	18	9
P 5	5	8	26	26	18

Gantt Chart



Average Completion Time = The Total Sum of Completion Times which is divided by the total number of processes is known as Average Completion Time.

Average Completion Time =(CT1 + CT2 + CT3 + ... + CTn) / n

Average Completion Time

Average CT = (6 + 8 +9 + 18 + 26) / 5
Average CT = 67 / 5
Average CT = 13. 4

Average Turn Around Time = The Total Sum of Turn Around Times which is divided by the total number of processes is known as Average Turn Around Time.

Average Turn Around Time = (TAT1 + TAT2 + TAT3 + ... + TATn) / n

Average Turn Around Time

Average TAT = (6 + 8 + 9 +18 +26) / 5
Average TAT = 13.4

Average Waiting Time = The Total Sum of Waiting Times which is divided by the total number of processes is known as Average Waiting Time.

Average Waiting Time = (WT1 + WT2 + WT3 + ... + WTn) / n

Average Waiting Time

```
Average WT = ( 0 + 6 + 8 + 9 + 18 ) / 5
Average WT = 41 / 5
Average WT = 8.2
```

Examples:

```
S. No   ProcessID     Process Name      Arrival Time    Burst Time
---  ------  -------  -------  -------
1       P 1         A           1          79
2       P 2         B           0          2
3       P 3         C           1          3
4       P 4         D           0          1
5       P 5         E           2          25
```

S. No	Process Id	Process Name	Arrival Time (AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time (TAT)	Waiting Time (WT)	Response Time (RT)
1	P 1	A	1	79	79	78	0	0
2	P 2	B	0	2	81	81	79	79
3	P 3	C	1	3	84	83	81	81
4	P 4	D	0	1	85	85	84	84
5	P 5	E	2	25	110	108	85	85
6	P 6	F	3	3	113	110	110	110

The Average Completion Time is:

Average CT = (79 + 81 + 84 + 85 + 110 + 113) / 6

Average CT = 92

The Average Waiting Time is:

Average WT = (0 + 79 + 81 + 84 + 85 + 110) /6

Average WT = 73.1666667

The Average Turn Around Time is:

Average TAT = (78 + 81 + 83 + 85 + 108 +110) / 6

Average TAT = 90.833334

The biggest problem in this is higher Completion Time, higher Waiting Time, higher Turn Around Time, lower efficiency.

FCFS is more predictable than most of other schemes since it offers time. FCFS scheme is not useful in scheduling interactive users because it cannot guarantee good response time. The code for FCFS scheduling is simple to write and understand. One of the major drawback of this scheme is that the average time is often quite long.

The First-Come-First-Served algorithm is rarely used as a master scheme in modern operating systems but it is often embedded within other schemes.

The problem which we found in the above solution can be resolved by using a new CPU Scheduling Techniques named Shortest Job First (SJF).

Shortest Job First CPU Scheduling Algorithm

This is another type of CPU Scheduling Algorithms. Here, in this CPU Scheduling Algorithm we are going to learn how CPU is going to allot resources to the certain process.

The Shortest Job is heavily dependent on the Burst Times. Every CPU Scheduling Algorithm is basically dependent on the Arrival Times. Here, in this Shortest Job First CPU Scheduling Algorithm, the CPU allots its resources to the process which is in ready queue and the process which has least Burst Time.

If we face a condition where two processes are present in the Ready Queue and their Burst Times are same, we can choose any process for execution. In actual Operating Systems, if we face the same problem then sequential allocation of resources takes place.

Shortest Job First can be called as SJF in short form.

Characteristics:

i. SJF (Shortest Job First) has the least average waiting time. This is because all the heavy processes are executed at the last. So, because of this reason all the very small, small processes are executed first and prevent starvation of small processes.
ii. It is used as a measure of time to do each activity.
iii. If shorter processes continue to be produced, hunger might result. The idea of aging can be used to overcome this issue.
iv. Shortest Job can be executed in Pre-emptive and also non pre-emptive way also.

