OPERATING SYSTEM

There are various types of Operating Systems.

1:SIMPLE BATCH OPERATING SYSTEM : central idea is to maximize processor utilization .

- The central idea behind simple batch systems uses software known as monitor, which interfaces with the processor directly and refrains users from accessing the processors directly.
- The user fed an input to the computer operator, who gathered similar jobs(tasks to be performed) and placed an entire batch (group of similar type tasks) for utilization by the monitor. Each program is built to branch back to the monitor when it completes processing.
- processor times alternates between the execution of the user's program and the execution of the monitor.
- Now some main memory is given to the monitor, and the monitor consumes some processor time. Both of these are some forms of overhead. Despite this overhead, simple batch systems improve the utilization of the processor.

  MULTIPROGRAMMED BATCH SYSTEMS :

  - Processor is often idle in batch system
  - The problem is I/O devices couldn't compete with the processors in terms of speed. Thus I/O devices are slow.
  - Main idea :one job/task needs to wait for I/O, OS could switch to a different task/job and start executing it; when the I/O devices finish their execution, then again, it switches back to the first job and resumes its execution where it left off.
  - No user interaction till this point

    TIME SHARING SYSTEMS:

    - It has been developed to provide user interaction directly with the computer

- At each clock interrupt, the Operating System reclaims control and could assign the processor to another user. This technique is known as time slicing.
- To retain the old user program status for later continuation, the old user programs and data were saved to disk before the new user programs and data were read.
- The motive of time-sharing systems was to allow many users to share the computer resources which means max utilization of resources.

  REAL TIME OPERATING SYSTEM :
  - Places where time for input-output requests is very small like military softwares, space softwares etc.

DISTRIBUTED OPERATING SYSTEMS:

- interconnected with each other communicate with each other through the shared communication network. Each machine has its own processors and memory unit for fast computation.

FEATURES OF OS:

- Program development : os comes up with utility programs which allows users to create programs for example : debugger , Editor etc.
- Program execution :A program execution consists of various steps in which the Operating System handles for us, such as loading the program into the main memory, linking external files, initializing files and I/O devices, and other required resources.
- Accessing Input/Output Devices . Operating System hides these details and provides an interface through which user programs can easily access these devices using easy writes and reads.
- Providing protection mechanisms to control access to the files with multiple users

- System access:The access functions must ensure data protection from unauthorized users and resolve resource access conflicts during concurrent access
- The Operating System must handle all these errors and respond that clears all the error conditions with the most negligible impact on running applications in each case.
- Accounting : gathering usage statistics for future enhancement and tuning the system to improve the performance

  FUNCTIONS OF OS :
  - Providing a user interface in which user can easily interact
  - File management :manages all the files, directories, and folders on a system. It can be manipulated via user command such as renaming a file, changing a directory location, modifying a file access privilege/permission, and so on.
  - Memory Management :It is the process that is responsible for assigning/allocating memory blocks (memory spaces) to several running applications and coordinating with the computer's main memory to optimize the overall performance of a system.
  - Security Management :The Operating System supports a variety of protection and security mechanisms. In general, we are concerned with controlling access to computer systems and the information stored in them
    Availability , confidentiality, data integrity ,authenticity

- Resource management : managing the fairness (use of a specific resource should be given equal and fair access. Trying to make allocation and scheduling to meet the entire set of requirements . Doing processes such that their is maximum throughput

