Important topics:

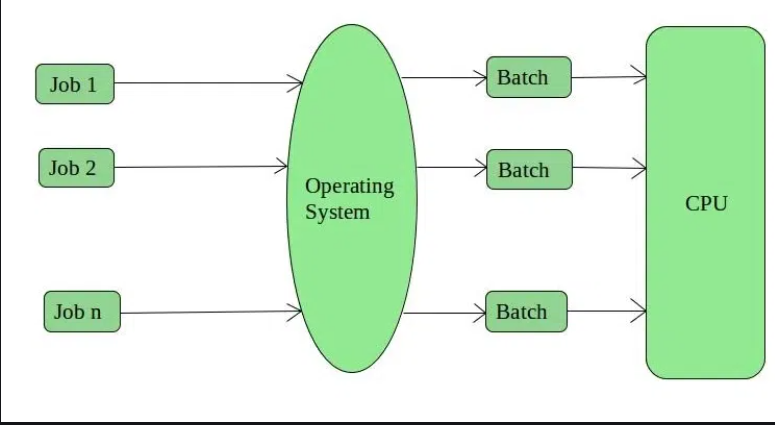
Explain Operating System in layman terms

* An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
* An operating system is software that enables applications to interact with a computer's hardware.
* The software that contains the core components of the operating system is called the kernel.

Types of OS - Batch OS, Multiprogramming OS, Multitasking OS, Time Sharing OS, Distributed OS, Real Time OS

Batch OS :

* 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.



Advantages of Batch Operating System

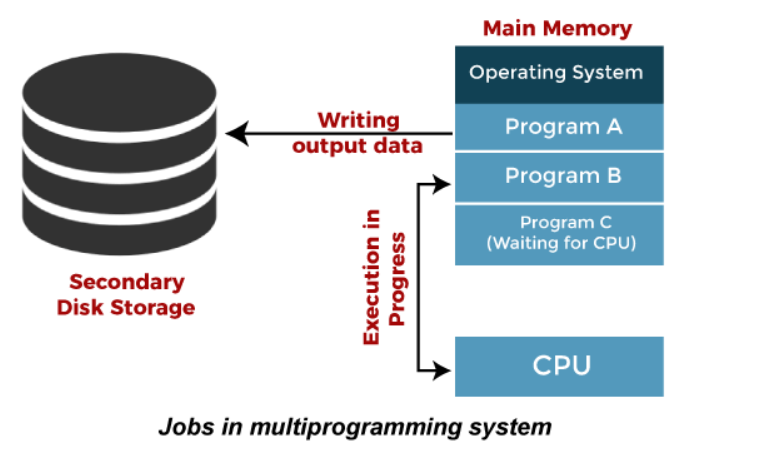
* Multiple users can share the batch systems.
* The idle time for the batch system is very less.

Disadvantages of Batch Operating System

* The other jobs will have to wait for an unknown time if any job fails.
* Batch systems are hard to debug.

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.



Advantages of Multiprogramming OS

* Throughout the system, it increased as the CPU always had one program to execute.
* Response time can also be reduced.

Disadvantages

 No user interaction with any program during execution.

Multiprocessing OS

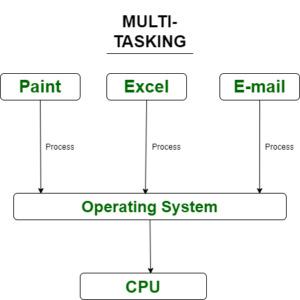
In Multiprocessing, Parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.

A diagram of a memory

Description automatically generated

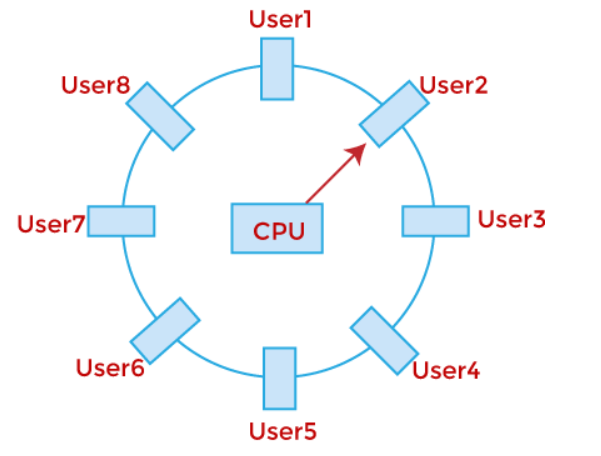
Multitasking os

* Multi tasking operating systems allow multiple users to perform multiple tasks at the same time.
* The allocation of system resources such as [input/output devices](https://www.geeksforgeeks.org/input-and-output-devices/), [CPU](https://www.geeksforgeeks.org/central-processing-unit-cpu/) and [memory](https://www.geeksforgeeks.org/computer-memory/) among processes can be easily managed by multi-tasking operating system.



Time sharing os

* 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.



Real time os

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.

