OPERATING SYSTEM

1. What is OS?

Ans. OS is an intermediary between the user of a computer and computer hardware. OS provides an environment for user to execute programs. An OS is a software program that manages computer hardware and runs all time(kernel). OS is responsible for allocation of resources & services, like memory, processors, devices & information.

Major Functionalities of OS:

1. Resource Management: Managing resources when multiple users are accessing the system.

2. Process Management: Tasks like CPU Scheduling, Termination of process.

3. Storage Management: The file system mechanism.

4. Memory Management: Management of primary memory.

5. Privacy Management: Using passwords so that unauthorized applications can't access programs or data.

2. Types of OS.

Ans.

1. Single process operating system

A computer system that allows only one user to use the computer at a given time is known as a single-user system. The goals of such systems are maximizing user convenience and responsiveness, instead of maximizing the utilization of the CPU and peripheral devices.

Example: Windows, Apple Mac OS,etc.

2. Batch-processing system

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs.

Example: Payroll System (Payroll systems manage everything having to do with the process of paying employees and filing employment taxes), Bank Statements

3. Time-Sharing Operating System

The Time Shared Operating System is also known as the Multitasking Operating System.Time-sharing operating systems implements CPU scheduling and multi programming systems which deliver to every user a small piece of operating time.

Example: UNIX, Multics, Linux, Windows 2000 server,etc.

4. Distributed Operating System

Distributed operating system allows distributing of entire systems on the couples of center processors, and it serves on the multiple real time products as well as multiple users.

Example: Windows Server 2003, Windows Server 2008, Windows Server 2012, Ubuntu, Linux(Apache Server),etc.

5. Real Time Operating System

Real time systems are used when strict time requirements are placed on the operation of a processor or the flow of data. These are used to control a device in a dedicated application.

Example: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

6. Network Operating System

Network Operating System is a computer operating system that facilitates to connect and communicate various autonomous computers over a network. An Autonomous computer is an independent computer that has its own local memory, hardware, and O.S. It is self capable to perform operations and processing for a single user. They can either run the same or different O.S.

Example: Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc1

3. What is Process?

Ans. The process is an example of a computer program used. Contains the program code and its current function. Depending on the operating system (OS), a process can be performed with multiple configurations that issue commands simultaneously. Each process has a complete set of variations.

3. What are different sections of the process?

Ans. There are mainly four sections in a process. They are as below:

- Stack: contains local variables, returns address
- Heap: Dynamically allocated memory via malloc, calloc,realloc
- Data: contains global and static variables.
- Code or text: contains code, program counter, and content of processor’s register.

Process Conditions:

- New State: This is the state when the process is just created. It is the first state of a process.
- Ready State: After the creation of the process, when the process is ready for its execution then it goes into the ready state. In a ready state, the process is ready for its execution by the CPU but it is waiting for its turn to come. There can be more than one process in the ready state.
- Ready Suspended State: There can be more than one process in the ready state but due to memory constraint, if the memory is full then some process from the ready state gets placed in the ready suspended state.

- Running State: Amongst the process present in the ready state, the CPU chooses one process amongst them by using some CPU scheduling algorithm. The process will now be executed by the CPU and it is in the running state.
- Waiting or Blocked State: During the execution of the process, the process might require some I/O operation like writing on file or some more priority process might come. In these situations, the running process will have to go into the waiting or blocked state and the other process will come for its execution. So, the process is waiting for something in the waiting state.
- Waiting Suspended State: When the waiting queue of the system becomes full then some of the processes will be sent to the waiting suspended state.
- Terminated State: After the complete execution of the process, the process comes into the terminated state and the information related to this process is deleted.

4. What is the Zombie process? A zombie process is a process that has completed and in the terminated state but has its entry in the process table. It shows that the resources are held by the process and are not free.

Process Control Block(PCB)

Each process is represented in the operating system by a process control block (PCB) also called a task control block. It contains many pieces of information associated with a specific process, including these:

Process: The state may be new, ready, running, and so on

Program counter: It indicates the address of the next instruction to be executed for this program.

CPU registers: These vary in number and type based on architecture. They include accumulators, stack pointers, general-purpose registers, etc.

