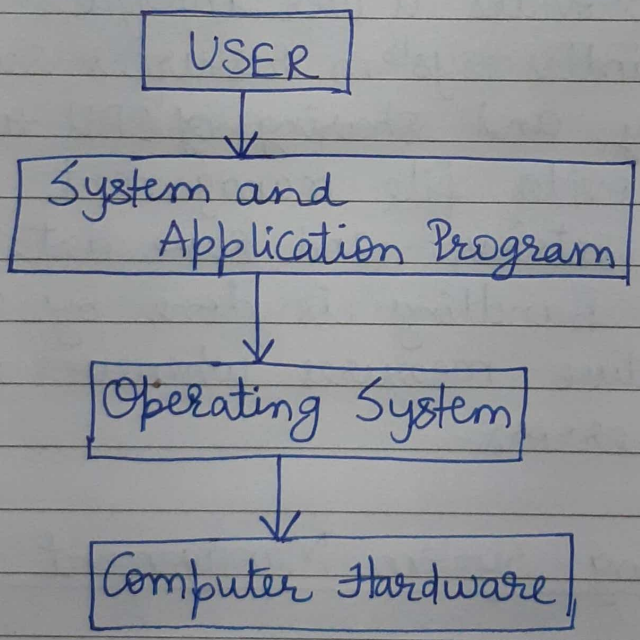
__/__/____

OPERATING SYSTEMS.

→ The Operating System acts as a manager of resources and allocates them to specific programs and users, whenever necessary to perform a particular task. The resources are processor, memory, files and I/O devices.

→ In simple terms, operating system is the interface between the user and the machine.

→ ## Components of a Computer System



→ ## Views of Operating System

1) ### User View

The user view of the computer refers to the interface being used. This system is designed for ease of use.

__/__/____

2> System View

Operating System can be viewed as a resource allocator.

→ **Functions of Operating Systems :-**

1> It boots the computer.
2> It performs basic computer tasks like managing the various peripheral devices eg. mouse.
3> It provides a user interface eg. GUI.
4> It handles system resources such as computer memory and sharing of CPU time by applications
5> It provides file management to manipulate, store, retrieve and save data.
6> Error handling is done by OS. It takes preventive measures whenever required to avoid errors.

→ **Operating System Management Tasks :-**

1> **Processor management**
Putting tasks in sequence and pairing them into manageable size before they go to the CPU

2) Memory Management

It coordinates data to and from RAM.

3) Device Management

Provides interface between connected devices.

4) Storage Management

Directs permanent storage.

5) Applications

Allows standard communication between software and computer.

6) User Interface

Allows communication between user and computer.

→ PROCESS

A process is a program in execution.
Process memory is divided into four sections
i) Text Section → Compiled program code.
ii) Data Section → Global and static variables.
iii) Heap → Dynamic memory allocation.
iv) Stack → Local variables space is reserved here.

__/__/____

→ Different Process States.

1) NEW - Process Created.
2) READY - The process is waiting to be assigned to a processor.
3) RUNNING - Instructions are being executed.
4) WAITING - Process waiting for some event to occur.
5) TERMINATED - Process finished execution.

→ PROCESS SCHEDULING.

The act of determining which process is in the ready state, and should be moved to the running state is known as Process Scheduling.

The prime aim of this System is to keep the CPU busy all the time and to deliver minimum response time for all programs.

→ Scheduling ~~Queues~~ Queues.

- All processes upon entering into the system are stored in Job Queue.
- Processes in the Ready state are placed in the Ready Queue.

__/__/____

- Processing waiting for a device to become available are placed in Device Queue.
- The process could issue an I/O request, then be placed in the I/O queue.

→ SCHEDULERS

1) Long Term Scheduler

Runs less frequently. It decides decide which program must get into the job queue. From the job queue, the Job Processor, selects processes and load into the memory for execution. Job Scheduler maintains good degree of Multiprogramming

2) Short Term Scheduler

This is known as CPU Scheduler and runs very frequently. The primary aim of this scheduler is to enhance CPU performance and increase process execution rate.

3) Medium Term Scheduler.

The scheduler removes the processes from memory and thus reduces the degree of Multiprogramming. Some later time, the process is reintroduced into memory for execution.

__/__/____

→ CONTEXT SWITCH

- Switching the CPU to another process requires saving the state of the old process and loading the saved state for new process. This is known as Context Switch.

- The context of a process is represented in the Process Control Block (PCB) of a process. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.