Advantages

i. SJF is used because it has the least average waiting time than the other CPU Scheduling Algorithms

ii. SJF can be termed or can be called as long term CPU scheduling algorithm.

Disadvantages

i. Starvation is one of the negative traits Shortest Job First CPU Scheduling Algorithm exhibits.

ii. Often, it becomes difficult to forecast how long the next CPU request will take

Examples for Shortest Job First

Process ID	Arrival Time	Burst Time
P0	1	3
P1	2	6
P2	1	2
P3	3	7
P4	2	4
P5	5	5

Now, we are going to solve this problem in both pre emptive and non pre emptive way

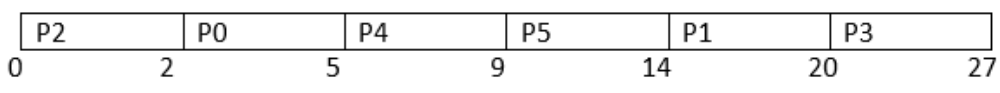
We will first solve the problem in non-pre-emptive way.

In Non pre-emptive way, the process is executed until it is completed.

Non Pre-Emptive Shortest Job First CPU Scheduling

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time TAT = CT - AT	Waiting Time WT = CT - BT
P0	1	3	5	4	1
P1	2	6	20	18	12
P2	0	2	2	2	0
P3	3	7	27	24	17
P4	2	4	9	7	4
P5	5	5	14	10	5

Gantt Chart:



Now let us find out Average Completion Time, Average Turn Around Time, Average Waiting Time.

Average Completion Time:

```
Average Completion Time = ( 5 + 20 + 2 + 27 + 9 + 14 ) / 6
Average Completion Time = 77/6
Average Completion Time = 12.833
```

Average Waiting Time:

```
Average Waiting Time = ( 1 + 12 + 17 + 0 + 5 + 4 ) / 6
Average Waiting Time = 37 / 6
Average Waiting Time = 6.666
```

Average Turn Around Time:

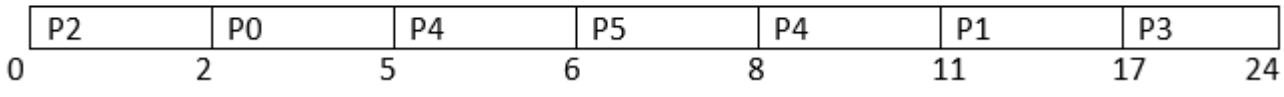
```
Average Turn Around Time = ( 4 +18 + 2 +24 + 7 + 10 ) / 6
Average Turn Around Time = 65 / 6
Average Turn Around Time = 6.166
```

Pre Emptive Approach

In Pre-emptive way, the process is executed when the best possible solution is found**.**

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time TAT = CT - AT	Waiting Time WT = CT - BT
P0	1	3	5	4	1
P1	2	6	17	15	9
P2	0	2	2	2	0
P3	3	7	24	21	14
P4	2	4	11	9	5
P5	6	2	8	2	0

Gantt chart:



After P4 P5 is executed just because of the Pre-Emptive Condition.

Now let us find out Average Completion Time, Average Turn Around Time, Average Waiting Time.

Average Completion Time

i. Average Completion Time = (5 + 17 + 2 + 24 + 11 +8) / 6
ii. Average Completion Time = 67 / 6
iii. Average Completion = 11.166

Average Turn Around Time

i. Average Turn Around Time = (4 +15 + 2 + 21 + 9 + 2) / 6
ii. Average Turn Around Time = 53 / 6
iii. Average Turn Around Time = 8.833

Average Waiting Time

i. Average Waiting Time = (1 + 9 + 0 + 14 + 5 + 0) /6
ii. Average Waiting Time = 29 / 6
iii. Average Waiting Time = 4.833

This is another type of CPU Scheduling Algorithms. Here, in this CPU Scheduling Algorithm we are going to learn how CPU is going to allot resources to the certain process.