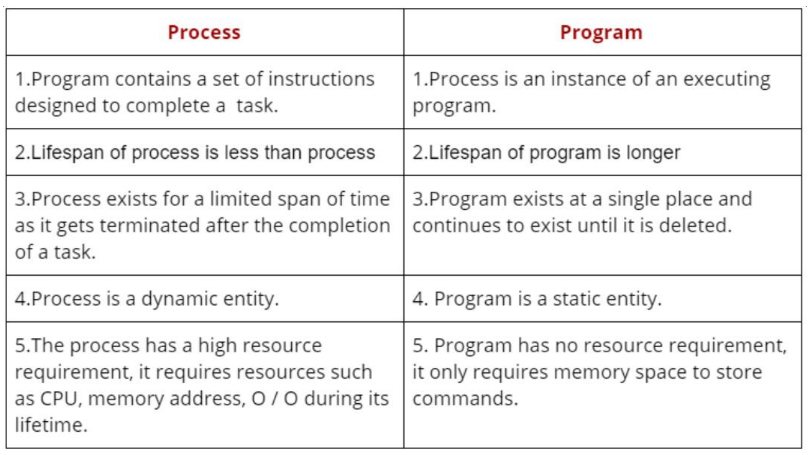
CPU scheduling: This includes process priority, pointers to scheduling queues, and any scheduling parameters.

Memory-management: This includes the value of base and limit registers (protection) and page tables, segment tables depending on memory.

Accounting: It includes the amount of CPU and real-time used, account numbers, process numbers, etc

I/O status information: It includes a list of I/O devices allocated to this process, a list of open files, etc

Process vs Program



Process	Program
1.Program contains a set of instructions designed to complete a task.	1.Process is an instance of an executing program.
2.Lifespan of process is less than process	2.Lifespan of program is longer
3.Process exists for a limited span of time as it gets terminated after the completion of a task.	3.Program exists at a single place and continues to exist until it is deleted.
4.Process is a dynamic entity.	4. Program is a static entity.
5.The process has a high resource requirement, it requires resources such as CPU, memory address, O / O during its lifetime.	5. Program has no resource requirement, it only requires memory space to store commands.

Multiprogramming

A computer running more than one program at a time (like running Excel and Firefox simultaneously). Multiprogramming increases CPU utilization by keeping multiple jobs (code and data) in the memory so that the CPU always has one to execute.

Multitasking

Multitasking has the same meaning of multiprogramming but in a more general sense, as it refers to having multiple (programs, processes, tasks, threads) running at the same time. Multitasking is a logical extension of multiprogramming.

CPU executes multiple tasks by switching among them.

The switching is very fast.

Requires an interactive (hands-on) computer where the user can directly interact with the computer.

Multiprocessing

Multi Processing sometimes refers to executing multiple processes (programs) at the same time.

A system can be both multi programmed by having multiple programs running at the same time and multiprocessing by having more than one physical processor.

THREADS

What is a thread?

Thread is an execution unit that consists of its own program counter, a stack, and a set of registers. Threads are also known as Lightweight processes. Threads are a popular way to improve the application through parallelism. The CPU switches rapidly back and forth among the threads giving the illusion that the threads are running in parallel. As each thread has its own independent resource for process execution, multiple processes can be executed parallelly by increasing the number of threads.

Methods of thread: start()-to start thread execution, getName(), setpoint()-to set priority, yield()-pause execution.

Thread Synchronization: In this process, when the thread gets inside the synchronized block, then it becomes unreachable for others, and no other thread can call that method.

