Anshika Gupta OS important topics

Chapter 01: Basics of operating system

An Operating System can be defined as an **interface between user and hardware**. It
is responsible for the execution of all the process scheduling,process synchronization,deadlock handling,memory management,disk management and file management.

Interaction flow:User->Applications->Operating System->Hardware(CPU,I/O Devices/RAM)
e.g. Windows

Functionalities:
//summary of all chapters can be told
1.Resource Manager (e.g. in parallel processing)
2.Process Manager (using CPU scheduling)
3.Storage Manager (using file system in Hard Disk)
4.Memory Manager (by using RAM)
5.Security and Privacy Manager

Swapping

It is a technique of removing a process from main memory and storing it into secondary memory, and then bringing it back into main memory for continued execution.
Purpose:To increase the degree of multiprogramming.

Context Switching: Sending a process back to ready queue and inserting another process into running queue.

Types of Operating Systems :

1. Batch OS:

A set of similar jobs are stored in the main memory for execution.A job gets
assigned to the CPU, only when the execution of the previous job completes.
Disadvantage:CPU sits idle for long.

2. Multiprogramming OS –

The main memory consists of jobs waiting for CPU time. The
OS selects one of the processes and assigns it to the CPU. Whenever the executing
process needs to wait for any other operation (like I/O), only then OS selects another process
from the job queue and assigns it to the CPU. This way, the CPU is never kept idle.

3. Time sharing OS –

Each job is given a fixed time quantum and after this time the job is sent back to ready queue and new job is called to running queue.

4.Multitasking OS:

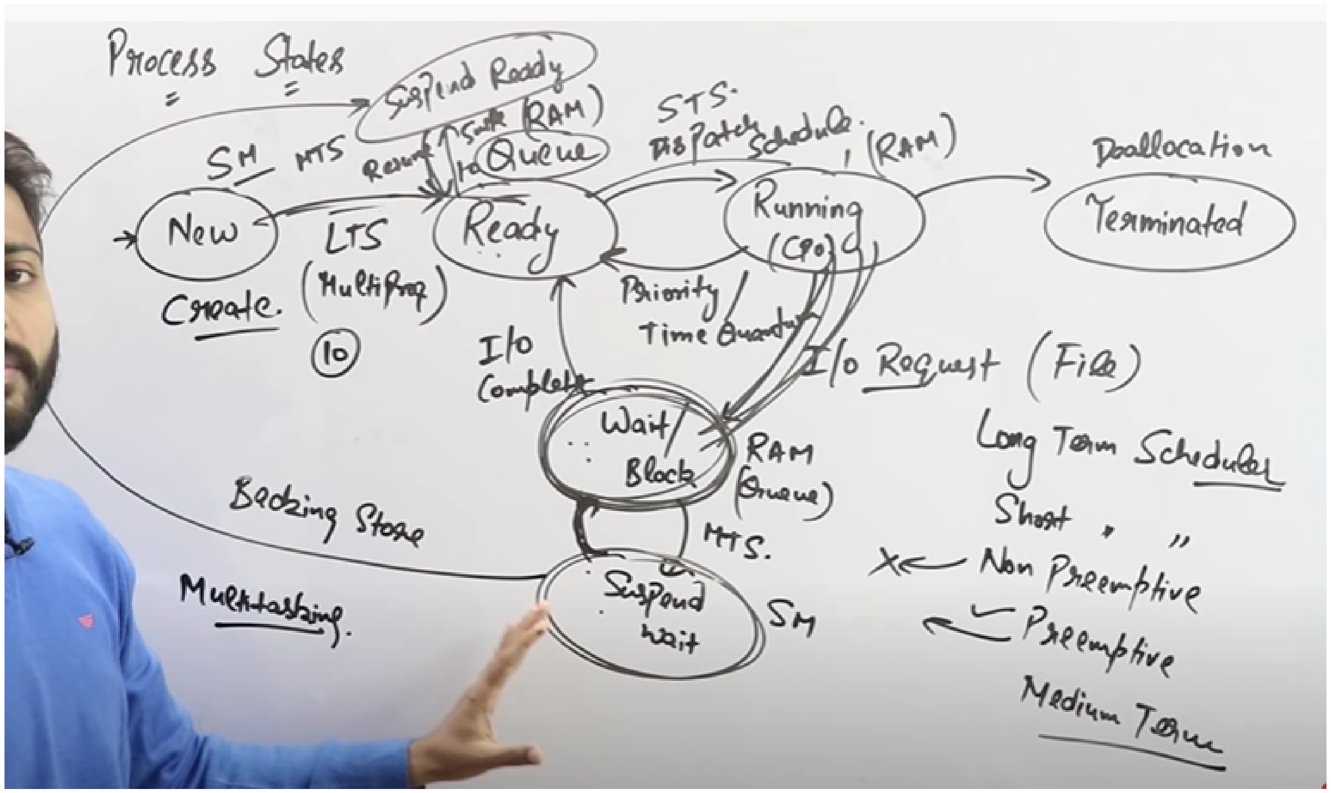
Combine benefits of multiprogramming and time sharing OS i.e. OS performs switch b/w jobs when time quantum allocated to the current program expires OR if it goes on to do I/O.

