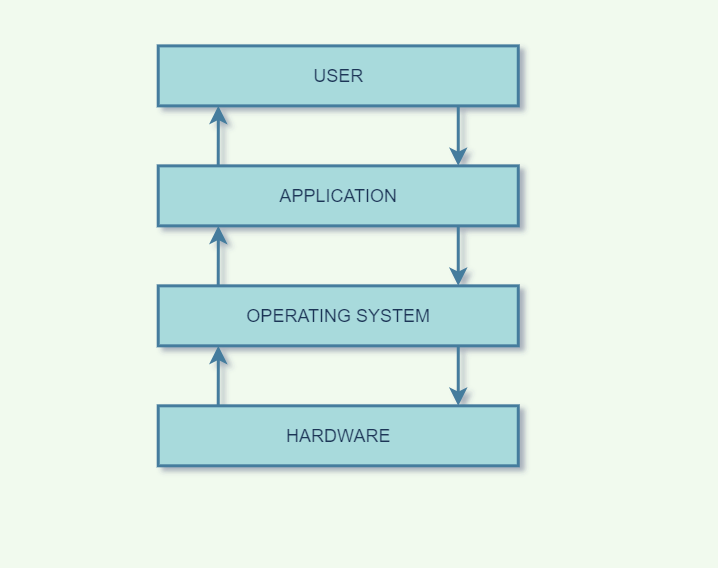
**Operation System**

An Operating system is a type of software that works as an interface between a user and a computer hardware Component. When the computer System boots up, the operating system is the first program that loads.

Kernel is an internal part of the Operating System that processes the data at the hardware level. It manages input-output management, memory, and process management.

Shell is the outer part of the OS that manages interaction between user and operating system. The Shell will communicate with os by either taking input from the user or using shell script.



**Type of Operation System**

1. Batch Operation system
2. Distributed Operating System
3. Time-Sharing Operating System
4. Network Operating System
5. Multiprogramming Operating System
6. Multiprocessing Operating system
7. Real-Time Operating System
8. Desktop Operating System
9. Mobile Operating System
10. Embedded Operating System

1. **Batch Operating System**: A batch OS grabs all programs and data in the batch form and then processes them. The main aim of using a batch processing system is to decrease the setup time while submitting similar jobs to CPU.

All Jobs are performed in repeating without the user’s permission.

It is difficult to debug this system.

Example- Payroll System, Billing System, Bank Invoice System, Daily Report System.

2. **Time-Sharing Operating System**: It is a logical extension of multiprogramming. It uses CPU scheduling and multiprogramming to provide each user with a small time- shared Computer.

The CPU executes multiple jobs by switching among them, but switches occur so frequently that users can interact with each program or task while it is running.

Example- Windows 2000 server, Windows NT server, Linux Unix Etc.

3. **Distributed Operating System**: This system is based on autonomous but interconnected computers communicating with each other via communication line or shared server or network.

Distributed Os server multiple applications and multiple users in real-time

Example- Locus, Mach, Dynix

4. **Network Operating System:** Network Operating system is an operating system that has a special function to connect the computer or devices into a local-area network and inter-network. Network OS has 2 types of operating system.

1. peer-to-peer network Operating system
2. Client/server network Operating system

5. **Real-Time Sharing Operating System**: Real time operating system is an operating system that runs a multi thread application and completes in a given time. Or can meet a real-time deadline.

Easier Testing.

Code Reuse.

Improved Efficiency.

Idle Processing.

Example- Airline traffic control systems, Command Control Systems, Airlines reservation system.

Heart Pacemaker, Network Multimedia Systems, Robotics.

6. **Multiprogramming Operating System**: Multiprogramming Operating system runs multi program on single processor computer. When the program is waiting for input-output transfer so at the same time other programs are able to use CPU. A multiprogramming system has 2 types.

1. Multitasking Operating system- Enables execution of multiple programs at the same time. The operating system accomplishes this by swapping each program in and out of memory one at a time.
2. Multiuser Operating System: This allows many users to share processing time on a powerful central computer from different terminals. The operating system accomplishes this by rapidly switching between terminals, each of which receives a limited amount of processor time on the central computer.

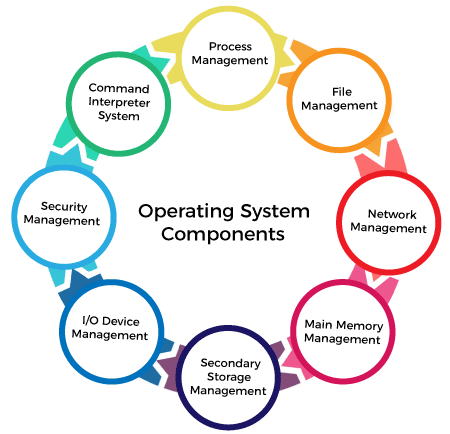
Example- Apps like office, chrome, Windows O/S, UNIX O/S.