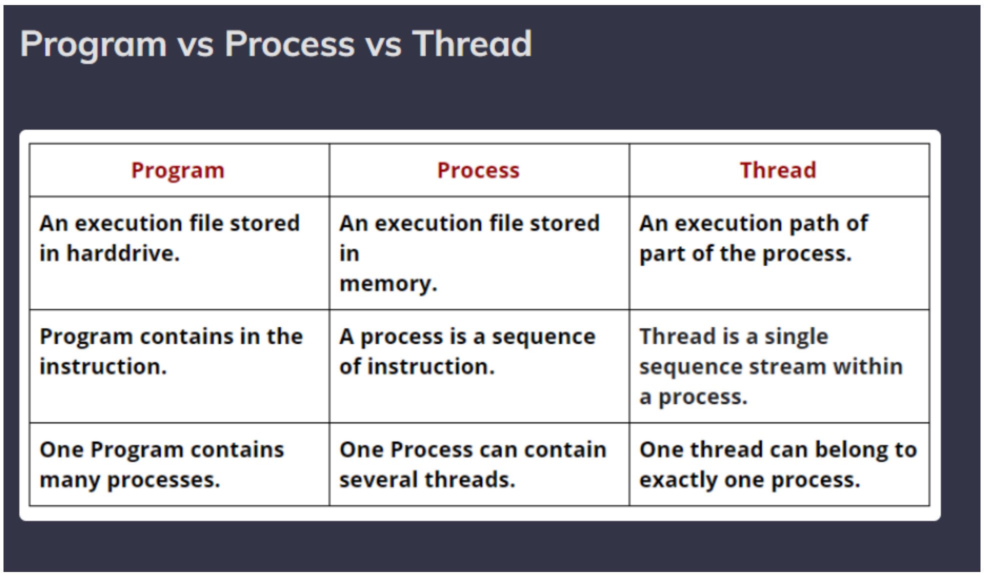
Q. What is Livelock?

Ans. When all the threads are in a blocked state and execution is stopped due to resource unavailability, then that situation is termed as livelock.

Types of Thread

- User threads are above the kernel and without kernel support. These are the threads that application programmers use in their programs.

- Kernel threads are supported within the kernel of the OS itself. All modern operating systems support kernel-level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously.

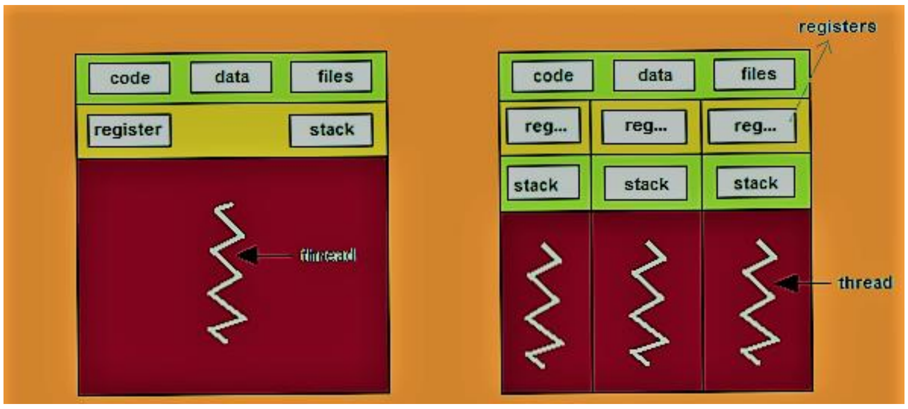


Program vs Process vs Thread

Program	Process	Thread
An execution file stored in harddrive.	An execution file stored in memory.	An execution path of part of the process.
Program contains in the instruction.	A process is a sequence of instruction.	Thread is a single sequence stream within a process.
One Program contains many processes.	One Process can contain several threads.	One thread can belong to exactly one process.

MULTITHREADING

Multithreading is a phenomenon of executing multiple threads at the same time. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc.

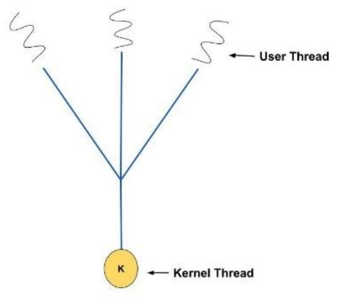


Multithreading Models

The user threads must be mapped to kernel threads, by one of the following strategies:

Many to One Model

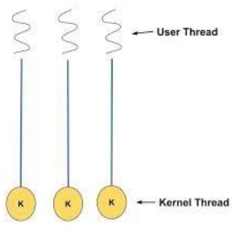
As the name suggests there is many to one relationship between threads. Here, multiple user threads are associated or mapped with one kernel thread. The thread management is done on the user level so it is more efficient.



Many-to-One Model

One to One Model

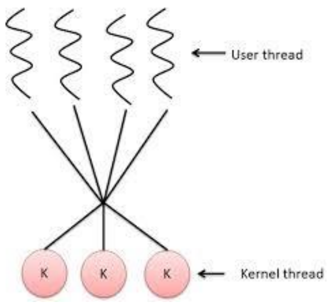
The one to one model creates a separate kernel thread to handle each and every user thread.Most implementations of this model place a limit on how many threads can be created.Linux and Windows from 95 to XP implement the one-to-one model for threads.