Network os

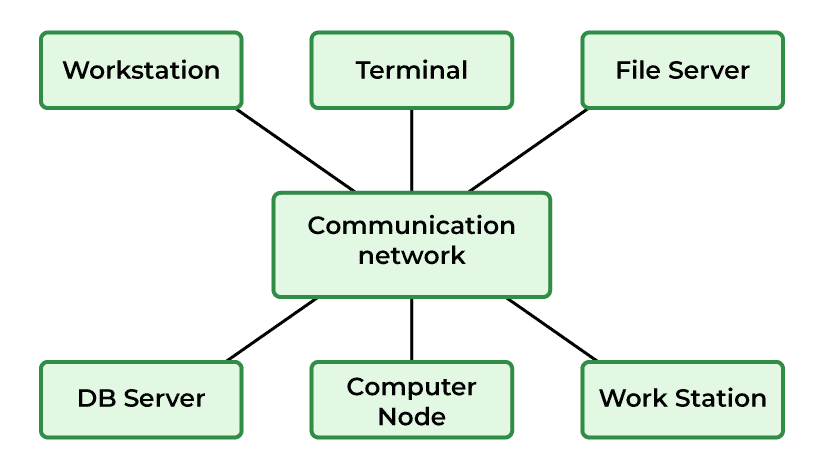
An Operating system, which includes software and associated protocols to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.

Types:-

* Peer to peer
* Client server

Distributed os

* A Distributed Operating System refers to a model in which applications run on multiple interconnected computers, offering enhanced communication and integration capabilities compared to a [network operating system](https://www.geeksforgeeks.org/what-is-a-network-operating-system/).
* In a Distributed OS, multiple [CPUs](https://www.geeksforgeeks.org/central-processing-unit-cpu/) are utilized, but for end-users, it appears as a typical centralized [operating system](https://www.geeksforgeeks.org/what-is-an-operating-system/).
* It enables the sharing of various resources such as CPUs, disks, [network interfaces](https://www.geeksforgeeks.org/python-network-interface/), nodes, and computers across different sites, thereby expanding the available data within the entire system.

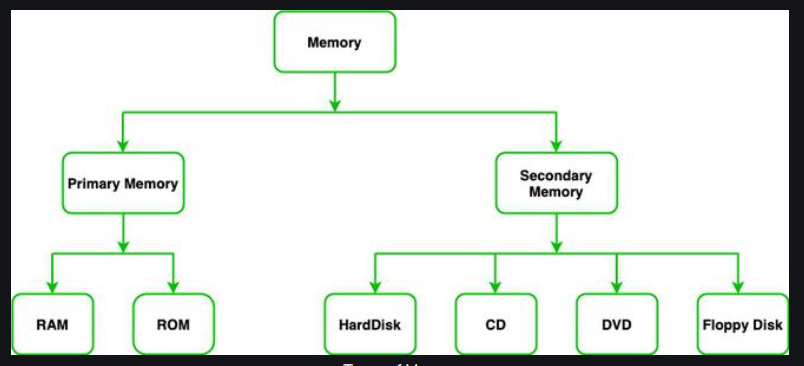


RAM vs ROM + Types (Asked in DE Shaw)

Memory is an important part of the Computer which is responsible for the storage of data and information on a temporary or permanent basis.

Memory can be classified into two broad categories:

* Primary Memory
* Secondary Memory



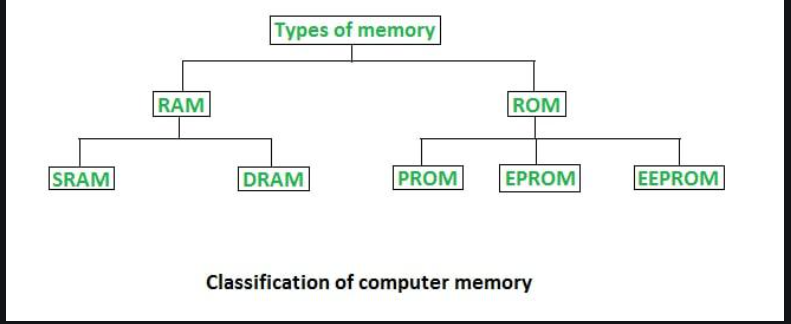
Primary Memory

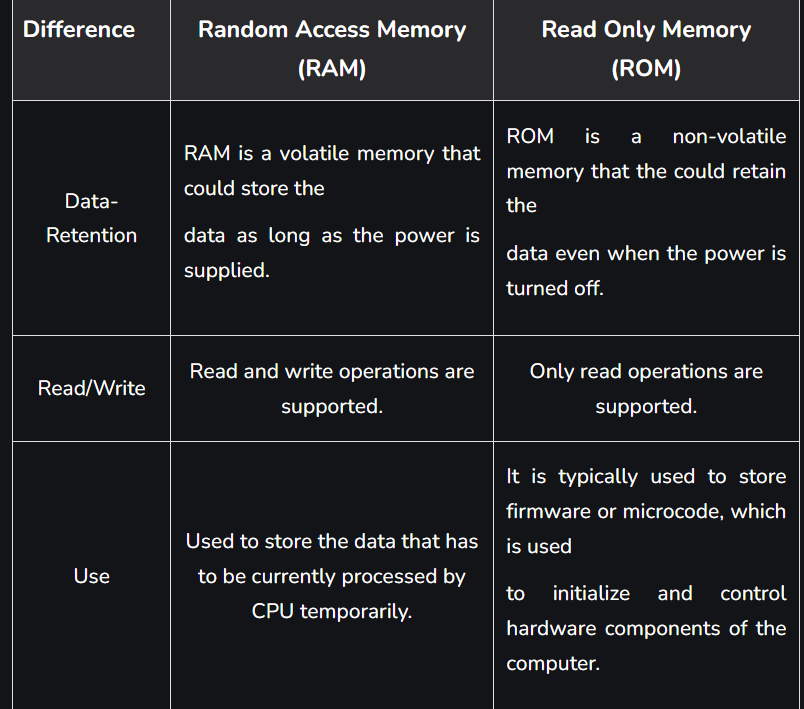
Primary Memory is a type of Computer Memory which is directly accessed by the Preprocessor. It is basically used to store data on which computer is currently working.