7. **Multiprocessing Operating System**: Multiprocessor operating system utilizes multiple processors, which are connected with physical memory, other computer hardware components. The main objective of using a multiprocessor OS is to consume high computing power and increase the execution speed of the system.

**Functions of Operating System**

1. **Booting**: An operating system manages the startup of a device.
2. **Memory Management**: An Operating system manages the allocation and deallocation of the memory to various processes and ensures that the other processes do not consume the memory allocated to one process.
3. **Loading and Execution** : an operating system starts and executes a program.
4. **Data Security**: An operating system protects your data from cyber attacks.
5. **Drive/Disk Management**: an operating system manages computer drives and divides disks.
6. **Device control**: an operating system enables you to allow or block access to devices.
7. **User interface**: allow users to enter and receive the information. It is known as UI.
8. **Error Detection** : the operating system checks the system for any external threat and any type of damage hardware then displays the several alerts to the user.
9. **File Management**: an operating system keeps track of information regarding the creation, deletion, transfer , copy and storage of files in an organised way.

**Components of Operating System**



**Process Management**: Process is a program in execution. It is a procedure for managing many processes running simultaneously on the operating system. Process management keeps processes running efficiently.It also uses memory allocated to them and shutting them down when needed.

An operating system supports two system calls to manage processes create or Kill-

1. Create a system cell used to create a new process.
2. Kill system call used to delete an existing process.

**File Management**: A file is a set of related information defined by its creator. It commonly represents programs and data. The file management service can also be treated as an abstraction as it hides the information about the disks from the user. The system call for file management include-

1. File creation.
2. File deletion.
3. Read and Write operations.

**Network Managemen**t: Network management is a set of processes and procedures that help organisations to optimise their computer networks. Mainly, it ensures that users have the best possible experience while using network applications and services.

It helps to access shared resources that help computation to speed up or offers data availability and reliability.

**Main Memory Management**: The memory management process is conducted by using a sequence of reads or writes of specific memory addresses. It should be mapped to absolute addresses and loaded inside the memory to execute a program.

Functions:

1. It helps you to keep track of primary memory.
2. Allocates the memory when a process requests.
3. It also de-allocates the memory when a process no longer requires or has been terminated.

**Secondary Memory Management**: Computers use hard drives/SSD as the primary storage of both program and data. However, the secondary storage management also works with storage devices, such as USB flash drives and CD/DVD drives.

Primary memory is very small to store all data and programs permanently so secondary memory works as back up of the main memory.

Function:

1. Storage allocation
2. Free space management
3. Disk scheduling

**I/O Devices Management**: The I/O device management component is an I/O manager that hides the details of hardware devices and manages the main memory for devices using cache and spooling.

Function:

1. It offers a buffer caching system.
2. It provides general device driver code.

**Security Management**: Security management refers to the various processes where the user changes the file, memory, CPU and other hardware resources that should have Authorization from the operating system.

**Command Interpreter System**: it is an important component of the OS. The command interpreter is the primary interface between the user and the rest of the system. It allows the user to enter a command on the CMD. The Command interpreter accepts and executes commands entered by the user.

**Kernel**

Kernel is the core component of an operating system. Kernel acts as a bridge between application and data processing performed at hardware level using inter-process communication and system cells. In other words, it acts as an interface between hardware and user application.

It is a system program which converts user commands into machine language. Kernel is the first program to load when the operating system loads.

Objectives of Kernel:

1. To control task management.
2. To control memory management.
3. To control disk management.
4. To decide the state of incoming processes.

Type of Kernel:

1. Monolithic Kernel: it has dependencies between system components. It has huge lines of code which is complex but it has good performance.

Example- Unix, Linux, Open VMS, XTS-400 etc.

1. Micro Kernel: it has virtual memory and thread scheduling. It is more stable with less services in kernel space.

Example- Mach, L4, Minix etc.

1. Hybrid Kernel: It is a combination of monolithic and micro kernel. It has speed and design of monolithic kernel and modularity and stability of micro kernel.

Example: Windows NT, Netware etc.

1. Exo Kernel: It follows end-to-end principle. It has the fewest hardware abstraction possible. It allocates physical resources to applications.

Example: Exos, Nemesis, etc.

1. Nano Kernel: It is the type of kernel that offers hardware abstraction but without system services.

Example- EROS etc.

**System Call**

A system call is a method for a computer program to request services from the kernel on which it is running. It is a method of interacting with the operating system via programs.

The Application Program Interface(API) connects the operating system’s functions to user programs. It acts as a link between the OS and process, allowing user-level programs to request operating system services.

Why we need system calls in OS:

1. It is required when a file system wants to create or delete a file.
2. Network connections require the system calls to send and receive data packets.
3. For reading and writing a file.
4. For accessing hardware devices.
5. For creating and managing processes.