One-to-One Model

Many to Many Model

The many to many model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models.Blocking the kernel system calls does not block the entire process.



Q. Multithreading vs Multiprocessing?

- A multiprocessing system has more than two processors whereas Multithreading is a program execution technique that allows a single process to have multiple code segments
- Multiprocessing improves the reliability of the system while in the multithreading process, each thread runs parallel to each other.

Kernel

A Kernel is the central component of an operating system that manages operations of computers and hardware. It basically manages operations of memory and CPU time. It is the core component of an operating system. Kernel acts as a bridge between applications and data processing performed at the hardware level using inter-process communication and system calls.

Main functions of Kernel:

- Process management

- Device management

- Memory management

- Interrupt handling

- I/O communication

- File system management

PROCESS SCHEDULING

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

Pre-emptive scheduling

In pre-emptive scheduling, a process can be forced to leave the CPU and switch to the ready queue.

Non-preemptive scheduling

In non-preemptive scheduling or cooperating scheduling, a process keeps the CPU until it terminates or switches to the waiting state. Some machines support non-preemptive scheduling only.

Q. What is the dispatcher and what is dispatcher latency?

Ans. The dispatcher is a CPU-scheduling component that gives control of the CPU to the process selected by the short-term scheduler.

The function involves the following:

*Switching Context

*Switching to user mode

*Jumping to the proper location in the user program to restart the program

Dispatcher Latency is the time taken by the dispatcher to stop one process and start another running.

FCFS(First-come-first-serve) scheduling

The process that asks for the CPU first is given to the CPU first. The implementation of the FCFS policy is easily handled with the FIFO queue. The average waiting time under the FCFS policy, however, is often quite long.

It is non-preemptive.

It has a high average waiting time.

Q. What is Convoy Effect?

Ans. Convoy Effect is a situation where many processes, which need to use a resource for a short time, are blocked by one process holding that resource for a long time.

This essentially leads to poor utilization of resources and hence poor performance.

Shortest-Job-First Scheduling

This algorithm is associated with each process length of the next CPU storage. When the CPU is available, it is given a process with the next minimum CPU bust. If the two processes are the similar length for the next CPU explosion, the FCFS configuration is used to break the tire.

If a new process arrives with CPU burst length less than the remaining time of the current executing process, preempt, his scheme is known as the Shortest-Remaining-Time-First (SRTF) or pre-emptive SJF.

SJF is optimal – gives a minimum average waiting time for a given set of processes.

Q. What is the main problem with the shortest job scheduling and what is its solution?

Ans. The main problem with the shortest job first algorithm is starvation (a process do). If there is a steady supply of short processes, the long process may never get the chance to be executed by the processor. There are a variety of scheduling algorithms proposed in the past to solve the issue of starvation of SJF.

Priority Scheduling

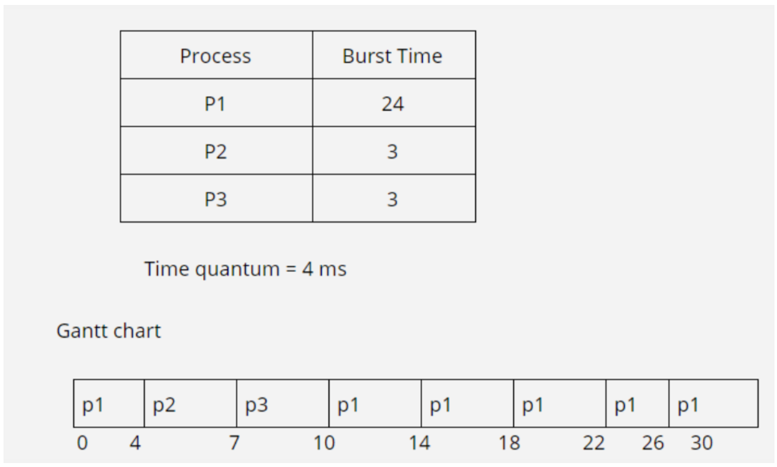
CPU is allocated to the particular process with the highest priority.