Disadvantages of Operating System

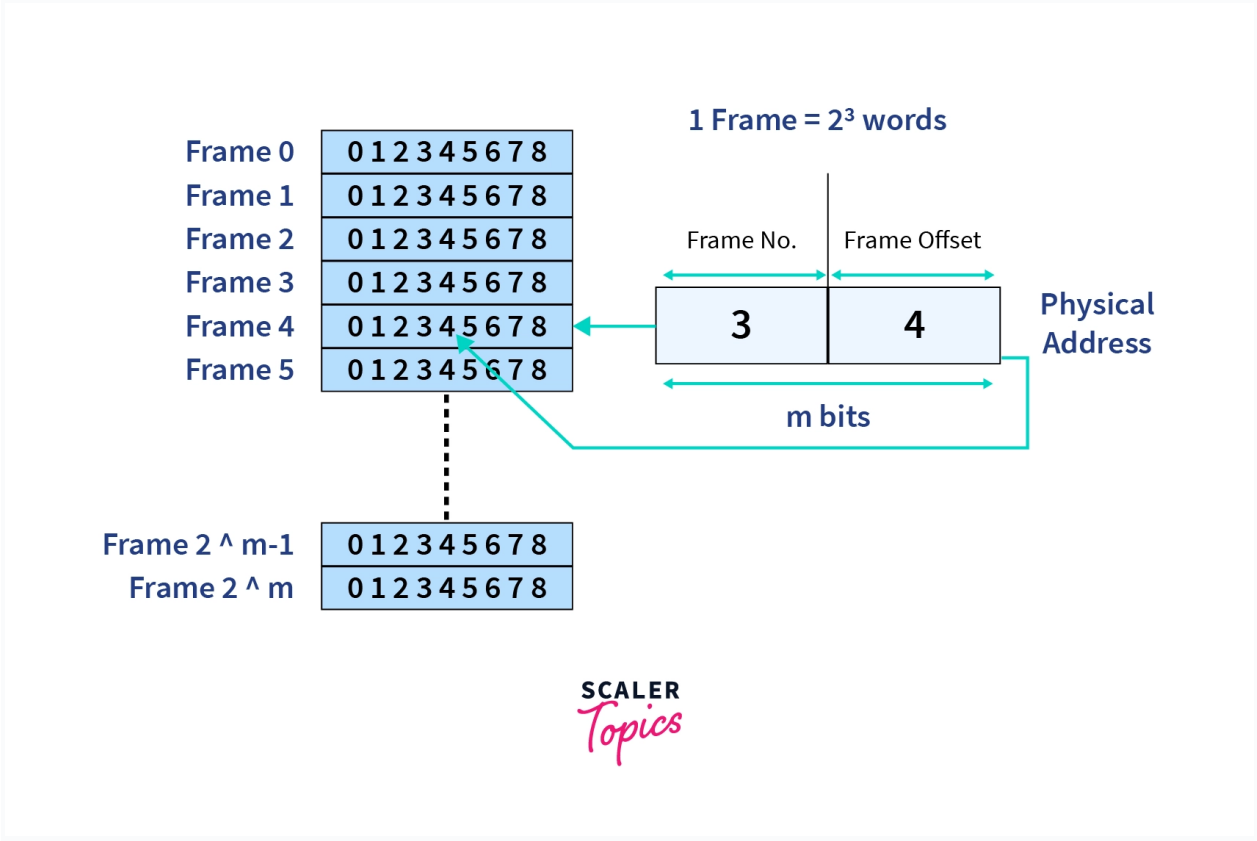
- It has broadened memory access times, for example, page table query.
- Need improvement with utilizing Translation Lookaside Buffer
- It required protected page tables. Need more memory for the memory board.
- The language used behind the Operating System makes it more complex.
- Page Table Length Register needs to be bound with virtual memory size.
- It required more significant improvement in staggered page tables and variable page sizes.
- Some Operating Systems are really costly which makes them more expensive.
- Fragmentation is another disadvantage of the Operating System.(programs cant be assigned to the memory blocks due to their small sizes)

MEMORY MANAGEMENT /VIRTUAL MEMORY:

- Keeping note of where what is stored : page table
- Virtual memory is a part of the system's secondary memory that acts and gives us a 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 (Random Access Memory).
- When running multiple heavy applications at once, the system's RAM may get overloaded. To mitigate this issue, some data stored in RAM that isn't being actively used can be temporarily relocated to virtual memory. This frees up RAM space, which may then be utilized to store data that the system will need to access.
- Assume that an operating system uses 500 MB of RAM to hold all of the running processes. However, there is now only 10 MB of actual capacity accessible on the RAM. The operating system will then allocate 490 MB of virtual memory and manage it with an application called the Virtual Memory Manager (VMM). As a

result, the VMM will generate a 490 MB file on the hard disc to contain the extra RAM that is necessary. The OS will now proceed to address memory, even if only 10 MB space is available because it considers 500 MB of actual memory saved in RAM. It is the VMM's responsibility to handle 500 MB of memory, even if only 10 MB is available.

PAGE FRAME : STRUCTURE PHYSICAL MEMORY :



Cpu can process logical address but main memory cant so memory management converts page number to frame number

PAGE TABLE : maps page number to its frame number .(mapping between logical and physical addresses (logical only read by CPU and physical by RAM)

Physical address = m bits log base 2 (M) {M is the size of physical address space }
Logical address = l bits log base 2 (L){L size of logical address }
Number of entries in page table = no. of pages in process

DEMAND SWAPPING : those programs which are not in use are swapped to virtual memory and those in use are only kept in RAM

Swap In and Swap Out

When the primary memory (RAM) is insufficient to store data required by several applications, we use a method known as swap out to transfer certain programs from RAM to the hard drive.

Similarly, when RAM becomes available, we swap in the applications from hard disk to RAM. We may manage many processes inside the same RAM by using swaps. Swapping aids in the creation of virtual memory and is cost-effective.

IMPORTANT :

- Page size=frame size
- CPU does not know paging concept

- Paging done in main memory
- Every process has its own page table
- Index must start from zero
- Main memory divided into frames and frame stores pages and pages stores the bytes
-

PAGE REPLACEMENT :When a page that is residing in virtual memory is requested by a process for its execution, the Operating System needs to decide which page will be replaced by this requested page. This process is known as page replacement and is a vital component in virtual memory management.

Needed to minimize page faults(accessing a page that is not present in the RAM

PAGE REPLACEMENT ALGO :

1.FIFO : whenever page fault occurs replace it by the oldest page which occurred and add the newly requested page at the rear end of the queue and remove the oldest page from the front end of the queue

The newly called page is added at the last in the queue and takes place of the front page of queue in the main memory

Disadvantage : belady's anomaly occurs .