Types of System calls:

1. Process control: process Control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process etc.
2. File Management: It is used to handle the files. Some file management examples include creating files, deleting files, open, close, read, write etc.
3. Device Management: It is used to deal with devices, Some Device management examples include read, write, device, get device attribute, release device etc.
4. Information Maintenance: It is used to maintain information. There are some examples- getting system data, set time and date, get time or date.
5. Communication: it is used for communication. There are examples: create, delete communication connections, send, receive messages etc.

| **Process** | **Windows** | **Unix** |
| --- | --- | --- |
| **Process Control** | CreateProcess()  ExitProcess()  WaitForSingleObject() | Fork()  Exit()  Wait() |
| **File Manipulation** | CreateFile()  ReadFile()  WriteFile()  CloseHandle() | Open()  Read()  Write()  Close() |
| **Device Management** | SetConsoleMode()  ReadConsole()  WriteConsole() | Ioctl()  Read()  Write() |
| **Information Maintenance** | GetCurrentProcessID()  SetTimer()  Sleep() | Getpid()  Alarm()  Sleep() |
| **Communication** | CreatePipe()  CreateFileMapping()  MapViewOfFile() | Pipe()  Shmget()  Mmap() |
| **Protection** | SetFileSecurity()  InitializeSecurityDescriptor()  SetSecurityDescriptorgroup() | Chmod()  Umask()  Chown() |

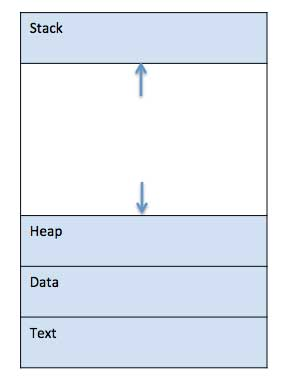
**Processes**

A process is basically a program in execution. In simple words, we write our computer program in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

In this, the operating system helps you to design, plan, schedule, and terminate the process used by CPU.

Properties of Process:

1. Creating each process requires separate system calls for each process.
2. Process is not share the data.
3. Processes use the IPC (Inter-Process Communication) for communication that significantly increases the number of system calls.
4. Process management takes more system calls.
5. It can be divided into four sections: stack, heap, text, and data.



Stack: The process stack contains the temporary data such as method/function parameters, return address and local variables.

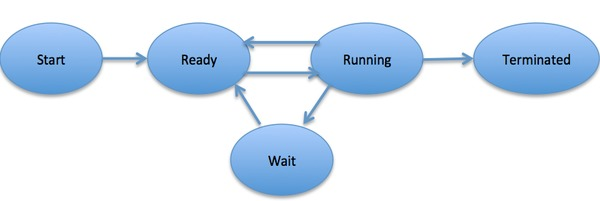
Heap: This is Dynamically allocated memory to a process during its run time.

Data: It contains the global and static variables.

Text: This includes the current activity represented by the value of the program counter.

**Process Life Cycle**

When a process executes, it passes through different states. These states may differ in different OS, and the names of these state are not also standardised.



1. Start: This is the initial state when a process is first started/created.
2. Ready: The process is waiting to be assigned to a processor(CPU time). 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.
3. Running: Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.
4. Waiting: Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.
5. Suspended Ready: When the ready queue becomes full, some processes are moved to suspended ready state. When Process is swapped out of main memory and placed into external storage by scheduler.
6. Suspended Block: When the waiting queue becomes full.
7. Terminated or Exit: Once the process finishes its execution, or it is terminated by the operating system.

**Types of Schedulers or Process Scheduler**

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

A scheduler is a type of system software that allows us to handle process scheduling.

1. Short-term Scheduler
2. Medium-term Scheduler
3. Long-term scheduler

Short-term scheduler(Context switching time/ CPU scheduler): it will decide which process to be executed next and then it will call the dispatcher.

A dispatcher is software that moves the process from ready state to runstate and vice versa.

Medium-term scheduler(Swapping time/ Swapping Scheduler): It is responsible for suspending or resuming the process. It is used for swapping which is moving the process form main memory to external memory and vice versa.

Long-term scheduler(performance/ job Scheduler): It brings the new process to the “ready state”. Make a decision about how many processes should be made to stay in ready state, this decides the degree of multiprogramming.