Q. What is the major problem with priority scheduling algorithms?

Ans. A major problem with priority scheduling is indefinite blocking or starvation. A solution to the problem of indefinite blockage of the low-priority process is aging. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long period of time. Aging is used to ensure that jobs with lower priority will eventually complete their execution.

Round Robin

The round-robin (RR) scheduling algorithm is designed especially for time-sharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time called a time quantum (or time slice) is defined.



Process	Burst Time
P1	24
P2	3
P3	3

Time quantum = 4 ms

Gantt chart

p1	p2	p3	p1	p1	p1	p1	p1

0 4 7 10 14 18 22 26 30

Multilevel Queue Scheduling

A multilevel queue-scheduling algorithm partitions the ready queue into several separate queues. The processes are completely assigned to one queue, generally based on some unique property of the process, such as memory size, process priority, or process type.

Five queues:

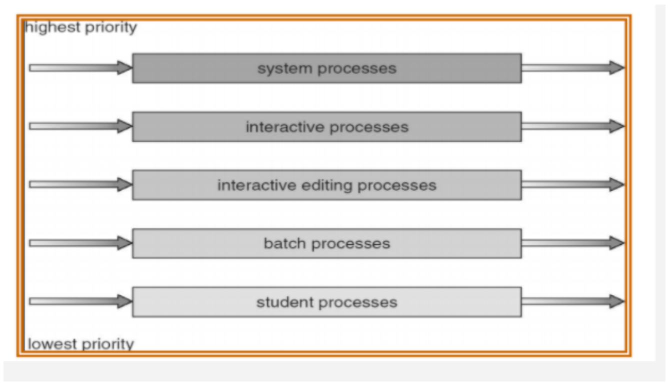
1. System processes

2. Interactive processes

3. Interactive editing processes

4. Batch processes

5. Student processes



Q. What is critical-section problem?

Ans. Critical Section is the part of a program which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device.

The important feature of the system is that, when one process is executing in its critical sections, no other processes are allowed to execute in its critical sections at the same time.

Q. How can we solve critical-section problem with Semaphores?

Ans. Semaphore is a type of flexible or non-flexible data that is used to control access to common shared resources.

A semaphore is an object that consists of a counter, a waiting list of processes, and two methods: signal and wait. All the modifications to the integer value of the semaphore in the wait() and signal() atomic operations must be executed indivisibly. i.e. when one process changes the semaphore value, no other process will change the same semaphore value simultaneously.

When the count for the semaphore is 0 means that all the resources are being used by some processes. Otherwise, resources are available for the processes to allocate. When a process is currently using a resource means that it blocks the resource until the count becomes > 0.

For example, let us consider two processes P1 and P2, and a semaphore initialized with a value of 1. The value of the semaphore becomes 0 if the process P1 enters the critical section. If the process P2 intends to enter the critical section then the value of the semaphore has to be greater than 0, until that the process has to wait. This is only possible when P1 completes the critical section and calls the Signal operation on the semaphore. Mutual exclusion is obtained this way.

Counting semaphore

The value of the Counting Semaphore can range over an unrestricted domain. Counting Semaphores are used to control the access of given resources each of which consists of some finite no. of instances. This counting semaphore is initialized to the number of resources available.

Binary semaphore

The value of the Binary Semaphore can range between 0 and 1 only.

In some systems, the Binary Semaphore is called Mutex locks, because they are locks to provide mutual exclusion. We can use the Binary Semaphore to deal with critical section problems for multiple processes.

Q. What is difference between Semaphore and Mutex?

Ans. Semaphore supports wait and signal operations modification, whereas Mutex is only modified by the process that may request or release a resource. Semaphore value is modified using wait () and signal() operations, on the other hand, Mutex operations are locked or unlocked. Mutex is lightweight and faster than semaphore. Hence, Mutex is even faster.

DEADLOCK

In a multiprogramming system, several processes may compete for a finite number of resources. A process requests resources, and if the resources are not available at the time then the process enters the waiting state. Sometimes, a process will wait indefinitely because the resources it has requested are being held by other similar waiting processes.

Deadlock is a state in which two or more processes are waiting indefinitely because the resources they have requested are being held by one another.