5. Real Time OS –

Real-Time OS are usually built to accomplish a specific set of jobs within deadlines.
Types:

1.**Hard**: If deadline passes,system fails(processes involved in Missile launch)
2.**Firm**: Have little delay toleration but system doesn’t fail after deadline(e.g. Youtube Live)
3.**Soft**:Have good delay toleration (web browsing)

Process States



Primary Memory(Main)	Secondary memory
1.Temporary	Permanent
2.Directly accessible by CPU	Not directly accessible by CPU

3.Volatile(e.g.RAM) & non-volatile (e.g.ROM) both	Only non-volatile
4.Also known as main memory or internal memory	Also known as external memory
5.**Examples**:RAM,ROM,cache,registers	HDD,SSD,floppy disk

ROM has no use in our concepts .so we can assume RAM=main memory.
→ Volatile:When power turned off,memory lost

Kernel:

Kernel is system software which is part of the operating system.It manages the interaction b/w hardware and software.It works through system calls.

Microkernel	**Monolithic kernel**
In microkernel user services and kernel, services are kept in separate address space.	In monolithic kernel, both user services and kernel services are kept in the same address space.

.

System Calls

System calls are usually made when a process in user mode requires access to a resource i.e. kernel mode.

Types:

1.Process Control
These system calls deal with processes such as process creation, process termination etc.

2.File Management
These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

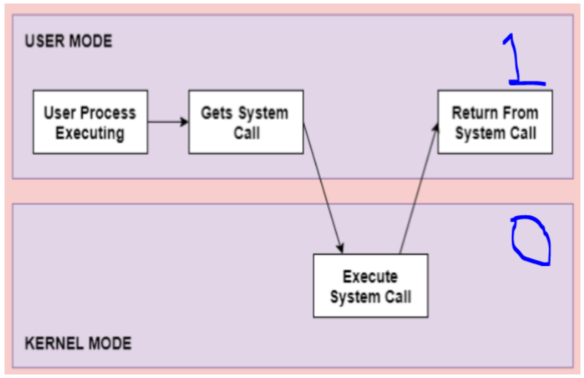
3.Device Management
These system calls are responsible for device/hardware manipulation such as reading from device buffers, writing into device buffers etc.

4.Information Maintenance
These system calls handle information of metadata and its transfer between the operating system and the user program.

5.Communication
These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.

User Mode vs Kernel Mode

A program is initially in user mode .And when a program requests to run (e.g. it requires hardware access),then CPU switches from user mode to kernel mode through system call.Then it executes the system call in kernel mode .After execution of process CPU switches back to user mode.
CPU keeps switching between user mode(mode bit=1) and kernel mode(mode bit=0).



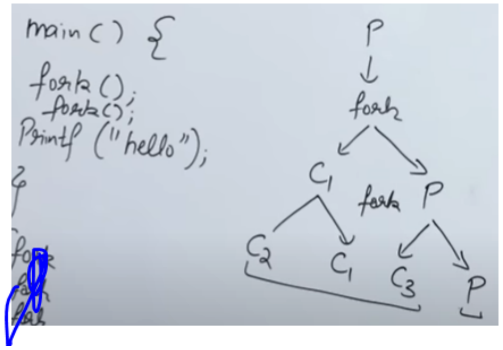
Fork System Call

Fork system call is used for creating a new process, which is called child process.The child process is a clone of the parent process and both child and parent process run simultaneously.
A child process uses the same pc(program counter), same CPU registers, same open files which use in the parent process.

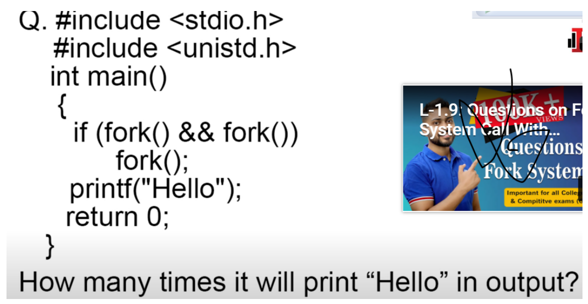
Fork() can return three types of value

1. +ve → for parent
2. 0 → For child
3. -ve → If no child created

A process with n fork() system calls generates $2^n – 1$ child processes and 1 parent.



```
main() {
fork();
fork();
Printf("hello");
}
```



```
Q. #include <stdio.h>
   #include <unistd.h>
   int main()
   {
     if (fork() && fork())
         fork();
     printf("Hello");
     return 0;
   }
```

How many times it will print “Hello” in output?