It has lesser storage than Secondary Memory.

It is basically of two types:

* [Random Access Memory (RAM)](https://www.geeksforgeeks.org/random-access-memory-ram-and-read-only-memory-rom/)
* [Read Only Memory (ROM)](https://www.geeksforgeeks.org/random-access-memory-ram-and-read-only-memory-rom/)





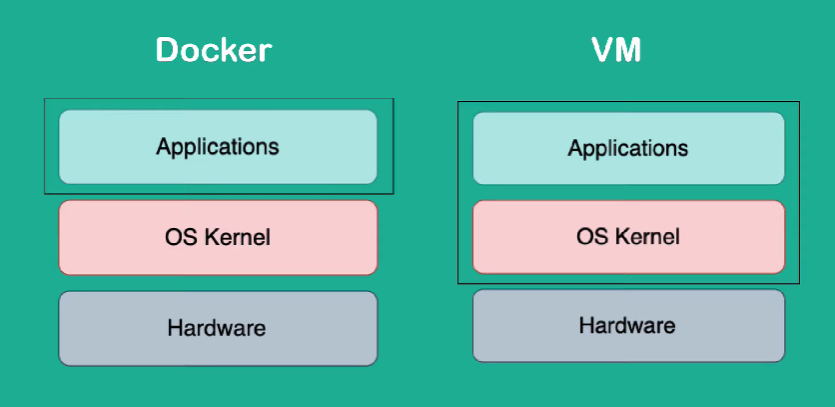
Virtualization vs Containerization (OR Virtual machine vs Docker)

Virtualization –

* Virtualization is the technology that can simulate your physical hardware (such as CPU cores, memory, disk) and represent it as a separate machine.
* It has its own Guest OS, Kernel, process, drivers, etc.
* Therefore, it is hardware-level virtualization. Most common technology is "VMware" and "Virtual Box".

Containerization –

* Containerization is "OS-level virtualization".
* It doesn't simulate the entire physical machine. It just simulates the OS of your machine.
* Containerization is the process of bundling the application code along with the libraries, configuration files, and dependencies required for the application to run cross-platform.



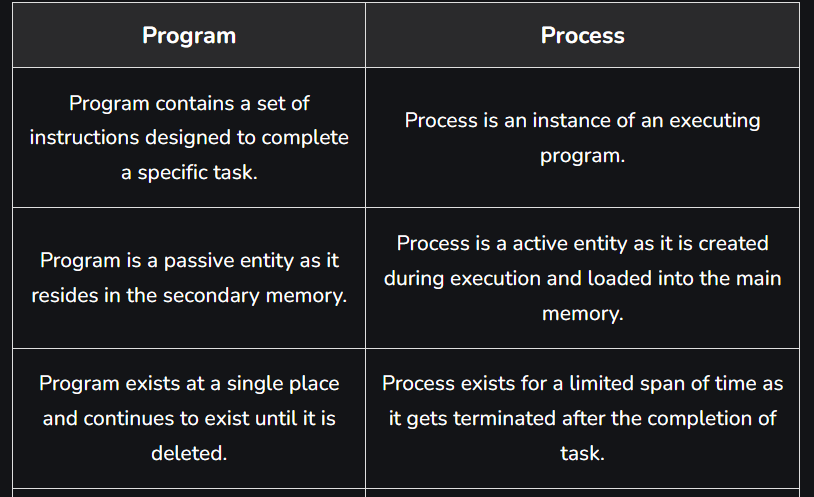
#### 1. Isolation

Virtualization results in a fully isolated OS and VM instance, while containerization isolates the host operating system machine and containers from one another. However, all containers are at risk if an attacker controls the host.

#### 2. Different Operating Systems

Virtualization can host more than one complete operating system, each with its own kernel, whereas containerization runs all containers via user mode on one OS. meaning Linux containers cannot be run on Windows and vice-versa.

Program vs Process (Asked in DE Shaw)



Process vs Thread

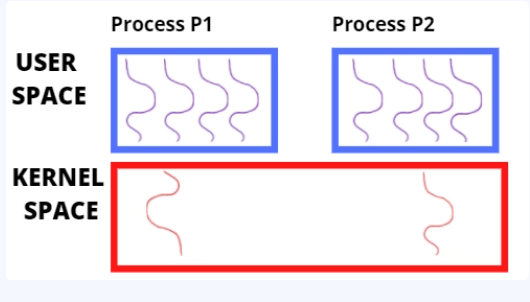
Process:

* Processes are basically the programs that are dispatched from the ready state and are scheduled in the CPU for execution
* The process takes more time to terminate and it is isolated means it does not share the memory with any other process.

Thread:

* Thread is the segment of a process which means a process can have multiple threads and these multiple threads are contained within a process.
* The [thread](https://www.geeksforgeeks.org/thread-in-operating-system/) takes less time to terminate as compared to the process but unlike the process, threads do not isolate.