4 Essential Conditions for Deadlock

Mutual Exclusion

The resources involved must be un-shareable. Every resource is either currently allocated to exactly one processor it is available. (Two processes cannot simultaneously control the same resource).

Hold & Weight

There must exist a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by other processes.

No Pre-emption Condition

If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released implicitly. Then the preempted resources are added to the list of resources for which the process is waiting.

Circular Wait

In circular wait, a chain of processes exists in which each process waits for one or more resources held by the next process in the chain.

Q. Explain Banker's algorithm?

Ans. Used for Deadlock avoidance.

A new task must declare the maximum number of instances of each resource type that it may need. This number should not exceed the total number of instances of that resource type in the system.

When a process requests a set of resources, the system must determine whether allocating these resources will leave the system in a safe state. If yes, then the resources may be allocated to the process. If not, then the process must wait till other processes release enough resources.

Q. What is Resource Pre-emption algorithm for Deadlock avoidance?

To eliminate deadlocks using resource preemption, preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.

There are 3 methods to eliminate the deadlocks using resource preemption. These are

a) SELECTING A VICTIM: Select a victim resource from the deadlock state and preempt that one.

b) ROLLBACK: If a resource from a process is preempted, what should be done with that process. The process must be rollbacked to some safe state and restart from that state.

c) STARVATION: It must be guaranteed that resources will not always be preempted from the same process to avoid starvation problems.

Memory Management

Logical & Physical address space

The logical address is the one that is generated by the CPU, also referred to as the virtual address. CPU also referred to as virtual address. The program perceives this address space. program perceives this address space. Logical address space is the set of all logical addresses generated by a program.

A physical address is an actual address understood by computer hardware i.e., memory understood by computer hardware i.e., memory unit. Logical to physical address translation unit. Physical address space is the set of all physical addresses generated by a program.

Static Relocation:

At load time, the OS adjusts the addresses in a process to reflect its position in memory.

This method is a slow process because it involves software translation. It is used only once before the initial loading of the program.

Once a process is assigned a place in memory and starts executing it, the OS cannot move it.

Dynamic Relocation:

Hardware adds relocation register (base) to virtual address to get a physical address. hardware compares address with limit register (address must be less than base).

If the test fails, the processor takes an address trap and ignores the physical address.

Relocation Register

Relocation register is a special register in the CPU used for program relocation means mapping of logical addresses used by the program to physical addresses of the system’s main memory.

BEST FIT

The best fit memory allocation method the memory is traversed until a suitable empty block is found. In this method, the memory wastage is minimal as the approach allocates the memory blocks with minimum memory wastage.

FIRST FIT

In this method, whichever partition has the memory space equal or more than the demanded memory by the process, is allocated to the process at the first attempt during memory traversing. This method continues until all the processes are allocated free memory blocks or no correct free memory block is left that can be assigned to a process.

WORST FIT

The worst fit approach is directly opposite to the best-fit approach. In this method, the CPU searches for a memory block which is greater to the memory in demand. In fact, it searches for an empty memory block which is exceptionally bigger than the demanded memory. The approach is known as the worst fit method as it causes the maximum amount of memory wastage in the memory.

Storage Allocation

Storage allocation methods in memory management

The Storage allocation can be of two types:

(i) Contiguous storage allocation.

(ii) Non-contiguous storage allocation.

Contiguous Storage Allocation

Contiguous storage allocation implies that a program’s data and instructions are assumed to occupy a single contiguous memory area.

Types of Contiguous Storage Allocation:

1. Fixed-partition contiguous storage allocation

The processes with small address space use small partitions and processes with large address space use large partitions. This is known as fixed partition contiguous storage allocation.

2. Variable - partition contiguous storage allocation

This notion is derived from the parking vehicles on the sides of streets where the one who manages to enter will get the space. Two vehicles can leave a space between them that cannot be used by any vehicle. This means that whenever a process needs memory, a search for the space needed by it, is done. If contiguous space is available to accommodate that process, then the process is loaded into memory.

Problem with Contiguous Storage Allocation:

External Fragmentation

This phenomenon of entering and leaving the memory can cause the formation of unusable memory holes (like the unused space between two vehicles). This is known as External Fragmentation.