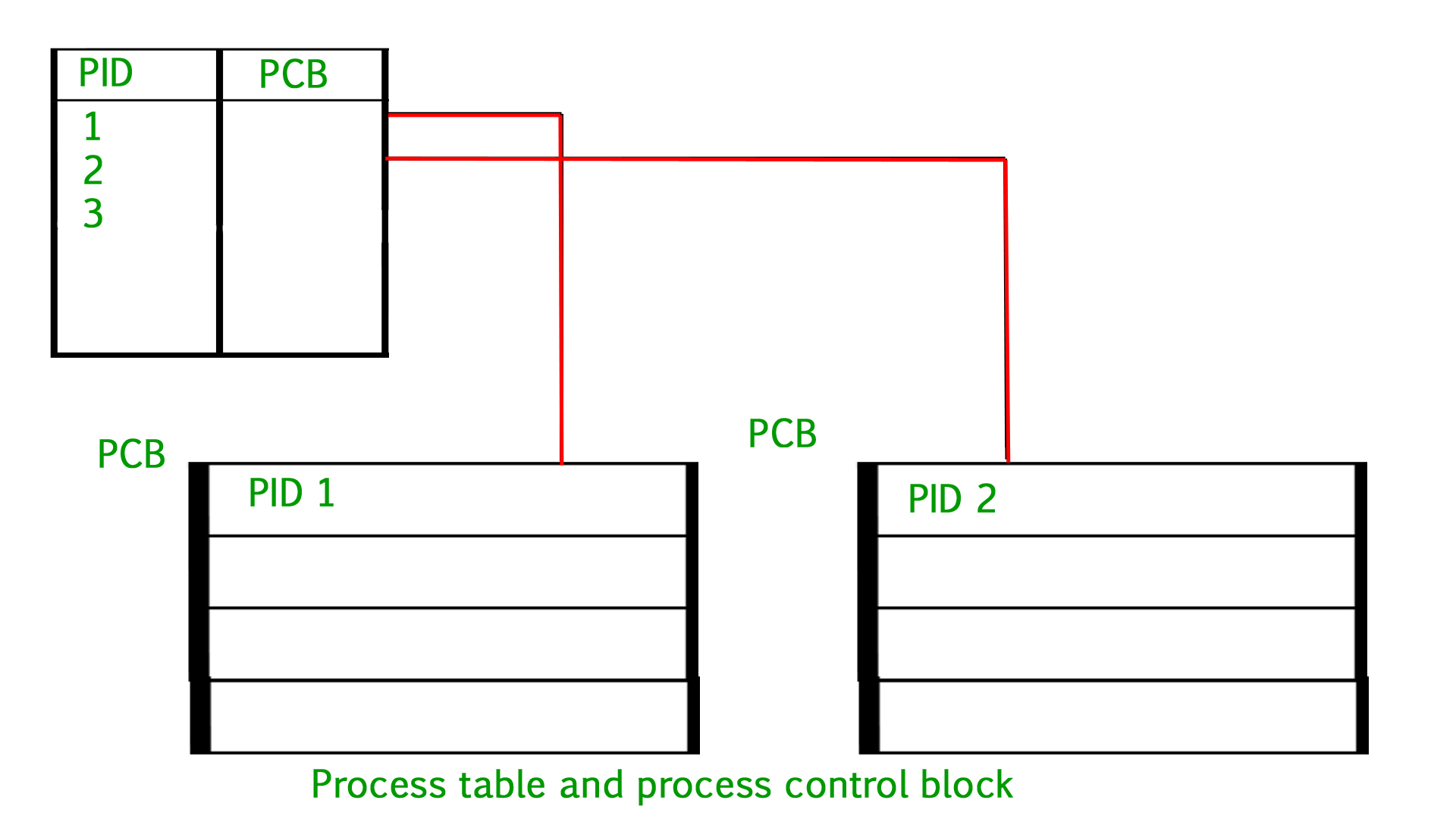
Multiprogramming- we have many processes ready to run.

**Process Control Block (PCB)/ Process Table**

The Process table (PT) contains an entry for each process present in the system. In simple words, a process table is an array of PCBs that logically contains PCBs for all of the current processes in the system.

A Process Control Block is a data structure maintained by the OS for every Process.The PCB is identified by an integer process ID (PID).

The process Control Block is used to track the process’s execution state.Each block of memory contains info about the process state, program counter, stack pointer, status of opened file, scheduling algorithm, etc. for every process transition( changes state) must update in the process's PCB.



1. Process ID: Unique Identification for each of the processes in the operating system.
2. Process State: The current state of the process . whether iti is ready, running, waiting etc.
3. Process Privileges: This is required to allow/disallow access to system resources.
4. Pointer: A pointer to the parent process.
5. Program Counter: It is a pointer to the address of the next instruction to be executed for this process.
6. CPU registers: Various CPU registers where processes need to be stored for execution for running.
7. CPU scheduling Information: Process priority and other scheduling information which is required to schedule the process.
8. Memory management information: This includes the information of page table, memory limits, segments table depending on memory used by the OS.
9. Accounting Information: This includes the amounts of CPU used for process execution, time limits, execution ID, etc.
10. IO state information: This includes a list of I/O devices allocated to the process.

Advantages/Disadvantages of PCB:

1. PCB provides an efficient way to manage processes in an operating system.
2. It allows the operating system to manage system resources.