User-level thread vs kernel level thread



User-level threads

* User-level threads are managed without kernel support by the run-time system and are supported above the kernel.
* The kernel does not know anything about the user-level threads.
* It treats them as if they are single-threaded processes.

Advantages:

* User-level threads are fast to create and efficient.
* User-level threads are easier to manage than kernel-level threads.

Disadvantages:

* OS is unaware of user-level threads, so the scheduler cannot schedule them properly.
* The entire process will get blocked if one user-level thread performs a blocking operation.

Kernel-level threads

* Kernel-level threads are supported and managed by the operating system. No runtime system is required in the case of this type of thread.
* The kernel has a thread table that keeps track of all threads in the system, rather than having a thread table in each process

Advantages:

* The kernel is fully aware of kernel-level threads, so the scheduler handles the process better.
* The kernel can still schedule another thread for execution if one thread is blocked.

Disadvantages:

* It is slower to create and not easy to manage
* Kernel-level threads are not generic and are specific to the Operating System.

Differences between multi-threading, multi-processing, multiprogramming, multi-tasking

<https://www.javatpoint.com/multiprogramming-vs-multiprocessing-vs-multitasking-vs-multithreading>

Microservices based architecture

Process scheduling

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

Preemptive vs Non-preemptive scheduling

Scheduling falls into one of two categories:

Non-preemptive: In this case, a process’s resource cannot be taken before the process has finished running. When a running process finishes and transitions to a [waiting state](https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/), resources are switched.

Preemptive: In this case, the OS assigns resources to a process for a predetermined period of time. The process switches from running state to ready state or from waiting for state to ready state during resource allocation. This switching happens because the CPU may give other processes priority and substitute the currently active process for the higher [priority process](https://www.geeksforgeeks.org/priority-of-process-in-linux-nice-value/).

Scheduling queue

* The Job Queue stores all processes that are entered into the system.
* The Ready Queue holds processes in the ready state.
* Device Queues hold processes that are waiting for any device to become available. For each I/O device, there are separate device queues.

The ready queue is where a new process is initially placed. The process waits in the ready queue till it is selected for execution.

## Key Concepts in CPU Scheduling

### Arrival Time

### **the arrival time refers to the moment in time when a**[**process**](https://www.baeldung.com/cs/process-scheduling)**enters the ready queue and is awaiting execution by the CPU**. In other words, it is the point at which a process becomes eligible for scheduling.

### Burst Time

### Burst time, also referred to as “execution time”. **It is the amount of CPU time the process requires to complete its execution**.

### Completion Time

**Completion time is when a process finishes execution and is no longer being processed by the CPU**. It is the summation of the arrival, waiting, and burst times.

### Turnaround Time

**The time elapsed between the arrival of a process and its completion** is known as turnaround time. That is, the duration it takes for a process to complete its execution and leave the system.

### Turnaround Time = Completion Time – Arrival Time

### Waiting Time

**This is a process’s duration in the ready queue before it begins executing**

### Waiting Time = Turnaround Time – Burst Time

### Response Time

### **It is the duration between the arrival of a process and the first time it runs.**

### Response Time = Time it Started Executing – Arrival Time

### CPU utilization

### CPU utilization refers to the amount of work handled by a CPU.

### Actual CPU utilization varies depending on the amount and type of managed computing tasks.

Throughput

A measure of the work done by the CPU is the number of processes being executed and completed per unit of time.

Types of process schedulers

## **Long Term Scheduler**

The job scheduler or long-term scheduler selects processes from the storage pool in the secondary memory and loads them into the ready queue in the main memory for execution.

## **Short Term Scheduler**

The short-term scheduler selects one of the processes from the ready queue and schedules them for execution. A scheduling algorithm is used to decide which process will be scheduled for execution next.

## **Medium Term Scheduler**

The medium-term scheduler swaps out a process from main memory. It can again swap in the process later from the point it stopped executing. This can also be called as suspending and resuming the process.

Different scheduling algorithms

### **First Come First Serve**

It is the simplest algorithm to implement. The process with the minimal arrival time will get the CPU first. The lesser the arrival time, the sooner will the process gets the CPU. It is the non-preemptive type of scheduling.

### **Round Robin**

In the Round Robin scheduling algorithm, the OS defines a time quantum (slice). All the processes will get executed in the cyclic way. Each of the process will get the CPU for a small amount of time (called time quantum) and then get back to the ready queue to wait for its next turn. It is a preemptive type of scheduling.

### **Shortest Job First**

The job with the shortest burst time will get the CPU first. The lesser the burst time, the sooner will the process get the CPU. It is the non-preemptive type of scheduling.

### **Shortest remaining time first**

It is the preemptive form of SJF. In this algorithm, the OS schedules the Job according to the remaining time of the execution.

### **Priority based scheduling**

In this algorithm, the priority will be assigned to each of the processes. The higher the priority, the sooner will the process get the CPU. If the priority of the two processes is same then they will be scheduled according to their arrival time.

Explain how does a process gets executed inside memory

Different stages

New (Create):

* In this step, the process is about to be created but not yet created.
* It is the program that is present in secondary memory that will be picked up by OS to create the process.

Ready:

* New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory.
* The process here is ready to run and is waiting to get the CPU time for its execution.
* Processes that are ready for execution by the CPU are maintained in a queue called ready queue for ready processes.

Run:

The process is chosen from the ready queue by the CPU for execution and the instructions within the process are executed by any one of the available CPU cores.