- It lowers the performance.

→ Operations on Process

1) Process Creation

2) Process Termination

→ CPU SCHEDULER

- CPU Scheduling is a process which allows one process to use the CPU while the execution of another process is on hold due to unavailabi-lity of any resource like I/O etc, thereby

making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

- Whenever the CPU becomes idle, the operating system must select one of the processors in the ready queue to be executed. The selection process is carried out by the short-term scheduler. The scheduler selects among the processes in main memory that are ready to execute, and allocates the CPU to one of them.

- Scheduling Criteria.

1) CPU Utilization
To make out the best use of CPU and not waste any CPU cycle, CPU should would be working most of the time (ideally 100% of time)
CPU usage range → 40% (light) to 90% (heavy)

2) Throughput
The total number of processes completed per unit time OR total amount of ~~time~~ work done in unit time. Eg:- 10/second, 1/hour.

3) Turnaround Time.
It is the amount of time taken to execute a particular process i.e. the interval from time of

submission of the process to the time of completion of the process.

4) Waiting Time.
The sum of the periods spent waiting in the ready queue / amount of time a process has been waiting in the ready queue to acquire control on the CPU.

5) Load Average.
It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.

6) Response Time.
Amount of time it takes from when a request was submitted until the first response is produced.

→ CPU Scheduling Algorithms

1) FCFS (First Come First Served).

2) SJF Scheduling - Shortest Job Scheduling.
It works on the process with the shortest burst time or duration ~~f~~ first.

__/__/____

3) Priority Scheduling.

The scheduling is done on the basis of priority of the process where the process which is most urgent is executed first.

Processes with same priority are executed in FCFS manner.

4) ~~Roud~~ Round Robin Scheduling.

A fixed time is allotted to each process, called quantum for execution.

Once a process is executed for given time period that process is preemptied and other process executes for given time period.

Context switching is used to save states of premptied processes.

5) Multilevel Queue Scheduling

A multilevel queue scheduling algorithm partitions the ready queue into several separate queues.

The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority or process type.

Each queue has its own scheduling algorithm.

→ THREADS

- Thread is an execution unit which consists of its own program counter, a stack, and a set of registers.
- They are also known as Lightweight processes.
- Threads are popular way to improve application through parallelism. The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel

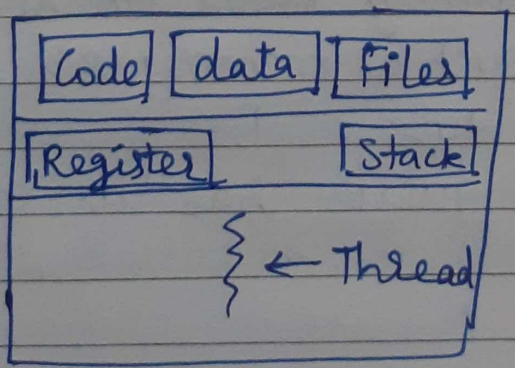
Types of Threads

1) User ~~level~~ threads

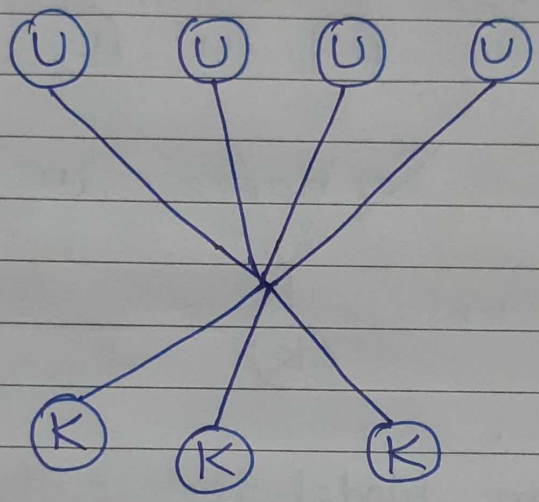
They are used by application programs. They are above kernel and without kernel support.

2) Kernel threads

They are supported within the kernel to perform ~~and~~ multiple simultaneous tasks.



Users can create any no. of threads.
Blocking the kernel system calls does not block the entire process.
Processes can be split across mutiple processors.



→ Process Synchronization.

It means sharing system resources by processes in such a way that concurrent access to shared data is handled thereby minimizing the chance of inconsistent data.

It was introduced to handle problems that arouse while multiple process executions.