2: Belady's anomaly : increasing the number of frames in the memory , the number of page faults should decrease but the opposite happens increase in the number of frames increases the page fault as well

- Advantages : simple to understand
- No overhead
- Disadvantages : poor performance

3**. Optimal page replacement** : pages are replaced with the ones that will not be used for the longest duration of time in future.

Pages that will be referred farthest in the future are replaced

Advantages : Excellent efficiency

- Less complexity
- Easy to use and understand
- Simple data structures can be used to implement
- Used as the benchmark for other algorithms

Disadvantages

- More time consuming
- Difficult for error handling
- Need future awareness of the programs, which is not possible every time

Least recently used PGA :

- Works on the principle of locality of a reference which states that a program has tendency to access the same set of memory locations repetitively over a short period of time .
- When a page fault occurs , the page that has been not used for the longest duration of time is replaced by the newly requested page
- In most cases LRU is better than FIFO
- Doesn't suffer Belady’s anomaly
- Disadvantages : more complex,high hardware assistance in required

LAST IN FIRST OUT Algorithm : the newest page is replaced by the requested page . Done through stack cause same number of page faults as optimal page replacement algo therefore **best algo for implementing without prior knowledge of future references**

RANDOM PAGE REPLACEMENT ALGO : chooses a page randomly to be replaced . not the worst performance , depends on the choice of pages chosen at random

BASIC INPUT/OUTPUT SYSTEM : identifies the computer's hardware ,configure the BIOS setup for you , test all workings for you and connect it to the operating system for your further instructions

Gary Kildall, a computer scientist, came up with the term BIOS in 1975, and in 1981 BIOS was integrated into the very first Personal computer of IBM and in those years BIOS established its position and became famous amongst other PCs and BIOS became an essential part of computers for certain years.

However, later when UEFI was introduced in 2007, BIOS's fame diminished because of new technologies.

- USES OF BIOS : mediator between Operating System and the hardware on which they run
- An intercessor between the microprocessor and the I/O devices controlling information and the flow of data
- The OS and its application do not need to take care of the exact details, such as computer hardware addresses, about the attached I/O(Input/Output) devices.
- When the details of the devices changes only the BIOS configuration needs to be changed

After computer is turned on BIOS does several things :

- The Complementary Metal Oxide Setup (CMOS)(small amount of memory on the motherboard that stores the BIOS) is checked by the custom setting.
- Device drivers and interrupt handlers are loaded.
- Registers and power management are initialized.
- Performs the power-on self-test(POST).(whether the keyboard , RAM,disk drives are working properly or not)
- Display system settings.
- Initiate the bootstrap sequence

Steps to configure BIOS setup utility :

- Power off or reset the computer.
- When your system turns back on, search for a message on the screen that says “entering setup” or anything similar to it. The message with the key is displayed which the user needs to press for entering system configuration. For example: “Press the key to enter BIOS setup”. Keys are also used as prompts such as Esc, Del, Tab, and any function keys F1, F2, etc.
- The moment you see the prompt key, press the specified key very quickly because there is a timer running back of the system so your accessing session won’t expire.

After entering the BIOS setup utility , user can : rest the password for set up of the BIOS , hardware settings ,change boot devices

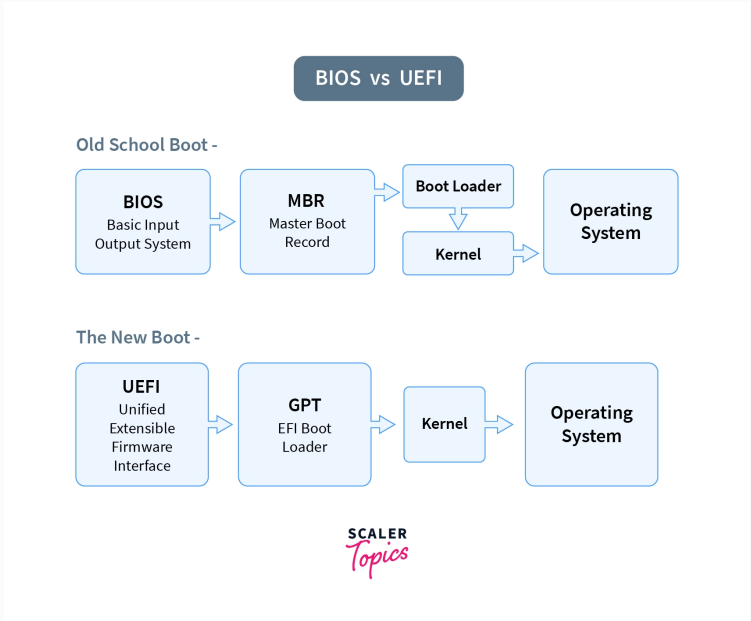
FUNCTIONS OF BIOS :

- POST : power on self test POST is used to test the system hardware when the system starts and to ensure there is no error loading the OS
- BIOS driver :driving basic operational command
- BIOS setup : provide all the configurations to configure hardware settings such as password, date ,time etc;
- Bootstrap loader : responsibility of loading , booting OS flashing , diag and debugging