Blocked or Wait:

* Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or waits for the state.
* The process continues to wait in the main memory and does not require CPU.
* Once the I/O operation is completed the process goes to the ready state.

Terminated or Completed:

* Process is killed.
* The resources allocated to the process will be released or deallocated.

PCB

* A Process Control Block in OS (PCB) is a [***data structure***](https://www.datatrained.com/post/data-structure-using-c/) used by an operating system (OS) to manage and control the execution of processes.
* It contains all the necessary information about a process, including the process state, program counter, memory allocation, open files, and CPU scheduling information.
* The PCB is created by the OS when a process is created and is used to manage and control the execution of that process.

Optimal number of threads for a process

Ideally the total thread count for all the jobs should be the number of cores of the system, except on systems that support hyper-threading, in which it should be twice the number of cores. So if the system doesn't have hyper-threading, there are 8 calculations running, each should run in one thread.

Many Intel processors come with hyper-threading, so each core can support two threads. For example an 8 core system which supports hyper-threading should have 16 threads to utilize the system fully.

hyper-threading

Allowing a single processor to execute multiple threads simultaneously can complete tasks more quickly and efficiently, resulting in better overall system performance.

Some important terms associated with scheduling algorithms - Problem of Ageing, Starvation, Deadlock

Starvation

Starvation or indefinite blocking is a phenomenon associated with the Priority scheduling algorithms, in which a process ready for the CPU (resources) can wait to run indefinitely because of low priority.

In a heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

Differences between [Deadlock](https://www.geeksforgeeks.org/operating-system-process-management-deadlock-introduction/) and Starvation in OS are as follows:

* Deadlock occurs when none of the processes in the set is able to move ahead due to occupancy of the required resources by some other process as shown in the figure below, on the other hand, Starvation occurs when a process waits for an indefinite period of time to get the resource it requires.
* Another name for deadlock is Circular Waiting. Another name for starvation is Lived lock.
* When deadlock occurs no process can make progress, while in starvation apart from the victim process other processes can progress or proceed.

Solution to starvation :- aging

* Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time.
* For example, if priority range from 127(low) to 0(high), we could increase the priority of a waiting process by 1 Every 15 minutes.

Limitations of the aging technique

* If the aging rate is too slow, it may take a long time for low-priority processes to receive the required resources.
* On the other hand, if the aging rate is too fast, it can cause high-priority processes to starve.

Synchronization

* When two or more process cooperates with each other, their order of execution must be preserved otherwise there can be conflicts in their execution and inappropriate outputs can be produced.
* The procedure involved in preserving the appropriate order of execution of cooperative processes is known as Process Synchronization.

Race condition

A Race Condition typically occurs when two or more threads try to read, write and possibly make the decisions based on the memory that they are accessing concurrently.

Critical Section

The regions of a program that try to access shared resources and may cause race conditions are called critical section. To avoid race condition among the processes, we need to assure that only one process at a time can execute within the critical section.

Requirements of Synchronization mechanisms

Mutual exclusions

By Mutual Exclusion, we mean that if one process is executing inside critical section then the other process must not enter in the critical section.

**Progress**

Progress means that if one process doesn't need to execute into critical section then it should not stop other processes to get into the critical section.

**Bounded Waiting**

We should be able to predict the waiting time for every process to get into the critical section. The process must not be endlessly waiting for getting into the critical section.

## Solutions To The Critical Section Problem

Petersons solution

It is a classical software-based solution.

In Peterson’s solution, we have two shared variables:

* boolean flag[i]: Initialized to FALSE, initially no one is interested in entering the critical section
* int turn: The process whose turn is to enter the critical section.

Peterson’s Solution preserves all three conditions

Disadvantages of Peterson’s Solution

It involves busy waiting

Mutex lock

* Mutex is a locking mechanism used to synchronize access to a resource in the critical section. In this method, we use a LOCK over the critical section.
* The LOCK is set when a process enters from the entry section, and it gets unset when the process exits from the exit section.

Semaphores:-

Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

* **Wait**

The wait operation decrements the value of its argument **S**, if it is positive. If **S** is negative or zero, then no operation is performed.

* **Signal**

The signal operation increments the value of its argument **S**.

## **types of Semaphores**

Counting Semaphores

* These are integer value semaphores and have an unrestricted value domain.
* These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources.
* If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.

Binary Semaphores

* The binary semaphores are like counting semaphores but their value is restricted to 0 and 1.
* The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0.
* It is sometimes easier to implement binary semaphores than counting semaphores.

Deadlock

A deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.



Deadlock can arise if the following four conditions hold simultaneously (Necessary Conditions)

Mutual Exclusion: Two or more resources are non-shareable (Only one process can use at a time)   
Hold and Wait: A process is holding at least one resource and waiting for resources.   
No Preemption: A resource cannot be taken from a process unless the process releases the resource.   
Circular Wait: A set of processes waiting for each other in circular form.

Methods for handling deadlock

* 1. Deadlock prevention or avoidance:
* The idea is to not let the system into a deadlock state.
* This system will make sure that above mentioned four conditions will not arise.
* Prevention is done by negating one of the above-mentioned necessary conditions for deadlock.
* Avoidance is kind of futuristic. By using the strategy of “Avoidance”, we have to make an assumption. We need to ensure that all information about resources that the process will need is known to us before the execution of the process.