The Priority CPU Scheduling is different from the remaining CPU Scheduling Algorithms. Here, each and every process has a certain Priority number.

There are two types of Priority Values.

i. Highest Number is considered as Highest Priority Value
ii. Lowest Number is considered as Lowest Priority Value

Priority for Prevention The priority of a process determines how the CPU Scheduling Algorithm operates, which is a pre-emptive strategy. Since the editor has assigned equal importance to each function in this method, the most crucial steps must come first. The most significant CPU planning algorithm relies on the FCFS (First Come First Serve) approach when there is a conflict, that is, when there are many processors with equal value.

This algorithm gives preferential treatment to important jobs. It allows the programs with the highest priority to be processed first, and they aren't interrupted until their CPU cycles (run times) are completed or a natural wait occurs. If two or more jobs with equal priority are present in the READY queue, the processor is allocated to the one that arrived first (FCFS within priority). With a priority algorithm, jobs are usually linked to one of several READY queues by the job scheduler based on their priority.

Characteristics

i. Priority CPU scheduling organizes tasks according to importance.
ii. When a task with a lower priority is being performed while a task with a higher priority arrives, the task with the lower priority is replaced by the task with the higher priority, and the latter is stopped until the execution is finished.

iii. A process's priority level rises as the allocated number decreases.

Advantages

i. The typical or average waiting time for Priority CPU Scheduling is shorter than First Come First Serve (FCFS).

ii. It is easier to handle Priority CPU scheduling

iii. It is less complex

Disadvantages

The Starvation Problem is one of the Pre-emptive Priority CPU Scheduling Algorithm's most prevalent flaws. Because of this issue, a process must wait a longer period of time to be scheduled into the CPU. The hunger problem or the starvation problem is the name given to this issue.

Examples:

Now, let us explain this problem with the help of an example of Priority Scheduling

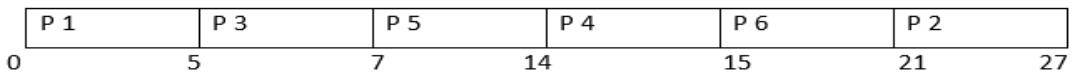
S. No	Process ID	Arrival Time	Burst Time	Priority
1	P 1	0	5	5
2	P 2	1	6	4
3	P 3	2	2	0
4	P 4	3	1	2
5	P 5	4	7	1
6	P 6	4	6	3

Here, in this problem the priority number with highest number is least prioritized. This means 5 has the least priority and 0 has the highest priority.

Solution:

S. No	Process ID	Arrival Time	Burst Time	Priority
1	P 1	0	5	5
2	P 2	1	6	4
3	P 3	2	2	0
4	P 4	3	1	2
5	P 5	4	7	1
6	P 6	4	6	3

Gantt Chart:



Average Completion Time

```
Average Completion Time = ( 5 +27 +7 +15 +14 + 21 ) / 6
Average Completion Time = 89 / 6
Average Completion Time = 14.8333
```

Average Waiting Time

```
Average Waiting Time = ( 0 + 20 + 3 + 11 + 3 + 11 ) / 6
Average Waiting Time = 48 / 6
Average Waiting Time = 7
```

Average Turn Around Time

```
Average Turn Around Time = ( 5 + 26 + 5 + 11  + 10 + 17 ) / 6
Average Turn Around Time = 74 / 6
Average Turn Around Time = 12.333
```

Round Robin CPU Scheduling

Round Robin is a CPU scheduling mechanism those cycles around assigning each task a specific time slot. It is the First come, first served CPU Scheduling technique with pre-emptive mode. The Round Robin CPU algorithm frequently emphasizes the Time-Sharing method.

Round Robin CPU Scheduling Algorithm characteristics include:

i. Because all processes receive a balanced CPU allocation, it is straightforward, simple to use, and starvation-free.

ii. One of the most used techniques for CPU core scheduling. Because the processes are only allowed access to the CPU for a brief period of time, it is seen as pre-emptive.