Considering all possibilities of fork() as child(0) and parent(1),Ans:4
//see in video

Program:
A Program is an executable file which contains a certain set of instructions written to complete the specific job or operation on your computer.It is a passive entity.

Processes :

- A process is an executing instance of a program.A program can have N no. of processes.
- **Process Control Block(PCB)** is a data structure that contains all information related to a process.
- Different processes have different copies of everything i.e. code,data etc.

Thread:

A process gets divided into multiple threads to perform multitasking. A thread is a lightweight process and forms the basic unit of CPU utilization.

● A thread has its own program counter, register set, and stack.(**The program counter (PC)** is a register that manages the memory address of the instruction to be executed next.)
● Unlike process,A thread shares resources with other threads of the same process: the code section, the data section, files and signals

→ There are two types of threads:
● User threads (User threads are implemented by users)
● Kernel threads (Kernel threads are implemented by OS)

Process vs User level thread vs Kernel level thread

Process:

- System call involved
- Different processes are treated as different tasks for the OS
- Context switching is slowest among three(as PCB updation takes time).
- As processes are independent,blocking of one process(for e.g. I/O calling) will not block all processes.

User level thread:

- No system call involved
- All user level threads of same process treated as single tasks for OS
- Context switching is fastest among three

- As user threads are **interdependent**,blocking of one thread(for e.g. I/O calling) will block the entire process.

Kernel level thread:
- System call involved(implemented by OS)
- All kernel level threads of same process treated as single tasks for OS
- Context switching is faster than process but slower than user level.
- As kernel threads are independent,blocking of one thread(for e.g. I/O calling) will not block the entire process.

………..

Chapter 02:Process scheduling

Preemptive vs Non-Preemptive Scheduling

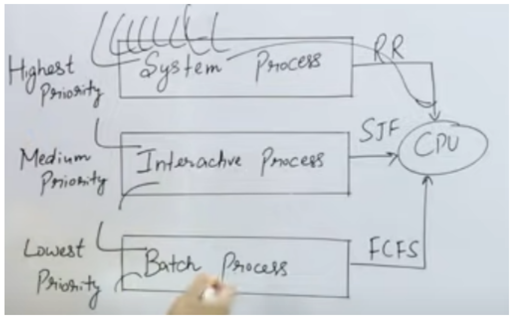
Preemptive: When a process is placed back into the ready queue and will execute in the next chance due to some reason like quantum time or priority
Non-Preemptive : In this a process gone from ready queue to running queue(CPU) never goes back to ready queue.(we can't empty the running queue)

Process Scheduling:
1. Arrival Time – Timestamp at which the process arrives in the ready queue.
2. Completion Time – Timestamp at which process completes its execution.
3. Burst Time – Time duration required by a process for CPU execution. //CPU ne kitni der mei burst kiya process
4. Turn Around Time = Completion Time - Arrival Time //Turn around=total time
5. Waiting Time (WT) =Turnaround Time - Burst Time
6.Response Time (RT)=Timestamp at which process gets CPU for first time- Arrival time .For non-preemptive ,always equal to waiting time.

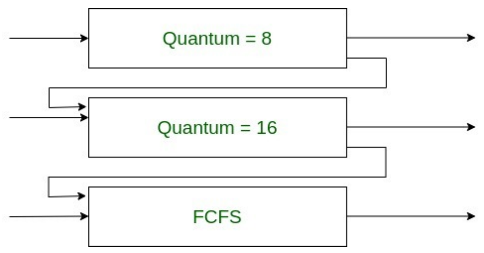
Starvation:When any process in the ready queue has to wait for an indefinite time due to its low priority.
Aging:It is a technique to prevent starvation.It keeps increasing the priority of a process as its waiting time increases.

Scheduling Algorithms :
1. First Come First Serve (FCFS) : Simplest scheduling algorithm that schedules according to arrival times of processes.
2. Shortest Job First (SJF): Processes which have the shortest burst time are scheduled first.
3. Shortest Remaining Time First (SRTF): It is a preemptive mode of SJF algorithm in which jobs are scheduled according to the shortest remaining time.
4. Round Robin (RR) Scheduling (preemptive obviously):
Each process is assigned a fixed time quantum, in a cyclic way.