Disadvantages

1. The process table and PCB can reduce the system performance.
2. It increases the system complexity and makes it more difficult to develop and maintain os.
3. Security risk

Que- what is a Distributed System?

Ans- distributed computing and distributed databases, a distributed system is a collection of independent components located on different machines that share messages with each other in order to achieve common goals.

**Distributed systems must have a network that connects all components (machines, hardware, or software) together so they can transfer messages to communicate with each other.**

**Telephone Network**

**Cellular Network**

**Shared Server- multiple users can use the same server over the internet or communication line. They can share the data to each other.**

Google doc.

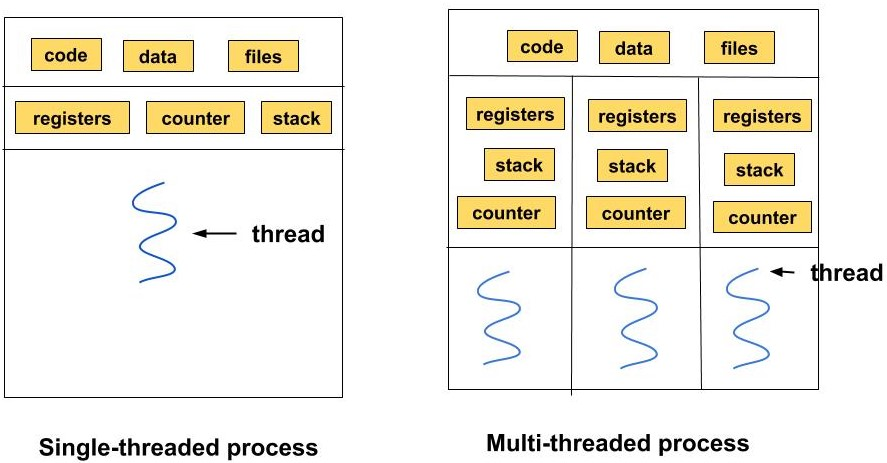
**Thread**

A thread is the smallest segment of instructions that can be handled independently by a scheduler.

In other words, thread is an execution unit that is part of a process. A process can have multiple threads, all executing at the same time. It helps you to improve the application performance using parallelism.

Multiple threads shared information like data, code , file, etc. We can implement threads in 3 different ways.

1. Kernel-level threads
2. User-level threads
3. Hybrid threads



User Level Threads:

1. They are managed by the user and the kernel is not aware of it.
2. These are faster to create and manage.
3. It is implemented using user-level libraries and not by system calls.
4. Each process has its own private thread table to keep the track of the threads.

Kernel Level Threads:

1. The threads are created and implemented using system calls.
2. The kernel knows about the thread and is supported by the OS.
3. The thread table is not present here for each process. The kernel has a thread table to keep the track of all the threads present in the system.

**Difference between Process and Thread**

| Parameter | Process | Thread |
| --- | --- | --- |
| Definition | Process means a program is in execution. | Thread means a segment of a process. |
| Lightweight | The process is not Lightweight. | Threads are Lightweight. |
| Termination time | The process takes more time to terminate. | The thread takes less time to terminate. |
| Creation time | It takes more time for creation. | It takes less time for creation. |
| Communication | Communication between processes needs more time compared to thread. | Communication between threads requires less time compared to processes. |
| Context switching time | It takes more time for context switching. | It takes less time for context switching. |
| Resource | Processes consume more resources. | Thread consume fewer resources. |
| Treatment by OS | Different processes are tread separately by OS. | All the level peer threads are treated as a single task by OS. |
| Memory | The process is mostly isolated. | Threads share memory. |
| Sharing | It does not share data | Threads share data with each other. |

**CPU Scheduling**

CPU scheduling is the way of deciding which process will own the CPU to use while another process is suspended.

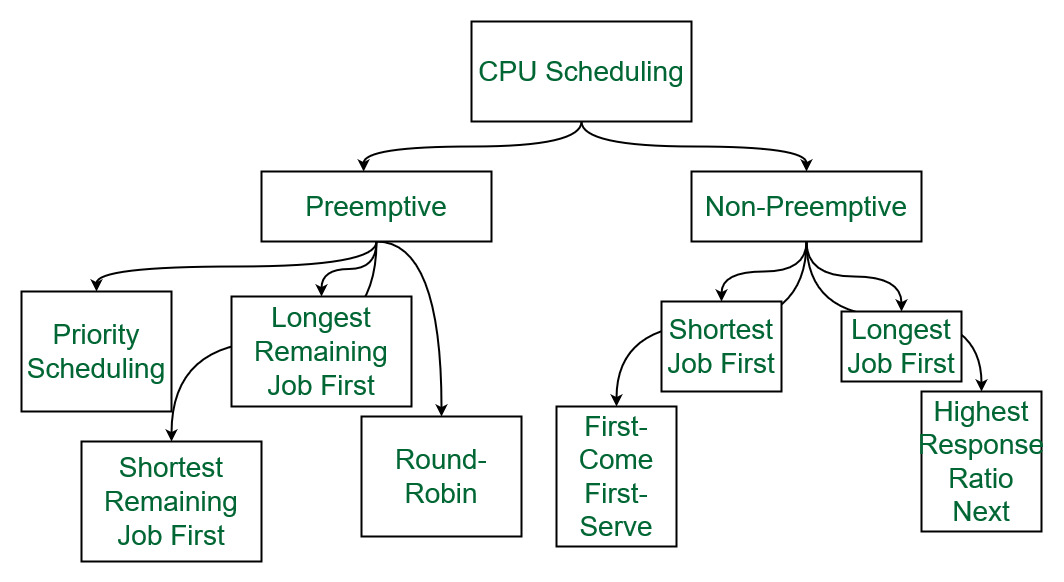
Arrival Time: Time at which the process arrives in the ready queue.