We can update the BIOS system also

BOOT loader : placing OS into the memory

UEFI : unified Extensible Firmware Interface



DIFFERENCE BETWEEN BIOS AND UEFI

BIOS	UEFI

No Animation	Multi colored with animation
No secure booting option	Secure boot facility to stop loading malicious softwares
Windows Blue Screen	User-friendly graphical user interface
Can’t recognize Ethernet, Wi-Fi, and Bluetooth	Support Ethernet, Wifi, and Bluetooth connectivity
No remote diagnosis and repair support	Support remote diagnosis and repair if the OS won’t boot
Firmware program in 16-bit assembly language	Firmware program in 64-bit C language
Support 2.2 terabytes	Supports drive size up to 9 zettabytes

MASTER BOOT RECORD

First sector of the hard disk determining the location of the operating system to complete the execution of the booting process

CPU SCHEDULING ALGORITHMS

- **Main objective :** to make sure that the cPU is never in the idle state i.e OS has at least one of the processes ready for execution among the available processes in the ready queue .
- **Two types of algorithm : 1.preemptive 2. Non-preemptive**
- **Preemptive** : whenever a higher priority process comes in the lower priority process which has been occupying the CPU is released and the high priority process takes the execution
- Non-preemptive : we cannot preempt the process . Once a process is running in the CPU , it will release it by either context switching or terminating (will happen whole cannot be switched in the middle)
- Arrival time : time at which a process arrives in the ready queue
- Completion time : time at which a process completes its execution
- Burst time : time required by a process for CPU execution
- Turnaround time : completion time -arrival time (interval from the time of submission of a process to the time of completion
- Waiting time = turnaround time -burst time
- Throughput : no. of processes that are completing their execution per unit time

Why do we need scheduling ?
In a multiprogramming system, there can be one process using CPU while another is waiting for I/O whereas, in a uni programming system, time spent waiting for I/O is completely wasted as CPU is idle at this time. The multiprogramming can be achieved by the use of process scheduling.
Objective the scheduling algorithm :

- Maximize the CPU utilization, meaning that keep the CPU as busy as possible.
- Fair allocation of CPU time to every process
- Maximize the Throughput
- Minimize the turnaround time
- Minimize the waiting time
- Minimize the response time

FIRST COME FIRST SERVE (FCFS) SCHEDULING TIME

First come first serve easiest algo to implement . the CPU is first allocated to the process with minimal arrival time will be executed first by the

CPU. **It is non-preemptive scheduling algo as the priority of the processes does not matter ,and they are executed in the manner they arrive in front of the CPU.**

Disadvantages :

- Waiting time for processes with less execution time is often very long

- Poor performance as the average wait time is high
- favors CPU-bound processes then I/O processes
- Leads to convey effect

SHORTEST JOB FIRST :

- Non-preemptive in which the process with shortest burst or completion time is executed first by the CPU.
- Arrival time of the processes must be the same and the processor should know the burst time of all the processes in advance . if two have same burst time then FIFO is used
- Disadvantages : may lead to starvation as shorter processes keeps on coming and longer process never gets chance to run
- Burst time should be known beforehand which is not possible always
- Advantages : more throughput , minimum average waiting time , best approach to minimize waiting time

ROUND ROBIN SCHEDULING ALGO

- Based on first come first serve technique but following a preemptive method .
- A small amount of time is allocated called time slice or time quantum
- CPU is allocated to a given process for the given time quantum and then added back to the queue . if within the time quantum it is completed then it will be preempted otherwise will be added back to the queue and executed when it times comes
- Mostly used for multitasking in time sharing systems on OS having multiple clients
- Advantages : No starvation as no process is left behind
- All processes have equal share of the CPU
- Disadvantages : throughput depends on the length of the time quantum . If the process is too short , it lowers the CPU efficiency as it leads to overhead
- If longer then gives poor response to the short processes and tend to exhibit the same behavior as FCFS
- Average waiting time is usually very high

SHORTEST REMAINING TIME FIRST SCHEDULING ALGO

- Preemptive method for SJF
- Algo is same that job with shorter burst time will be executed first but if a job with shorter burst time arrives then that process is suspended and newer job will be executed first
- At every steps you need to calculate the job with the lowest burst time
- Advantages : better performance than SJF
- Disadvantages : may lead to starvation and requires knowledge of process time beforehand
- Impossible to implement interactive systems

STARVATION OCCURS in :

- Shortest JOb first
- Shortest time remaining first

DEADLOCK PREVENTION : it involves resources needed by two or more processes at the same time that cannot be shared

Necessary conditions for deadlock :

1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait.

Ways to handle deadlock :

- Deadlock Prevention : possibility of deadlock is excluded before making a request i.e elimination of one of the necessary condition for deadlock
- Deadlock avoidance : Any request that can cause deadlock is avoided so any request which may lead to deadlock is not granted
- Deadlock detection and recovery : OS detects deadlock by regularly checking system state, and recovers to safe state using recovery techniques

Deadlock Prevention Techniques

Deadlock prevention techniques refer to violating any one of the four necessary conditions. We will see one by one how we can violate each of them to make safe requests and which is the best approach to prevent deadlock.