Make two separate running queue and ready queue.During context switching ,in the ready queue,first insert all new arrivals at time ‘t’ in increasing arrival time then insert the process at last which has come back from CPU.
Avoids starvation but high context switching overhead.
5. Priority Based scheduling (Non Preemptive & preemptive both): In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is scheduled first. If priorities of two processes match, then scheduling is according to the arrival time.
6. Highest Response Ratio Next (HRRN)(Non-preemptive):In this scheduling, processes with the highest response ratio is scheduled. This algorithm avoids starvation.
Response Ratio = (Turn around time) / Burst time
7. Multilevel Queue Scheduling (MLQ): According to the priority of the process, processes are placed(categorised) in the different ready queues.
Each queue can have their own unique scheduling algorithm.
Disadvantage:Since only after completion of processes from the top level queue, lower level queued processes are Scheduled,it may result in starvation of lower priority queue processes.



8. Multilevel Feedback Queue (MLFQ) Scheduling: It allows the process to
move in between queues. The idea is to separate processes according to their burst time.If a process uses too much CPU time, it is moved to a lower-priority queue.And if a process is kept for longer time in lower priority queue ,it can be shifted to higher priority queue.
e.g.



When the process enters Q1 it is allowed to execute and if it does not complete in 8 milliseconds it is shifted to Q2 and receives 16 milliseconds. Again it is preempted to Q3 if it does not complete in 16 seconds

Practice a good scheduling problem of Mix Burst Time (CPU / I/O Both) here.

....

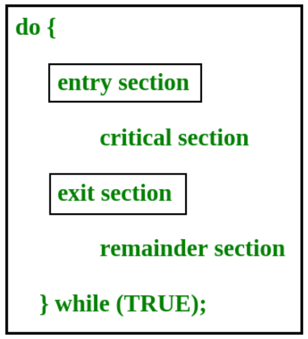
Chapter 3:Process Synchronization:

Process Synchronization is a way to coordinate processes that use shared data

On the basis of **synchronization**, processes are categorized as one of the following two types:

Independent Process : Execution of one process does not affect the execution of other processes because the processes share nothing with each other.
Cooperative Process : Execution of one process affects the execution of other processes as the processes share something (variable,memory,resource,code etc) with each other.

Race Condition
When more than one processes are executing the same code or accessing the same memory or any shared variable in that condition there is a possibility that the output or the value of the shared variable is wrong as all the processes doing the race say that my output is correct, this condition is known as a race condition. Several processes access and process the manipulations over the same data concurrently, then the outcome depends on the particular order in which the access takes place.
Critical Section – The portion of the code in the program which contains shared variables.
These variables need to be synchronized to maintain consistency of data variables.



```
do {
    entry section
        critical section
    exit section
        remainder section
  } while (TRUE);
```

In the entry section, the process requests for entry in the Critical Section

Remainder Section – The remaining portion of the program excluding the Critical Section.

A solution for the critical section problem must satisfy the following three conditions:

1. Mutual Exclusion – If a process Pi is executing in its critical section, then no other process is allowed to enter into the critical section.
2. Progress – If no process is executing in the critical section, then the decision of a process to enter a critical section cannot be made by any of those processes that are executing in its remainder section. Also the selection of the process cannot be postponed indefinitely.
3. Bounded Waiting – There exists a bound on the number of times other processes can enter into the critical section otherwise there is a possibility one process will go again and again and some particular process will never get a chance(resulting in starvation).

Semaphores:
A semaphore is a variable that indicates the number of resources that are available in a system at a particular time.It is used to achieve the process synchronization.
It can be accessed only through two standard operations : wait() and signal().
The wait() operation reduces the value of semaphore by 1 and the signal() operation increases its value by 1.

Two types:

Binary Semaphore	Counting Semaphore
Semaphore value can be 0 or 1 i.e. maximum 1 process can go in the critical section.	Semaphore value can be anything from 0 to N where N=maximum no. of processes that can go in the critical section.
It guarantees mutual exclusion	It doesn’t guarantees mutual exclusion since more than 1 process can go in critical section
It doesn’t guarantees bounded wait	It guarantees bounded wait, since it maintains a list of all the process or threads, using a queue, and each process or thread get a chance to enter the critical section once. So no question of starvation.

Binary Semaphore working:
Let there be two processes P1 and P2 and a semaphore ‘s’ is initialized as 1. Now if suppose P1 enters in its critical section then the value of semaphore s becomes 0. Now if P2 wants to enter its critical section then it will wait until s > 0, this can only happen when P1 finishes its critical section and calls signal operation on semaphore s. This way mutual exclusion is achieved.
Counting Semaphore working:
Suppose there are 4 processes P1, P2, P3, P4, and they all call wait operation on S(initialized with 4). If another process P5 wants the resource then it should wait until any one of the four processes calls the signal function and the value of semaphore becomes positive.