Completion Time: Time at which process completes its execution.

Burst Time/Execute Time: Time required by a process for CPU execution.

Turn Around Time: Time Difference between completion time and arrival time.

Waiting Time: Time Difference betweenTurn Around Time and Burst time.



**Preemptive Scheduling**: it is used when a process switches from Running state to ready state or waiting state to the ready state.

**Non Preemptive Scheduling**: It is used when a process terminates, switching from running to waiting state.

**First Come First Serve(FCFS)**:

1. Jobs are executed on a first come, first serve basis.
2. It is a non-preemptive, preemptive scheduling algorithm.
3. Easy to understand and implement.
4. Its implementation is based on the FIFO queue.
5. Poor in performance as average wait time is high.

**Shortest Job First(SJF):**

1. Its implementation is based on the smallest execution time (burst time).
2. It is a scheduling policy that selects the waiting process with the smallest execution time to execute next.
3. It is a Greedy Algorithm (It is an approach for solving a problem by selecting the best option available at the moment).
4. Best approach to minimize waiting time.

**Shortest Remaining time:**

1. It is a preemptive version of the SJF algorithm.
2. The process with the smallest amount of time remaining until completion is selected to execute.
3. It is faster than SJF.

**Longest Job First(LOF):** it is opposite to shortest job scheduling.

**Priority Scheduling**:

1. It is a method where processes are scheduled based on preference.
2. Processes with higher priority are executed first and processes with equal priority are executed based on FCFS or round-robin scheduling algorithm.
3. Priority can be decide based on memory requirements, time requirement or any other requirement.

**Round robin scheduling**:

1. Each process is provided a fixed time to execute, it is called quantum.
2. It is an algorithm where each process is cyclically assigned a fixed time slot.
3. It focuses on time sharing technique.

**IPC (Inter Process Communication)**

It is the way by which multiple processes can communicate with each other.

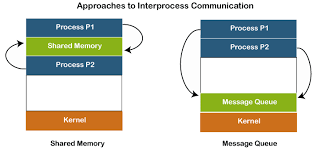
Systems can have 2 types of processes.

1. Independent Process: It is not affected by the execution of the process.
2. Cooperating Process: it is affected by each other and shared data.

IPC is a mechanism that allows processes to communicate with each other and synchronize their action.

Approaches for Inter process communication

1. Pipes
2. Shared memory
3. Message queue
4. Message passing
5. FIFO
6. Direct Communication
7. Indirect Communication



**Pipes**:

1. It is a half-duplex method( one-way communication) used for IPC between two related processes.
2. It is like a scenario like filling water with a tap into a bucket. The filling process is writing into the pipe and the reading process is retrieved from the pipe.

**Shared Memory:**

1. Communication between processes using the shared memory requires processes to share some variable.
2. Multiple processes can access a common shared memory.
3. Multiple processes communicate by shared memory, where one process makes change at a time and then others view the change.
4. Shared memory does not use kernels.

**Message Passing:**

1. Communication between processes via message passing to each other.
2. It is not using any kind of shared memory.
3. If two processes P1 and P2 want to communicate with each other, the proceed as follow:

Establish a Communication link.

Start exchanging messages using - send(message, destination) [ send(“hello”, P2)]

receive(message, host)

1. It is slower than shared memory technique.

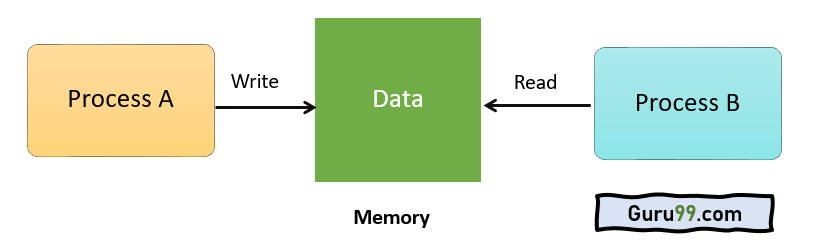
**Message Queue:**

1. We have a linked list to store messages in a kernel of OS and a message queue is identified using “message queue identifier”.