The benefits of round robin CPU Scheduling:

i. Every process receives an equal amount of CPU time, therefore round robin appears to be equitable.

ii. To the end of the ready queue is added the newly formed process.

Examples:

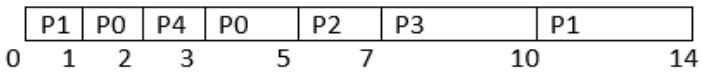
Problem

Process ID	Arrival Time	Burst Time
P0	1	3
P1	0	5
P2	3	2
P3	4	3
P4	2	1

Solution:

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P0	1	3	5	4	1
P1	0	5	14	14	9
P2	3	2	7	4	2
P3	4	3	10	6	3
P4	2	1	3	1	0

Gantt Chart:



Average Completion Time = 7.8

Average Turn Around Time = 4.8

Average Waiting Time = 3

This is all about the CPU Scheduling Algorithms in Operating System

Interrupts

A memory manager can issue page interrupts to accommodate job requests. An interrupt can also occur when the time quantum expires and the processor is de-allocated from the running job and allocated to another one. I/O interrupts are issued when a reader or writer command is issued.

Internal interrupts also called synchronous interrupts, also occur as a direct result of the arithmetic operation or job instruction currently being processed. Illegal arithmetic operations such as the following can generate interrupts:

i. Attempts to device by zero
ii. Floating-point operation guarding an overflow or underflow.
iii. Fixed-point addition or subtraction that causes an arithmetic overflow.

Illegal job instructions such as the following can also generate interrupts:

i. Attempts to access protected or none existent storage locations.
ii. Attempts to use an undefined operation code;
iii. Attempts to make system changes, such as trying to change the size of the time quantum.

The control program that handles the interruption sequences of events is called the interrupt handler. When the operating system detects a non recoverable error, the interrupt handler typically follows this sequence;

i. The type of interrupt is described and stored to be passed on to the user as an error message.
ii. The state of the interrupted process is saved, including the value of the program counter the mode specification, and the contents of all registers.
iii. The interrupt is processed: the error message and state of the interrupted process are sent to the user, program execution is halted, any resources allocated to the job are released and the job exits the system.
iv. The processor resumes normal operation.

Resources Management

A resource can be hardware or software. Examples of hardware resources are plotters CD-ROM readers, CD-ROM recorders, tape drives backup system etc. examples of software resources are records in a database system.

Reusable Resources

- Used by one process at a time and not depleted by that use
- Processes obtain resources that they later release for reuse by other processes i.e. Processors, I/O channels, main and secondary memory, files, databases, and semaphores
- Deadlock occurs if each process holds one resource and requests the other

Consumable Resources

- Created (produced) and destroyed (consumed) by a process i.e. Interrupts, signals, messages, and information in I/O buffers
- Deadlock may occur if a Receive message is blocking
- May take a rare combination of events to cause deadlock

Resources are categorised either as **pre-emptive** or non **pre-emptive**.

A **pre-emptive resource** is one that can be taken away from the process owning it without causing errors e.g. the memory.

A **non pre-emptive resource** is one that can not be taken away from its current owner without causing the computation to fail e.g. a printer.

Most deadlocks involve non pre-emptable resources. Deadlocks that involve pre-emptable resources can usually be resolved by re-allocating resources from one process to another.

The sequence of events required to use a resource is

1) Request the resource
2) Use the resource
3) Release the resource.

Example of Deadlock

Process P

Step	Action
p_0	Request (D)
p_1	Lock (D)
p_2	Request (T)
p_3	Lock (T)
p_4	Perform function
p_5	Unlock (D)
p_6	Unlock (T)

Process Q

Step	Action
q_0	Request (T)
q_1	Lock (T)
q_2	Request (D)
q_3	Lock (D)
q_4	Perform function
q_5	Unlock (T)
q_6	Unlock (D)

Figure 6.4 Example of Two Processes Competing for Reusable Resources