1. Mutual exclusion :Mutual exclusion means that unshareable resources cannot be accessed simultaneously by processes.

Not possible to remove mutual exclusion as some processes are non shareable

Spooling : In spooling, when multiple processes request for the printer, their jobs (instructions of the processes that require printer access) are added to the queue in the spooler directory.(fcfs followed)

The printer is allocated to *jobs* on a first come first serve (FCFS) basis. In this way, the process does not have to wait for the printer and it continues its work after adding its *job* in the queue.

2. Hold and wait :Hold and wait is a condition in which a process is holding one resource while simultaneously waiting for another resource that is being held by another process.

   Can remove either waiting or holding conditions i.e waiting by specifying earlier the resources used to process earlier and then allocating and holding by releasing previous resources before getting new one .

   Disadvantages :

- As a process executes instructions one by one, it cannot know about all required resources before execution.
- Releasing all the resources a process is currently holding is also problematic as they may not be usable by other processes and are released unnecessarily.

3. Preemption :Preemption is temporarily interrupting an executing task and later resuming it.

Ways to eliminate it :

- If a process is holding some resources and waiting for other resources, then it should release all previously held resources and put a new request for the required resources again. The process can resume once it has all the required resources.
- If a process P1 is waiting for some resource, and there is another process P2 that is holding that resource and is blocked waiting for some other resource. Then the resource is taken from P2 and allocated to P1. This way process P2 is preempted and it requests again for its required resources to resume the task.
- If a process P1 is waiting for some resource, and there is another process P2 that is holding that resource and is blocked waiting for some other resource. Then the resource is taken from P2 and allocated to P1. This way process P2 is preempted and it requests again for its required resources to resume the task.

CIRCULAR WAIT : two or more processes wait for resources in a circular order.

FEASIBILITY OF DEADLOCK PREVENTION :

1. Mutual exclusive cannot be eliminated completely
2. Hold and wait cannot be eliminated because we cannot know in advance always
3. Eliminating circular wait is the only practical option

INTERPROCESS COMMUNICATION

IPC provide a mechanism to exchange data and information across multiple processes which might be single or multiple computers connected by a network

Why IPC ?

1. Information and data sharing
2. Computational speedup
3. Privilege separation

Ways to implement IPC ?

PIPES : half duplex way (one way communication) used for IPC between two related processes

//Sample pseudo-code program to implement IPC using Pipe//

Start:

Store any message in one character array (char *msg=”Hello world”)

Declare another character array

Create a pipe by using pipe() system call

Create another process by executing fork() system call

In parent process use system call write() to write message from one process to

another process.

In the child process display the message.

Shared Memory

Multiple processes can access a common shared memory. Multiple processes communicate by shared memory, where one process makes changes at a time and then others view the change. Shared memory does not use kernels.

MESSAGE PASSING

1. This is the process for communication and synchronization
2. Processes can communicate without any shared messages
3. Slower than shared memory technique
4. Two actions : sending and receiving messages

MESSAGE QUEUES:

1. Linked list to store messages in a kernel of OS and a message is identified using message queue identifier
2. Create a message queue
3. Write into message queue
4. Read from message queue
5. Perform control operations on the message queue

DIRECT COMMUNICATION :

1. Must name the sender and the receiver
2. A link is establishes between every pair of communication processes

INDIRECT COMMUNICATION :

1. Pair of communicating processes have shared mailboxes
2. Link is established between pair of processes
3. Sender process puts messages in front of the mailbox and receiver process takes out the data from the mailbox

FIFO :

1. Used to communicate between two processes that are not related
2. Full duplex method(bi-directional) - P1 is able to communicate with P2 and vice versa

TERMS USED IN IPC :

To maintain the consistency of data when a number of processes is accessing the same data we synchronize the data , common methods to achieve that :

1. Semaphore : A variable that manages several processes access to a shared resource . example :Binary and counting semaphores

2. Mutual exclusion or mutex : term used to describe a situation where only one process or thread is able to enter the crucial part due to mutual exclusion . (avoiding race condition- when a system attempts to perform two or more operations at the same time)

message size can be fixed as well as variable sized

IPC allows process to communicate and synchronize data even when they are not in the same address-space

PROCESS CONTROL BLOCK

Data structure that stores information of a process

RAM divided into two parts : kernel space and user space

- PCB are stored in a special memory in the kernel space
- It is unique for every process which consists of various attributes such as process ID , priority , registers , program counters , process states , list of open files etc.
- Whenever a user triggers a process(like print command) , a PCB is created which stores the information for the process used by the

operating system to execute and manage the processes when the operating system is free

STRUCTURE OF PCB

- PROCESS ID :

1. unique id assigned to each of the pcb to identify the process stored in it by the OS.

2. If N processes then id - (0 to N-1)

3. If each PCB acquires X bytes and there are N pcb then N*X bytes will be reserved for the PCB .
4. Whenever a process is triggered a free slot is generated which has same slot number as the Process id which helps the OS to identify the free slots available so to check whether new processes can be created or not

Number of processes executed by CPU at a time is 1

- PROCESS STATE : 5 states process

1. New state : contains processes which are ready to be loaded in the main memory
2. Ready state : contains both processes those already present in the RAM or are ready for execution
3. Running state : processes which are currently executed by the CPU
4. Block or Wait : a process may transition from running to block state depending upon the scheduling algo or the internal behavior of the process.
5. Termination : A process that completes its execution comes to a termination state . All content will also be deleted from the OS .