2) Deadlock detection and recovery

If Deadlock prevention or avoidance is not applied to the software then we can handle this by deadlock detection and recovery. which consist of two phases:

* In the first phase, we examine the state of the process and check whether there is a deadlock or not in the system.
* If found deadlock in the first phase then we apply the algorithm for recovery of the deadlock.

Recovery from Deadlock

PROCESS TERMINATION

1. Abort all deadlocked processes:

This approach breaks the deadlock cycle, but it comes at a significant cost. The processes that were aborted may have executed for a considerable amount of time, resulting in the loss of partial computations. These computations may need to be recomputed later.

2. Abort one process at a time:

Instead of aborting all deadlocked processes simultaneously, this strategy involves selectively aborting one process at a time until the deadlock cycle is eliminated. However, this incurs overhead as a deadlock-detection algorithm must be invoked after each process termination to determine if any processes are still deadlocked.

Resource Preemption

Selecting a victim

Resource preemption involves choosing which resources and processes should be preempted to break the deadlock. The selection order aims to minimize the overall cost of recovery.

 Rollback:

If a resource is preempted from a process, the process cannot continue its normal execution as it lacks the required resource. Rolling back the process to a safe state and restarting it is a common approach.

Starvation prevention:

To prevent resource starvation, it is essential to ensure that the same process is not always chosen as a victim. If victim selection is solely based on cost factors, one process might repeatedly lose its resources and never complete its designated task. To address this, it is advisable to limit the number of times a process can be chosen as a victim, including the number of rollbacks in the cost factor.

1. Deadlock ignorance:

If a deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take. we use the ostrich algorithm for deadlock ignorance.

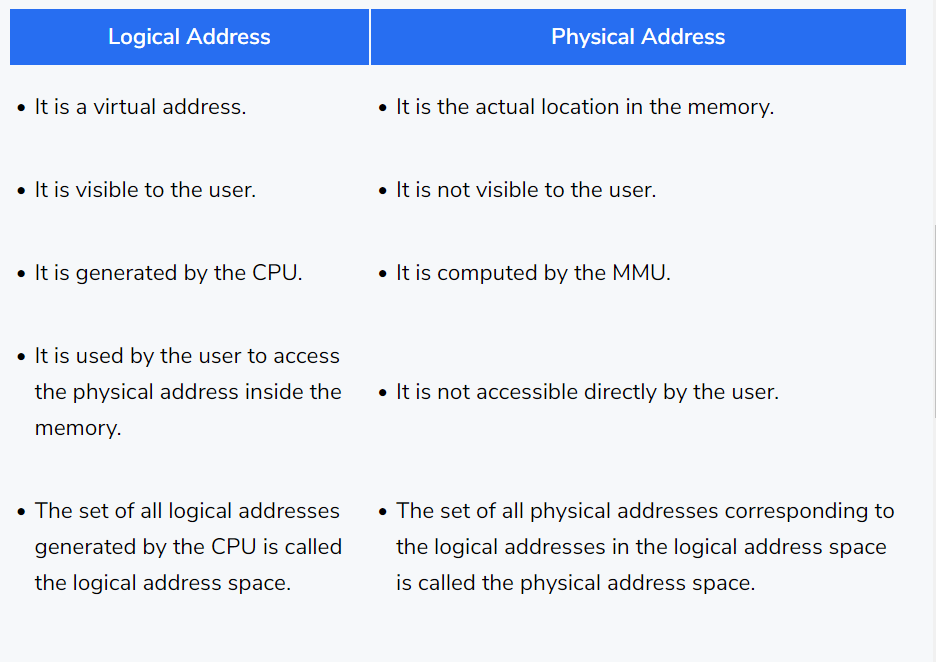
In Deadlock, ignorance performance is better than the above two methods but the correctness of data.

Memory Management [Very IMP]: Primary vs Secondary Memory, memory Allocation while running a process [IMP], Paging and segmentation (Basics should be clear)

Memory is an essential component of a computer used to store data. Because the amount of main memory available in a computer system is fairly restricted, its management is crucial to the computer system.

Memory management in OS is the process of regulating and organizing computer memory in order to allocate and deallocate memory space efficiently for programs and applications that require it. This helps to guarantee that the system runs efficiently and has enough memory to run apps and tasks.

Logical vs physical address



Functions

Memory Allocation

Memory management ensures that the needed memory space is allocated to the new process whenever a process is created and requires memory. Memory Management also keeps track of the system's allocated and free memory.

Memory Deallocation

Like memory allocation, whenever a process completes its execution, memory management ensures that the space and the memory resources it holds are released. Any newly created process can use the freed memory.