→ Critical Section Problem

A Critical Section is a code segment that accesses shared variables and has to be executed as an atomic action.

It means that in a group of cooperating processes, at a given point of time, only one process must be executing its critical section.

If any other process also wants to execute its critical section, it must wait until the first one finishes.

→ MUTEX LOCKS

In the entry section of code, a LOCK is acquired over the critical resources modified and used inside critical section, and in the exit section that LOCK is released.

As the resource is locked while a process executes its critical section part hence no other process can access it.

→ Deadlocks

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.

How to avoid Deadlocks?

1) Mutual Exclusion.
Resources shared such as read-only do not lead to deadlocks but resources such as printers and tape drives, require exclusive access by a single process.

2)

Conditions for Deadlocks :-

1) Mutual Exclusion
It implies if two processes cannot use the same resource at the same time.

2) Hold and Wait
A process waits for some resources while holding another resource at the same time.

__/__/____

3> Non - pre-emption

The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.

4> Circular Wait.

All the processes must be waiting for the resource in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

→ Deadlock ~~Aor~~ Avoidance

- Whenever a database is stuck in a deadlock its better to abort the transactions that resulting deadlock but this is a waste of resources and time.
- Deadlock avoidance mechanism used to detect any deadlock situation in advance

→ Deadlock prevention

- Prevention techniques are used for larger database.
- Resource allocation is done in such a way that deadlock never occurs.
- DBMS analyze the transactions to determine whether any deadlock situation can arise or not, if there

is any possibility then DBMS never allows such transaction of execution.

1> Wait-Die Scheme

T1 and T2 are in deadlock states.
TS(T) is the timestamp of any transaction.

If T1 is older transaction than T2. Then, if
i) TS(T1) < TS(T2)
If T2 is holding a ~~transaction~~ resource then T1 is allowed to wait until resource is available for execution.

~~ii) TS(T1)~~
If T1 is holding a resource and if T2 is waiting for it, then T2 is killed and restarted later with the random delay but with same timestamp.

2> Wound Wait Scheme

Whenever an older transaction request for a resource which is held by a younger transaction, older transaction forces the younger transaction to kill its transaction and release the resource to the older one. After a delay, younger transaction is restarted with the same timestamp.

__/__/____

- Whenever older transaction has held a resource which is requested by the younger transaction, then the younger transaction is asked to wait until older releases it.

MEMORY MANAGEMENT

1> Memory Protection
The main aim of this is to prevent a process from accessing memory that has not allocated to it.

2> Memory Allocation

- First Fit : The first hole is big enough to allocate the program.
- Best Fit : The smallest hole that is big enough to allocate the program.
- Worst Fit : The largest hole that is big enough to allocate the program.

3> Fragmentation
This occurs in dynamic memory allocation system when most of the free blocks are too small to satisfy any request.
It is generally termed as inability to use the available memory.

4) Segmentation
Logical address is converted to linear address.

5) Paging
Linear address is converted to physical address.

→ Semaphores

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

Signals in semaphores :-

1) Wait
It decrements the value of its arguement S, if it is positive. If S is negative or zero, then no operations is performed.

2) Signal
It increments the value of its arguement.

Types of Semaphores :-

__/__/____

1> Binary Semaphore

- Imagine that there are two processes A and B.
- At the beginning the value of semaphore is initialized to 1.
- Now, process A wants to enter the critical section Before it can do that, it checks the value of semaphore which is 1 thus, it can enter the C.S. and semaphore value is turned to 0.
- Now, process B wants to enter too. It checks the semphore value which is 0, thus it can't enter and waits until the value is non-zero positive value.
- Now, Process A finishes and Signals semaphore which in turn changes semaphore value to 1. So, now B can enter C.S.

2> Counting Semaphore

- For Counting Semaphore, we initialize the value of semaphore as the number the concurrent access of critical sections we want to allow.
- Assume that the value is 3. Process 1 enters the Critical Section and ~~semip~~ semaphore value is changed to 2. Process 2 also enters and the value is now 1.
- Process 2 signals semaphore and comes out of critical section and value is 2.

__/__/____

- At this moment, only process 1 is in C.S.
- Process 3 and 4 also enter critical section and value is set to 0.
- Three process are now in C.S:: Processes 1, 3, 4.
- Now, suppose process 5 wants to enter the C.S. It would not be able to enter as semaphore value is 0.
- It can only enter once any of the process 1, 3, 4 signals out of the critical section.