Non-Contiguous Storage Allocation

To resolve the problem of external fragmentation and to enhance the degree of multiprogramming to a greater extent, it was decided to sacrifice the simplicity of allocating contiguous memory to every process. The process is divided into parts so that the same process can exist in different places of the storage.

Types of Non-Contiguous Storage allocation:

1. Paging

2. Segmentation

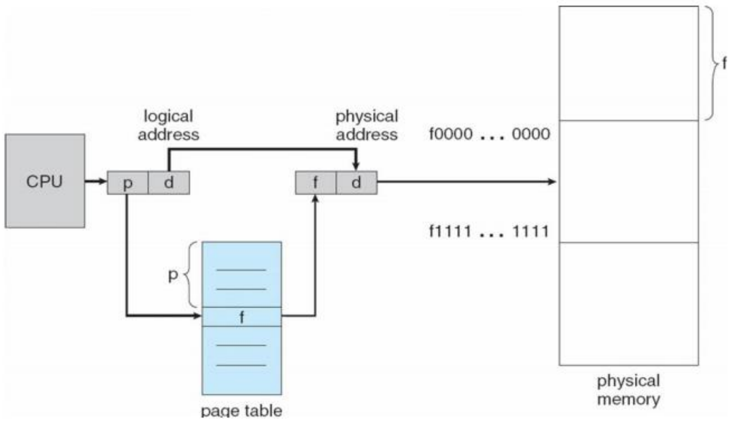
Paging

Physical memory is divided into fixed-size- blocks called FRAMES. (size is the power of 2, for example, 512 bytes)

Logical memory is divided into blocks of the same size called PAGES.

The size of a page is the same as that of a frame. The key idea of this method is to place the pages of a process into the available frames of memory, whenever, this process is to be executed. The address mapping is done by Page table.

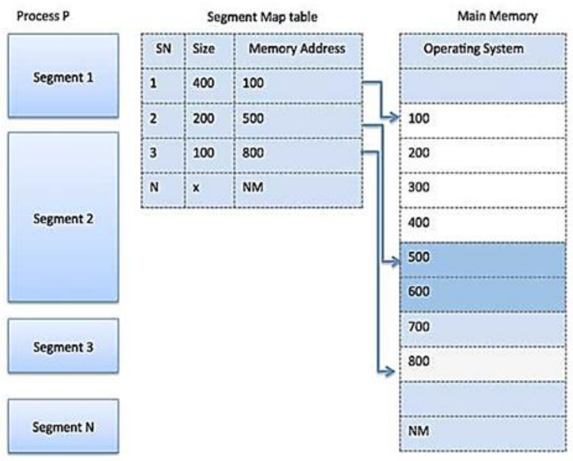
The main advantage of paging is that there is no external fragmentation but internal fragmentation can arise in paging.



Segmentation

For a programmer, it might be more relevant to divide the logical address space of his program into variable-sized segments (with respect to his view of the main program, subroutines, data, etc.) than to divide it into fixed-size pages. Such variable-sized segments, which are a collection of logically related information, are the basis of the segmentation technique.

No internal fragmentation is there in Segmentation but its very costly algorithm.



SN	Size	Memory Address
1	400	100
2	200	500
3	100	800
N	x	NM

Q. What is Virtual Memory?

Ans. A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. The paging technique plays an important role in implementing virtual memory.

Q. What are Page faults?

Ans. Page fault dominates like an error. If any program tries to access a piece of memory but which does not exist in physical memory, meaning main memory, then page fault will occur. The fault specifies the O/S that it must trace all data into virtual memory management, and after that moves it from secondary memory like a hard disk to the primary memory of the system.

Disk Scheduling Algorithms

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

1. FCFS (First-come-first-serve)

It is the simplest form of disk scheduling algorithms. The I/O requests are served or processed according to their arrival. The request arrives first and will be accessed and served first.

2. SSTF (Shortest-seek-time-first)

In SSTF requests having shortest seek time are executed first. Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.

3. SCAN

In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as elevator algorithm. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

4. CSCAN

In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in CSCAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

5. LOOK

It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

6. CLOOK

As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end’s last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

7. RSS (Random scheduling)

8. LIFO

In LIFO (Last In, First Out) algorithm, newest jobs are serviced before the existing ones i.e. in order of requests that get serviced the job that is newest or last entered is serviced first and then the rest in the same order.