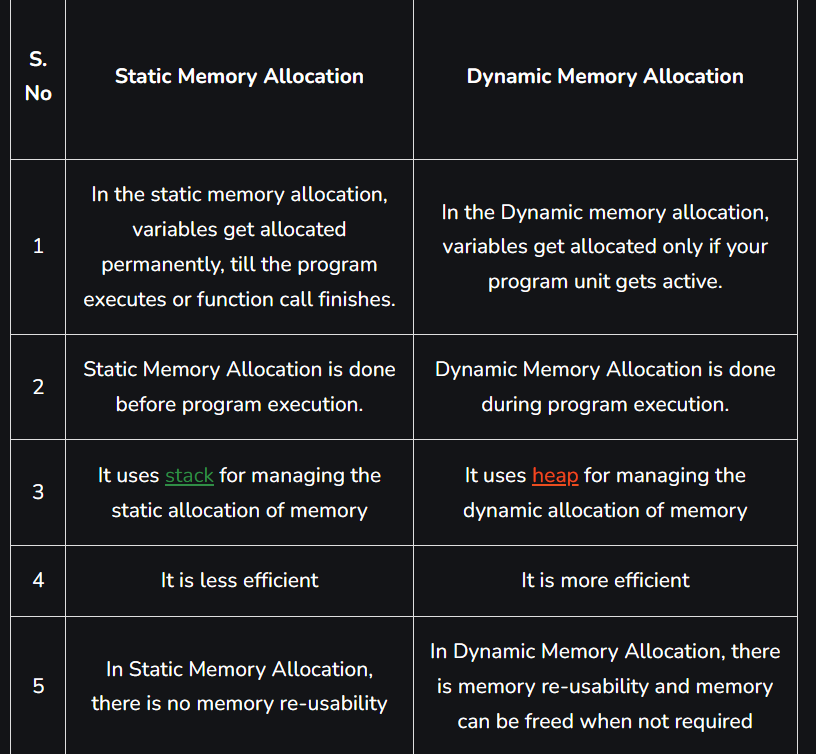
Memory Sharing

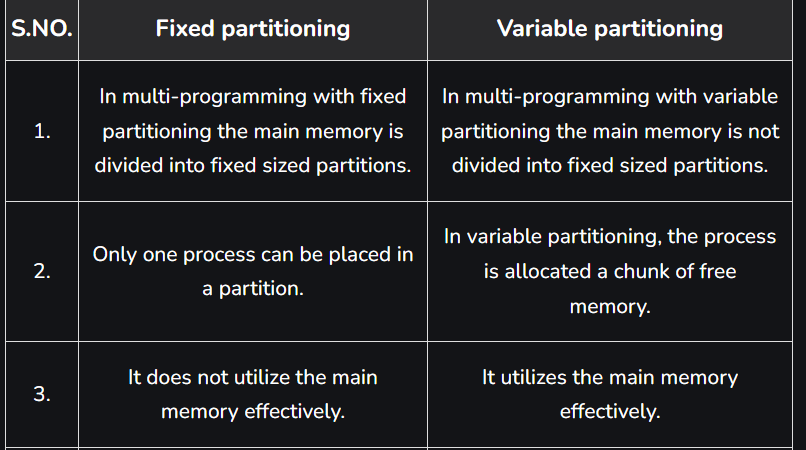
Memory sharing is also one of the main goals of memory management in OS. Some processes might require the same memory simultaneously. Memory management ensures that this is made possible without breaking any authorization rules.

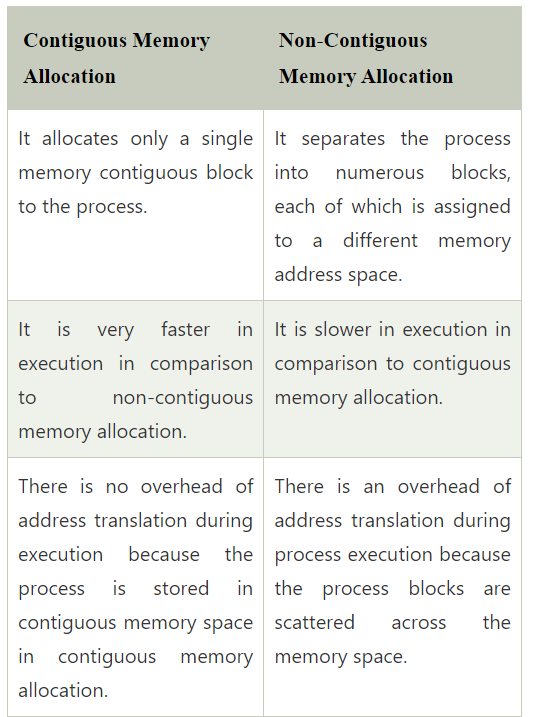
Memory Protection

Memory Protection refers to preventing any unauthorized memory access to any process. Memory management ensures memory protection by assigning correct permissions to each process.

Memory Allocation Schemes







Memory Management in OS Techniques

Swapping

* Swapping is a mechanism for temporarily moving a process from main memory to secondary storage and making that memory available to other processes.
* The system switches the process from secondary storage to main memory at a later time.

Fragmentation

The process of dividing a computer file, such as a data file or an executable program file, into fragments that are stored in different parts of a computer’s storage medium, such as its hard disc or RAM, is known as fragmentation in computing.

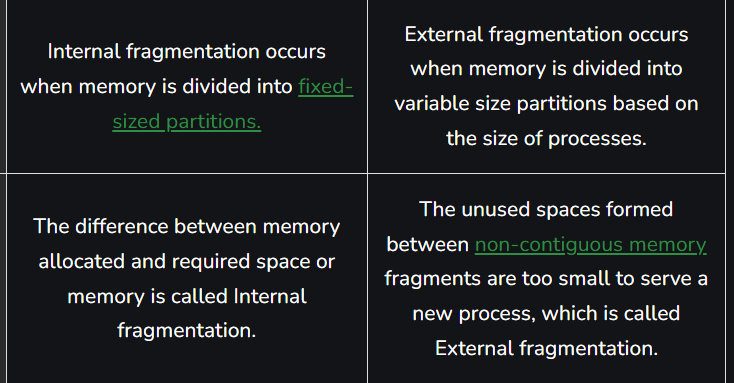
cause

This can happen when a file is too large to fit into a single contiguous block of free space on the storage medium, or when the blocks of free space on the medium are insufficient to hold the file.