- PROCESS PRIORITY : Process priority is a numeric value that represents the priority of each process. The lesser the value, the greater the priority of that process. This priority is assigned at the time of the creation of the PCB and may depend on many factors like the age of that process, the resources consumed,. User can externally also assign the priority

- PROCESS ACCOUNTING INFORMATION : gives information of the resources used by that process in its lifetime
- PROGRAM COUNTER : pointer points to next instruction to be executed in the process
- CPU REGISTERS : whenever a CPU switches from one process to another , it needs to store the state of the process in memory to resume execution at some later point (context switching). This information is stored temporarily in the CPU registers . These CPU registers are stored in the virtual memory .
- PCB POINTER: contains address of the next PCB which is in the ready state
- LIST OF OPEN FILES : contains information on all the opened files used by that process . helps the OS to close all the opened files at the termination level.
- PROCESS I/O INFORMATION : list of all the input output devices required by that process during execution

PCB STORAGE : stored in the form of the linked list in memory

PROCESS TABLE : contains : 1. Process id and its corresponding PCB

PCB DOES NOT CONTAIN BOOTSTRAP PROGRAM

SYSTEM CALLS: provides an interface so that the process can communicate with the operating system

- When process needs to perform certain actions which needs to be carried out by the OS , process needs an interface to interact with the kernel which is system calls
- Two modes : user code and kernel mode
- User mode : no direct access to the hardware
- Kernel mode : direct access to the hardware (privileged mode)(kernel works on the behalf of the user)
- If program crashes in kernel mode , our system will crash so not very safe
- These calls are generally routine written in C and C++

- System call is executed in the kernel mode using priority basis
- Types of system calls :
    1. Process control : handles process creation , deletion ,example: load , abort , execution
    2. File management : file manipulation events like creating , deleting , reading
    3. Device management : request the device , release the device , logically attach and detach the device
    4. Information maintenance : information about the system like day and time etc.
    5. Communication : send or receive the message, create or delete the communication connections,to transfer status information

Types of System Calls	Windows	Linux
	CreateProcess()	fork()

Process Control	ExitProcess()	exit()
	WaitForSingleObject()	wait()
	CreateFile()	open()
	ReadFile()	read()
File Management	WriteFile()	write()
	CloseHandle()	close()
	SetConsoleMode()	ioctl()

Device Management	ReadConsole()	read()
	WriteConsole()	write()
	GetCurrentProcessID()	getpid()
Information Maintenance	SetTimer()	alarm()
	Sleep()	sleep()
	CreatePipe()	pipe()
Communication	CreateFileMapping()	shmget()

	MapViewOfFile()	mmap()

Trap instruction is used which causes the transfer of control from user mode to kernel mode.

RULES FOR PASSING PARAMETERS IN SYSTEM CALLS :

1. The floating-point parameters can't be passed as a parameter in the system call.
2. Only a limited number of arguments can be passed in the system call.
3. If there are more arguments, then they should be stored in the block of memory and the address of that memory block is stored in the register.
4. Parameters can be pushed and popped from the stack only by the operating system.

IMPORTANT SYSTEM CALLS USED IN OS :

1. WAIT(): when the current process needs to wait for the another process to complete its task
2. fork(): creates a copy of the process which has called it . parent- which has called fork and the copy is the child
3. exec():running process wants to execute another executable file . process id remains same while the other resources used the process are replaced by the newly created process

4. kill(): terminating a file which its being worked on
5. exit(): when we need to actually require to terminate the file exit is used
6. **System call is made by sending a trap signal to the kernel, which reads the system call code from the register and executes the system call.**

THREADS IN OPERATING SYSTEM

Threads : basic unit of execution

1.sequential flow of tasks within a process

2. Has their own id ,program counter , stack and set of registers

3. One process can be single threaded or multithreaded

4.share code , data and other os with other thread belonging from same process

5. Traditional /heavyweight process- single thread

6. Context switching is much faster in threads as compared to the processes.

6.advantages :

- Responsiveness : allow program to continue running even when it is blocked or performing a lengthy operation
- Resource sharing : code , data shared between threads so many task can be performed at the same
- Economical : better to create multi process then to create multi process
- Utilization of multiprocessor architecture :multithreading can be run on in parallel on different processors

Process	Thread
Processes use more resources and hence they are termed as heavyweight processes.	Threads share resources and hence they are termed as lightweight processes.
Creation and termination times of processes are slower.	Creation and termination times of threads are faster compared to processes.

Processes have their own code and data/file.	Threads share code and data/file within a process.
Communication between processes is slower.	Communication between threads is faster.
Context Switching in processes is slower.	Context switching in threads is faster.
Processes are independent of each other.	Threads, on the other hand, are interdependent. (i.e they can read, write or change another thread's data)
Eg: Opening two different browsers.	Eg: Opening two tabs in the same browser.