Mutex:

Mutex uses the lock-based technique to handle the critical section problem.
Whenever a process requests for a resource from the system, then the system will create a mutex object with a unique name or ID. So, whenever the process wants to use that resource, then the process occupies a lock on the object. After locking, the process uses the resource and finally releases the mutex object. After that, other processes can create the mutex object in the same manner and use it.

A thread can simultaneously hold any number of mutexes.And a mutex can be locked multiple times iff it is a recursive mutex. The programmer must unlock the mutex as many times as it was locked.

Mutex	**Semaphore**
A mutex is a locking mechanism	Semaphore is signaling mechanism
A mutex is an object .	Semaphore is an integer variable.
In mutex, the lock can be acquired and released only by the same process(called owner of lock) .	The value of the semaphore variable can be modified by any process that needs some resource but only one process can change the value at a time.
Guarantees mutual exclusion	Doesn’t guarantee mutual exclusion(in counting semaphore)

Producer-consumer problem

//Peterson's Solution and TestAndSet not studied because not so important.

…………………………………………………………………………………………………

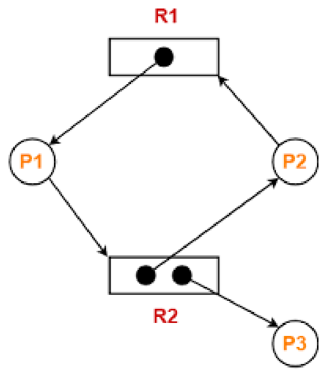
Chapter 04:Deadlock

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

4 conditions required for deadlock

Mutual Exclusion: A resource can only be used by one process at a time.
Hold and Wait: A process in deadlock is holding at least one resource and waiting for resources.
No Preemption: A resource cannot be taken from a process unless the process releases the resource on its own
Circular Wait: A set of processes are waiting for each other in circular form.

Rules for making Resource Allocation Graph (RAG):



Resource Allocation Graph

1.Process vertex – Every process is represented as a **circular** vertex.
2. Resource vertex – Every resource is represented as a **box shaped** vertex. No. of dots inside the box tells how many instances can be there of a resource.

There are two types of edges in RAG –
1. Assign Edge – P1<-R1 denotes that process P1 is holding resource R1.
2. Request Edge – P1->R1 denotes that process P1 is waiting for resource R1

If there is circular wait && all resources are single instance type: always a deadlock
If circular wait && one or more resources are multi instance type: may or may not be a deadlock

(Optional) Practice a question of predicting deadlock here.

Methods for handling deadlock

1. **Ignorance**:Let the deadlock happen and reboot the system.For less critical jobs deadlock are ignored because handling deadlock is expensive (as first they are very rare and second,a lot of codes need to be altered which decreases the performance. Ostrich algorithm is used in deadlock Ignorance. Used in windows, Linux etc.

2. **Prevention:** We design such a system where we eliminate at least one of the four conditions required for deadlock.

    1. Preventing mutual exclusion:All resources must be shared by processes.This is very difficult because some resources are not sharable e.g. printers (hardware limitation).
    2. Preventing Hold and wait: A process will not start i.e. hold resources until it has acquired all the resources.
    3. Preemption:We forcefully deallocate resources from one process and give it to another process.
    4. Removing Circular wait:In this,we assign a number to every resource and the process can request for resources only in the increasing order i.e. if a process resource no. is 4,it can’t request for resources 1,2 or 3.It can do so only if it releases all the high number acquires resources and then make a fresh request.

3.Deadlock detection and recovery: Let deadlock occur, detect it and start killing processes in increasing priority until the deadlock is not removed.

4.Deadlock avoidance: In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system.Implemented using banking algorithm.

Limitations:

1. Each process should know its maximum resource need.
2. During the time of Processing, this algorithm does not permit a process to change its maximum resource need.

→ A system is said to be in safe state if there exists a safe sequence of allocating resources to processes such that deadlock can be removed.
→ To practice numerical of banking algorithms,watch this vedio.

....

Chapter 5:Memory Management

These techniques allow the memory to be shared among multiple processes in an efficient manner.

Fragmentation:

Memory gets wasted either by internal or by external fragmentation.

Internal Fragmentation
When the allocated memory space is more than required memory space,the problem is termed Internal Fragmentation.

Removing internal fragmentation:

1. First fit:Allocate the first hole (starts searching from beginning) that is big enough.
2. Next fit:Allocate the first hole that is big enough but starts searching from last allocated hole.Faster than first fit.
3. Best fit:Allocate the smallest hole that is big enough.Leaves such small holes that can’t be filled.
4. Worst fit:Allocate the largest hole

External Fragmentation
When there are small and non-contiguous memory blocks which cannot be assigned to any process, the problem is termed as External Fragmentation.

Contiguous Memory Allocation :

Each process is contained in a single contiguous segment of memory.