**Synchronization techniques**

The main objective of process synchronization is to ensure that multiple processes access the same shared resources without interfering with each other, and to prevent the possibility of inconsistent data due to concurrent access.

It is specially needed in a multi-processes system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or data at the same time. To deal with this problem, the processes need to be synchronized with each other.



**Sections of Program:**

Entry section: It is part of the process which decides the entry of a particular process.

Critical section: This part allows one process to enter and modify the shared variable.

Exit section: The entry of other processes in the shared data after the execution of one process is handled by the Exit section.

Remainder section: remaining part of program,which is not Critical, exit, entry section.

**Race Condition:** When more than one process is either running the same code or modifying the same memory or any shared data, there is a risk that the result of the shared data may be incorrect because all processes try to access and modify this shared resource.

Thus, all the processes race to say that my result is correct. This condition is called the race condition.

To avoid this race condition- by treating the critical section as a section that can be accessed by only a single process at a time. This kind of section is called an atomic section.

**The Critical Section Problem**: A critical section is a segment of code which can be accessed by a signal process at a specific point of time. The section consists of shared data resources that are required to be accessed by other processes.

1. The entry to the critical section is handled by the wait() function, and it is represented as P().
2. The exit from a critical section is controlled by the signal() function, represented as V().

Rules of Critical Section: there are 3 rules.

1. Mutual Exclusion: If a process is executing in its critical section, then no other process is allowed to execute in the critical section.
2. Progress: If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next, and the selection can not be postponed indefinitely.
3. Bound Waiting: A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

**Solution of The Critical Section**:

1. Peterson Solution
2. Synchronization Hardware
3. Mutex Locks
4. Semaphore Solution

**Peterson Solution**: The solution is based on the idea that when a process is executing in a critical section, then the other process executes the rest of the code and vice-versa is also possible, i.e., this solution makes sure that only one process executes the critical section at any point in time.

In Peterson's solution, we have two shared variables that are used by the processes.

* A boolean Flag[]: A boolean array Flag which is initialized to FALSE. This Flag array represents which process is which process wants to enter into the critical solution.
* int Turn: A integer variable Turn indicates the process number which is ready to enter into the critical section.

**Synchronization Hardware:** Sometimes the problems of the Critical Section are also resolved by hardware. Some operating systems offer lock functionality where a Process acquires a lock when entering the Critical section and releases the lock after leaving it.

So when another process is trying to enter the critical section, it will not be able to enter as it is locked. It can only do so if it is free by acquiring the lock itself.

**Mutex Locks:** Synchronization hardware is not a simple method to implement for everyone, so a strict software method known as Mutex Locks was also introduced.

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.

**Semaphore:** the semaphore is a variable that can hold only a non-negative Integer value, shared between all the threads, with operations wait(P) and signal (V).

**Wait** operation decreases the value of the semaphore and it is also known as sleep, or down.

**Signal** operation increments the value of the semaphore and it is also known as wake-up.

When the value of semaphore is Zero, any process that performs a wait operation will be blocked until another process performs a single operation.

**Type of Semaphore:**

1. **Binary Semaphore** – This is also known as a mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.
2. **Counting Semaphore** – Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

**Monitors:** we can use a monitor to achieve mutual exclusion among the processes. In other words, monitors are defined as the contractor of the programming language, which helps in controlling shared data access.

It uses Lock and Condition variables to provide higher-level synchronization. It can notify waitting threads when a condition becomes true.

**Memory Management**

The memory can be defined as a collection of data in a specific format. It is used to store instruction and process data.

To achieve a degree of multiprogramming and proper utilization of memory, memory management is important.

Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

In other words, Memory management moves processes from primary memory to secondary memory and vice versa.

1. Allocate and de-allocate memory before and after process execution.
2. It keeps track of the status of each memory location, whether allocated or free.

**Fragmentation:** Fragmentation occurs when most free blocks are too small/large to satisfy requests perfectly. There are 2 types of fragmentation.

External fragmentation: we have a free memory. But we can not assign it to process because blocks are not contiguous (block size is not enough)

Internal fragmentation: when memory blocks are allocated to the process more (larger than process size) than their requested size.

Memory management Techniques:

1. **Segmentation**: Segmentation is a method of dividing the primary memory into multiple blocks. Each block is called a segment which can be allocated to a process and has a specific length. The corresponding segments are loaded into main memory when the process in executed.

2. **Paging:** in this secondary memory is divided into fixed-size blocks called pages, and main memory is divided into fixed-size blocks called frames. The frame has the same size as that of a page.