TYPES OF THREADS :

1. User level thread :
    - implemented using user level libraries and the os does not recognise these threads

- Faster to create and manage as compare to kernel level threads
- Context switching is faster
- Ex java thread, POSIX thread

2. Kernel level thread : implemented and managed by OS
    - Implemented using system calls
    - Slower to create and manage compare to user level threads ex. Windows solaris

ISSUES WITH THREAD :

- During a fork() call issues is whether the whole process should be duplicated or just the thread which makes the fork() call should be. The exec() call replaces the whole process that called it including all the threads in the process with a new program.
- Thread cancellation : termination of thread before its completion is called thread cancellation and the terminated thread is called as target thread .

- Two types :Asynchronous Cancellation: In asynchronous cancellation, one thread immediately terminates the target thread.
- Deferred Cancellation: In deferred cancellation, the target thread periodically checks if it should be terminated.
- Signal handling: In UNIX systems, a signal is used to notify a process that a particular event has happened. Based on the source of the signal, signal handling can be categorized as:
    - Asynchronous Signal: The signal which is generated outside the process which receives it.
    - Synchronous Signal: The signal which is generated and delivered in the same process.

PRODUCER CONSUMER PROBLEM(BOUNDED BUFFER PROBLEM)

- Producer : process which is able to produce data/item
- Consumer : process which is able to consume data/item
- Both Producer and Consumer share a common memory buffer. This buffer is a space of a certain size in the memory of the system which is used for storage.

> The producer produces the data into the buffer and the consumer consumes the data from the buffer.

Problems producer-consumer

- Producer-not produce data when shared buffer is full
- Consumer-not consume data when shared buffer is empty
- Access to shared buffer should be mutually exclusive i.e only one process should be allowed to access and make change to it

Solution : semaphores are used (variables used to indicate the no. or resources available in the system at a particular time

Full- track the space filled in the buffer(initialised to 0)

Empty- track the empty space in the buffer (initialised to BUFFE_SIZE)

Mutex-used to achieve mutual exclusion , ensures that at any particular time only the producer or consumer is accessing the buffer

Use signal or wait() operation :

signal():increases semaphores value by 1(V functions , wake-up ,increase or up function called when the process exits the critical section so that other processes are able to access the critical section)

wait(): decreases semaphores value by 1(called as P function , it controls the entry of a process into the critical section .only called before the process enters the critical section)

Semaphores : two types binary and counting

Critical section of code basically changes the value of semaphores

Binary semaphores : value 0 and 1 1-entry allowed 0-not

Goal of semaphore : synchronize data and provide mutual exclusion in the critical section of the code

Semaphores-signal whether process can proceed to the critical section of the code depending on its value

Disadvantages :

- Usage of semaphores may cause priority inversion where the high priority process might get access to the critical section after the low priority processes

Mutex is different than a semaphore as it is a locking mechanism while a semaphore is a signalling mechanism. **A binary semaphore can be used as a Mutex but a Mutex can never be used as a semaphore**

CONCURRENT PROCESSES : those processes that are executed simultaneously or parallely and might or might no dependent on the other processes

PROCESS SYNCHRONIZATION: coordination between two processes that have access to common materials

DIFFERENT SECTIONS OF A PROGRAM :

- Entry : decides the entry of a process
- Critical section : allows and make sure only one process is modifying the shared data
- Exit section : entry of other processes in the shared data after the execution of one process is handled by exit section
- Remainder section : remainder part which is not categorized above

Critical section : ensures data consistency and no race condition occurs

If we remove critical section , we cannot guarantee the consistency of the end outcome

Requirements of synchronization :

- Mutual exclusion : if a process is running in the critical section , no other process should be allowed to run in that section
- Progress : if no process is still in the critical section then any one of the threads should be permitted to enter the critical section
- No starvation : process should not wait forever to enter inside the critical section

Solution to critical section problem :

1. **Peterson's approach :** based on the idea that when a process is executing in the critical section , then the other process executes rest of the code and vice versa

   They are basically two shared variables :

   1. Boolean flag : represents process which wants to enter into the critical solution

2. Int turn : process number which is ready to enter into the critical section

Disadvantages :

- Involves busy waiting
- Solution limited to only 2 processes

SYNCHRONIZATION HARDWARE

Hardware sometimes assists in the solving of critical section issues . some operating systems provides lock feature (mutex lock).As a result , additional processes are unable to access a critical section if anyone process is already using the section.

MUTEX LOCK : Implementation of synchronization hardware is not an easy method so mutex locks were introduced

Lock is set when a process enters from the entry section and its gets unset where the process exits from exit section

SEMAPHORES:

Integer variable , a process can signal a process that is waiting for a semaphore

This differs from a mutex which can be notified only by the shared lock

Binary semaphores : A single binary semaphore is shared between multiple processes.

- When the semaphore is set to 1, it means some process is working on its critical section, and other processes need to wait, and if the semaphore is set to 0, that means any process can enter the critical section.