Fixed partitioning:

In this ,RAM is divided into partitions of fixed (may or may not be equal) size in advance.
Disadvantages: Internal & external fragmentation,process size must be limited,degree of multiprogramming is bounded by no. of partitions

Overlay is a technique to run a program that is bigger than the size of the physical memory .It divides the program into modules in such a way that not all modules need to be in the main memory at the same time. Overlays concept says that whichever module you require, you load it and once that is done, then you just unload it,and get the new module you require and run it.

Variable Partitioning:

Unlike fixed partitioning ,partitions are made during the run-time and their size is equal to the incoming process.
Advantages:No Internal fragmentation,no restriction on process size
Disadvantage: External fragmentation(when a process gets completed,it leaves hole),no bound on degree of multiprogramming

Non-Contiguous Memory Allocation :

Main memory is divided into equal sized frames and the process is divided into equal sized pages such that a page(a part of process) can be accommodated in a frame(part of Main Memory).
Frame size = page size.

Pages or frames are numbered starting from zero.Bytes in frames/pages are also numbered starting from zero.

Logical address(virtual) vs physical address
Logical address is generated by CPU and is divided into two parts : **page no.(p) and page offset (d) //offset=size**
Physical address is a location that exists in the memory unit.It is divided into two parts:**frame no.(f) and frame offset(d)**
The mapping from virtual(of processes) to physical address(of main memory) is done by the **memory management unit (MMU)** which is a hardware device and this mapping is known as **paging technique.**

Page table: Each process has its own unique page table .It gives frame no. corresponding to page no. No. of entries in this table will be equal to no. of pages of a process.It is inside main memory.

TLB(translation Look-aside buffer):It is a cache memory which contains page table entries that have been most recently used.
First logical address is searched in TLB and if it is not present there,then only it is searched in the page table.
EMAT = p(c+m) + (1-p)*(c+2m) //* think yourself how this formulae came or read here.
EMAT=effective memory access time,p=probability of TLB hit,m = Memory access time ,c = TLB access time

How to get the physical address of a bit by logical address(p & d are given)?

MMU[p]=f and dth bit of frame no. ‘f’ is our desired bit.

Some Formulae for paging:

Question: Logical address space(LAS)= 4GB,Physical address space=64MB,Page size=4KB.Find no. of pages,no. of frames
,no. of entries in page table & size of page table.

Watch video 5.10 to 5.16 later…

Page Fault

We bring more and more processes in RAM so that the CPU never sits idle (when any process goes to perform I/O operation etc).But since the size of RAM is limited ,we don’t bring the complete process but only some limited no. of pages of each process.

Page fault is a condition when CPU tries to access the page that is not found in main memory.Whenever page fault occurs, then the required page has to be fetched from the secondary memory into the main memory using page fault techniques otherwise a page fault trap arises.

Steps for page fault service:

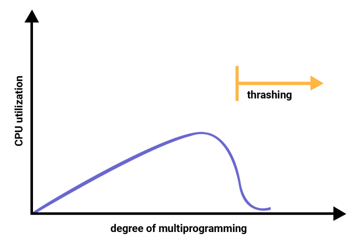
TL,DR; If the page is not found in the page table by MMU,OS searches that page in hard disk and swaps it in RAM and updates the page table.

Step 1:1.CPU asks for a specific page to MMU.
Step 2:The page will not be found in the page table and a trap will be generated(page fault).Here context switching will take place and control will be given from user to OS.
Step 3: OS will search for the page in hard disk
Step 4: That page will be swapped into RAM.
Step 5: Address of that page is updated in the page table.
Step 6: Control will be given back to user from CPU (context switching)

If p=probability that page fault will occur
Effective Memory Access Time=p (page fault service time) +(1-p) (main memory access time)
Page fault service time is in milliseconds and main memory access time is in nanoseconds.

Thrashing

In case, if the page fault and swapping happens very frequently at a higher rate, then the operating system has to spend more time swapping these pages. This state in the operating system is termed thrashing. Because of thrashing the CPU utilization is going to be reduced.