The processes are initially in secondary memory, from where the processes are shifted to main memory when there is a requirement.Each process is mainly divided into parts where the size of each part is the same as the page size.

3. **Swapping**: It is efficient management of the memory. Swapping includes two tasks, swapping-in, swapping-out. When the process is to be executed then the process is swapping in( swap from secondary to main.). And later on, the process is swapping-out.

**Memory Allocation/ deallocation**

**Memory allocation**: it is a process by which computer programs are assigned memory or space.

To gain proper memory utilization, memory allocation must be allocated efficiently. One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions and each partition contains exactly one process.

Multiple partition allocation

Fixed partition allocation

First Fit Allocation: The first hole(block) that is big enough is allocated to the program.

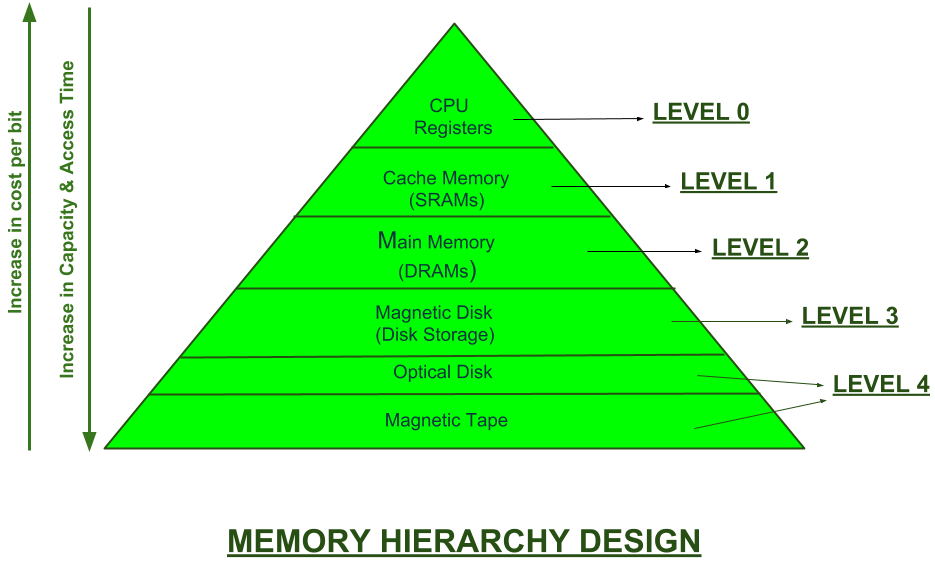
Best Fit Allocation: The smallest hole that is big enough is allocated to the program.

Worst Fit Allocation: The largest hole that is big enough is allocated to the program.

**Memory Deallocation**: it is the process of making a previously allocated block of memory available for reuse. Deallocation generally occurs when the last remaining reference to the block is released.

**Memory Hierarchy**

Memory hierarchy is an enhancement to organize the memory such that it can minimine the access time.



Mainly memory is divided into 2 parts.

**External Memory or Secondary Memory** – Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.

**Internal Memory or Primary Memory** – Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

There are typically four levels of memory in a memory hierarchy:

**Registers**:

1. Registers are small, high-speed memory units located in the CPU.
2. They are used to store the most frequently used data and instructions.
3. Registers have the fastest access time.
4. the smallest storage capacity, typically ranging from 16 to 64 bits.

**Cache Memory**:

1. Cache memory is a small, fast memory unit located close to the CPU.
2. It stores frequently used data and instructions that have been recently accessed from the main memory.

**Main Memory**:

1. Main memory, also known as RAM (Random Access Memory), is the primary memory of a computer system.
2. It has a larger storage capacity than cache memory, but it is slower.
3. Main memory is used to store data and instructions that are currently in use by the CPU.

**Secondary Storage**:

1. Secondary storage, such as hard disk drives (HDD) and solid-state drives (SSD),etc.
2. It is a non-volatile memory unit that has a larger storage capacity than main memory.
3. It is used to store data and instructions that are not currently in use by the CPU.
4. Secondary storage has the slowest access time.
5. it is typically the least expensive type of memory in the memory hierarchy.

**Virtual memory:**

Virtual memory is a part of secondary storage that acts and gives the feel as if it is a part of the main memory. Virtual Memory allows a system to execute heavier applications or multiple applications simultaneously without exhausting the RAM(main memory).

1. Virtual memory in OS works mainly by transferring processes between the computer's RAM and hard disk depending on the requirements.
2. Page Frames are used to structure physical memory.
3. Page tables are used to record the mapping between virtual and physical memory.
4. When there isn't enough space in the main memory to store the applications, certain applications can be swapped out of the RAM to the hard disk.
5. It increases the degree of multiprogramming, effective CPU Usage, and effective memory that can be used, and eases data sharing without the need for relocation.
6. Virtual Memory can make a system slower, reducing hard disk space and system stability.