SUMMARY:

- Synchronization is the effort of executing processes such that no two processes have access to the same shared data.
- Four elements of program/data are:
    - Entry section
    - Critical section
    - Exit section
    - Reminder section
- The critical section is a portion of code that a single process can access at a specified moment in time.
- Three essential rules that any critical section solution must follow are as follows:
    - Mutual Exclusion
    - Progress
    - No Starvation(Bounded waiting)
- Solutions to critical section problem are:
    - Peterson's solution

- Synchronization hardware
- Mutex Locks
- Semaphore

USE OF COMMAND INTERPRETER : TO GET AND EXECUTE THE NEXT USER SPECIFIED COMMAND

Which one of the following errors will be handle by the operating system?

a) lack of paper in printer

b) connection failure in the network

c) power failure

d) all of the mentioned

D: ALL OF THE ABOVE

OS IS PLACED IN THE HIGH OR LOW MEMORY(DEPENDING ON THE LOCATION OF THE INTERRUPT VECTOR)

REAL TIME OPERATING SYSTEM : vxworks, QNX, RTLinux

Cascading termination refers to the termination of all child processes if the parent process terminates normally or abnormally

THUNDERING HERD PROBLEM

A large number of processes or threads waiting for an event are awoken when that event occurs, but only one process is able to proceed further.

Disadvantages / problems :

- All waiting processes wakes up and moves to ready queue
- Leads to lot of context switching
- Could lead to starvation

Solution

1. When entering critical section , push into a queue before blocking
2. When exiting the critical section , wake up only the first process in the queue

LOCKS AND PRIORITIES

When a low priority task and high priority task share the same data and have a critical section

Suppose the low priority task gets executed first in the critical section . So , now a low priority task is not able to enter the critical section

This dilemma of having a high priority task waiting for a low priority task to release the lock is called **priority inversion problem**

Solution :making the priority tasks equal to that of high priority

DISADVANTAGES OF MUTEX :

- Use of mutex can lead to starvation

  Result in many context switches and could lead to starvation. This solved using a queue

DISK MANAGEMENT IN OS :

Important built-in utility or functionality provided by the OS which can be used to create , delete, format disk partitions

METHODS TO ACCESS DISK MANAGEMENT IN OS :

1. In Windows Operating System:
   - Using Run Box: Click on Start Button > Click on Run > Type *diskmgmt.msc* > Click OK. The Disk Management will open.
2. In Linux Operating System: There are several commands used to check Disk Partitions and Disk Space on the Linux operating system.

- Some of them are mentioned below: a. fdisk b. sfdisk c. cfdisk d. parted e. df

FUNCTIONS OF DISK MANAGEMENT :

1. Disk Management helps to format disk drives.
2. Disk Management enables the user to rename a disk.
3. Disk Management also enables the user to change the file system of a disk drive.
4. Using Disk Management, the user can assign a Drive Letter to a disk. For example, C Drive or D drive, etc. can be found in Windows File System.

IMPORTANT DIRECTORIES IN THE UNIX FILE SYSTEM :

1. /bin : contains all the executable files commands that may be used by the system
2. /boot: contains all the necessary files for booting or starting the system
3. /dev : location of special or device files
4. /etc : contains system configuration and database
5. /home : directory of particular user of the system
6. /lib: helpful libraries used by the system
7. /root : home directory

FILES ALLOCATION TECHNIQUES :

1. CONTIGUOUS : continuous set of blocks on the disk
2. Linked File aLLocation : files can be scattered anywhere in the memory

3. Indexed File allocation : storing pointers

DISK SCHEDULING ALGO:

Used to manage I/O requests .

1. FCFS :first come first serve . does not provide the best possible service
2. Shortest seek time first : seek time-time taken by the hard disk controller to locate a specific piece of the stored data

   Better than FCFS as it has better response time increases throughput

   Not most optimized as overhead of calculating the seek time in advance

3. SCAN AND C-SCAN algo : moves the disk arm in a particular direction and serves the I/O request coming in its path .
4. C-SCAN : disk do not reverse its direction simply goes to the other end of the disk and starts serving the request from there

   Major disadvantage : long waiting time for the request for the location just visited

5. LOOK AND C-LOOK :In LOOK Algorithm, the disk arm reverses its direction from the last served request. Hence, the LOOK algorithm prevents useless delay in the movement of the Disk to the last.

6. In C-LOOK or Circular-LOOK algorithm, the disk arm does not reverse its direction from the last request. It simply goes to the other end's last request and starts serving the requests from there.
7. LESSER KNOWN : RSS: RSS stands for Random Scheduling, and as the name suggests, RSS selects a random I/O request and serves it.
8. LIFO: LIFO stands for Last In First Out. So, the newest request is serviced before any other request.
9. Operating System improves the performance and efficiency of the computer by using the temporary memory space or buffer inside the main memory
10. SPOOL or Simultaneous Peripheral Operation On-line is an acronym for Spooling. Spooling is a temporary memory space used to store the tasks for a system. The tasks are stored until the system is ready to accept or execute new tasks. Spooling considers the DiskDisk as a large buffer that can store many jobs.
11. Buffering is the storing of data temporarily in the buffer so that the data transfer rate between the sender and the receiver is matched. If the sender's transmission speed is slower than the receiver's, then the buffer is created in the main memory of the

receiver. On the other hand, if the receiver's transmission speed is slower than the transmitter, then the buffer is created in the main memory of the transmitter.