Effect

This can reduce system performance and make it more difficult to access the file. It is generally best to defragment your hard disc on a regular basis to avoid fragmentation, which is a process that rearranges the blocks of data on the disc so that files are stored in contiguous blocks and can be accessed more quickly.

Types :-



#### Non-Contiguous Memory Allocation Technique

1)Paging

Paging is a non-contiguous memory allocation technique in which secondary memory and the main memory is divided into **equal size**partitions. The partitions of the secondary memory are called **pages**while the partitions of the main memory are called **frames**. They are divided into equal size partitions to have maximum utilization of the main memory and avoid external fragmentation.

#### Advantages of Paging

1. There is no external fragmentation as it allows us to store the data in a non-contiguous way.
2. Swapping is easy between equal-sized pages and frames.

#### Disadvantages of Paging

1. As the size of the frame is fixed, so it may suffer from internal fragmentation. It may happen that the process is too small and it may not acquire the entire frame size.
2. For every process, we have an independent page table and maintaining the page table is extra overhead.

#### Segmentation

In paging, we were blindly diving the process into pages of fixed sizes but in segmentation, we divide the process into modules for better visualization of the process. Here each segment or module consists of the same type of functions. **For example,**the main function is included in one segment, library function is kept in other segments, and so on. As the size of segments may vary, so memory is divided into variable size parts.

#### Advantages of Segmentation

1. The size of the segment table is less compared to the size of the page table.
2. There is no internal fragmentation.

#### Disadvantages of Segmentation

1. When the processes are loaded and removed ( during swapping ) from the main memory then free memory spaces are broken into smaller pieces and this causes external fragmentation.

Compaction => external fragmentation solution

Compaction refers to combining of all the empty spaces together and processes. Compaction helps to solve the problem of fragmentation, but it requires a lot of CPU time. It moves all the occupied areas of storage to one end and leaves one large free space for incoming jobs, instead of numerous small ones.

Placement Algorithm

• Best-fit:-

It chooses the block, that is closest in size to the given request from the beginning to the ending free blocks. We must search the entire list, unless it is ordered by size. This strategy produces the smallest leftover hole.

First-fit

First-fit allocation method is a memory allocation technique used in operating systems where the first free partition that can accommodate the process is selected for allocation. When a process requests memory, the operating system searches for the first free partition that can accommodate the process.

### **Next-Fit**

Next-fit allocation method is a memory allocation technique used in operating systems where the next free partition that can accommodate the process is selected for allocation. When a process requests memory, the operating system starts searching for a free partition from the location of the last allocation and continues searching until it finds the next free partition that can accommodate the process.

Worst-fit:-

It allocates the largest block. We must search the entire the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

First-fit algorithm is the simplest, best and fastest algorithm. Next-fit produce slightly worse results than the first-fit. Best-fit is the worst performer, even though it is to minimize the wastage space

What is Virtual memory and why?

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Demand Paging

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory.

A page is copied to the main memory when its demand is made or page fault occurs. There are various page replacement algorithms which are used to determine the pages which will be replaced.

Page Replacement Algorithms:

* First In First Out (FIFO): This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.
* Optimal Page replacement: In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
* Least Recently Used: In this algorithm, page will be replaced which is least recently used.

Thrashing [IMP concept]

* **Thrashing** is when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages.
* This state in the operating system is known as thrashing.
* Because of thrashing, the CPU utilization is going to be reduced or negligible.
* The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults.
* As a result, no valuable work would be done by the CPU, and the CPU utilization would fall drastically

What is Cache and why is it used?

* Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU.
* Cache Memory holds frequently requested data and instructions so that they are immediately available to the CPU when needed.
* Cache memory is costlier than main memory or disk memory but more economical than CPU registers.

Levels of Memory

* Level 1 or Register: It is a type of memory in which data is stored and accepted that are immediately stored in the CPU. The most commonly used register is Accumulator, Program counter, Address Register, etc.
* Level 2 or Cache memory: It is the fastest memory that has faster access time where data is temporarily stored for faster access.
* Level 3 or Main Memory: It is the memory on which the computer works currently. It is small in size and once power is off data no longer stays in this memory.
* Level 4 or Secondary Memory: It is external memory that is not as fast as the main memory but data stays permanently in this memory.

LRU Cache implementation

* Cache replacement algorithms are efficiently designed to replace the cache when the space is full. The Least Recently Used (LRU) is one of those algorithms.
* here when the cache memory is full, LRU picks the data that is least recently used and removes it in order to make space for the new data.
* The priority of the data in the cache changes according to the need of that data i.e. if some data is fetched or updated recently then the priority of that data would be changed and assigned it to highest priority, and the priority of the data decreases if it remains unused operations after operations.

Also, practice standard interview questions from:

Interview questions:

<https://www.geeksforgeeks.org/commonly-asked-operating-systems-interview-questions-set-1/>

<https://www.interviewbit.com/operating-system-interview-questions/>

Other questions for practice: <https://www.javatpoint.com/operating-system-interview-questions>

Revision:

<https://www.geeksforgeeks.org/last-minute-notes-operating-systems/>

Book (if enough time):

# Operating System Principles by Galvin: <https://amzn.to/2UuxwEJ>