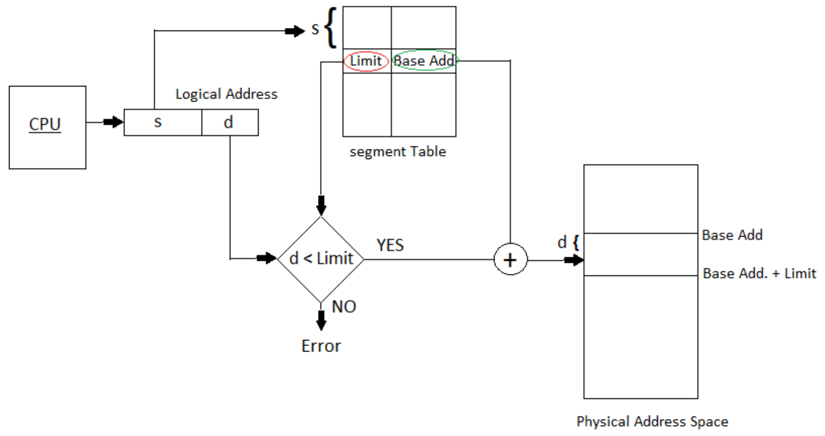


Techniques to handle thrashing:
1.Increase RAM size so that more pages can sit in RAM
2. Decrease degree of multiprogramming.

Demand paging:
It suggests keeping all pages of the frames in the secondary memory until they are required.That is maximum page fault.

Segmentation

In segmentation, the program is divided into variable size sections instead of pages.
In segmentation,the logical address generated by the CPU has two parts:segment no. and segment offset(size).
Segment table:
Input: Segment no.
Output: Base address of segment and its maximum size (in physical memory)
Segment offset (d) (generated by CPU) should be <= size of segment otherwise there will be a trap.



Paging	**Segmentation**
Page size is determined by hardware.	Here, the section size is given by the user.
Faster	Segmentation is slow.
Paging is invisible to the user.	Segmentation is visible to the user.
For paging, the operating system is accountable.	For segmentation ,the compiler is accountable.

Virtual memory

Virtual memory is a memory management technique to compensate for main memory shortages.

Virtual memory frees up RAM by transferring data from it that has not been used recently over to a storage device, such as a hard drive .

Page replacement algorithms

1.First in first out:

Self-explanatory

Belady’s anomaly:

Belady’s anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.

2. Optimal Page replacement –

In this algorithm, that page is replaced which will not be used for the **longest duration of time** in the future.
Optimal page replacement is perfect, but not possible in practice as an OS cannot know future requests.The use of Optimal Page replacement is to setup a benchmark so that other replacement algorithms can be analyzed against it.

3. Least Recently Used (LRU) :

In this algorithm,the page will be replaced with the one which is least recently used.

SPOOLing

It stands for **S**imultaneous **P**eripheral **O**perations **O**n**l**ine.Spooling is a process in which data is temporarily held on the secondary memory to be used and executed later.

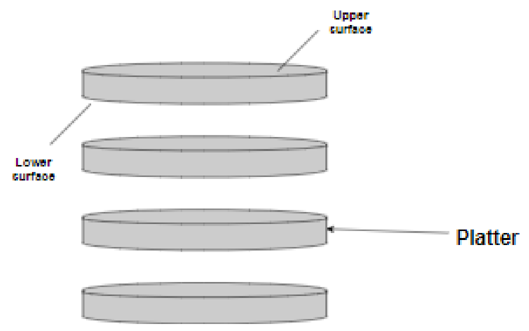
Since time taken in the I/O operation is very large compared to the time taken by the CPU for the execution of the instructions.So if we perform tasks one by one i.e. taking input of one task → performs CPU work → showing output,then repeating same thing for another task.This make CPU idle while taking input or giving output.

For example In printer, the documents/files that are sent to the printer are first temporarily stored in the secondary memory or the spool. Once the printer is ready, it fetches the data from the spool and prints it.

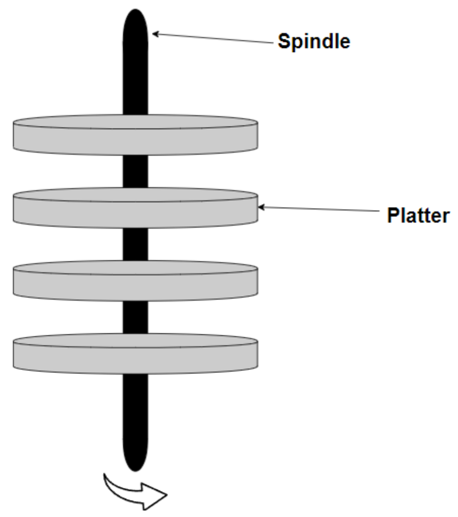
……

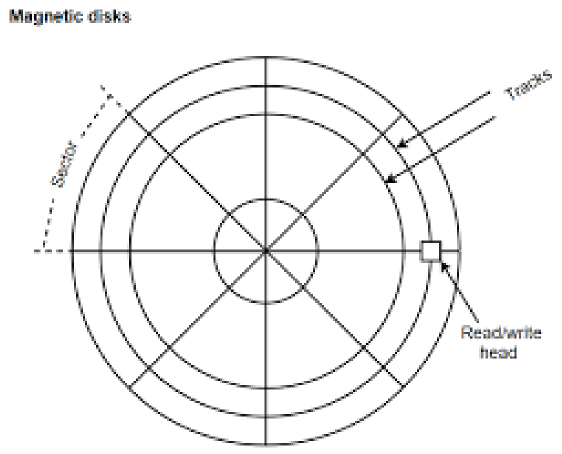
Chapter 6: Disk Management

Disk architecture:



Platters



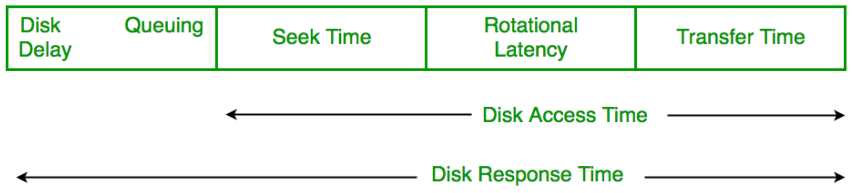


- Platters : A hard disk is a stack of multiple circular disk.Each individual disk is called platter.

- Spindle : The platters within the hard disk are connected by a spindle that runs through the middle of the platters.The spindle moves in a unidirectional manner along its axis which causes the platters to rotate as well.
- Read/write heads :Each surface on a platter contains a read/write head that can move back and forth along the surface of a platter.It is used to read or write data onto the disk.
- Tracks : Each platter is divided into multiple circular tracks.
- Sectors :Each track is further divided into equal no. of sectors.

These platters are divided into tracks(circular) .Each track is further divided into equal no. of sectors.

Seek Time: Seek time is the time taken by R-W (read/write) head to reach a specified track where the data is to be read or written.
Rotational Latency: Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.It is half of rotational speed.
Transfer rate: No. of surfaces * Capacity of one track * rotation frequency



Disk scheduling algorithm

In what order,requests are executed from disk queue.
Tries to minimize seek time.

1.FCFS:

self-explanatory

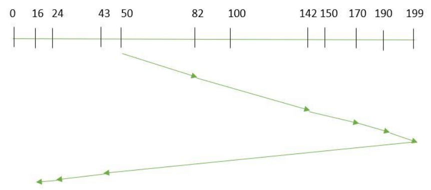
Advantage: No starvation
Disadvantage: Not optimal

2.SSTF(Shortest seek time first):

self-explanatory
Advantage: 1.Optimal
Disadvantage: 1. Can cause Starvation for a high seek-time request
2.Overhead to calculate seek time in advance.

3. SCAN:

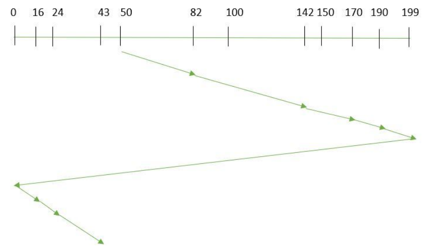
In the SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the **end of the disk**,it reverses its direction and again services the request arriving in its path.



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

4. CSCAN:

In CSCAN algorithm the disk arm instead of reversing its direction **goes to the other end of the disk** and starts servicing the requests from there till the initial starting point.



5. LOOK:

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

6. CLOOK:

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

RAID (Redundant Arrays of Independent Disks)

It is a way of storing the same data in different places on multiple hard disks for data redundancy and b etter performance.

Purpose of Data Redundancy:
In case of disk failure, if the same data is also backed up onto another disk, we can retrieve the data and go on with the operation.

Evaluation:

1. **Reliability:** How many disk faults can the system tolerate?
2. **Capacity:** Given a set of N disks each with B blocks, how much useful capacity is available to the user i.e. effective storage?

Blocks:Disk is divided into a predefined number of contiguously addressable disk blocks.

Different types

Raid-0	Instead of storing all the data in one HDD,different blocks are spanned across all the drives.This is called **striping**.Faster I/0 but 0 reliability.
Raid-1	Every block has at least two copies in different HDDs.This is called **mirroring**
Raid-4	One disk is dedicated to store parity (XOR).If one disk fails,we can retrieve data by looking at values of all other disks & the parity bit.
Raid-5	XOR values are distributed among all disks instead of storing in one separate disk.Faster read-write as compared to Radi-4 as it is random

	Raid-0	**Raid-1**	**Raid-4**	**Raid-5**
Reliability	0	1 to N/2	1	1
Capacity	100%	(N*B)/2	(N-1)*B	(N-1)*B

Chapter 7:File System

A file is a collection of logically related entities called records.

A file system is a process of managing how and where files are stored on a storage disk.
A directory is a container containing files.A directory contains all information about files, its attributes.

File Block: It is a physical record as a sequence of bytes or bits, usually containing some whole number of records, having a maximum length; a block size.

File allocation in OS:

Objective:

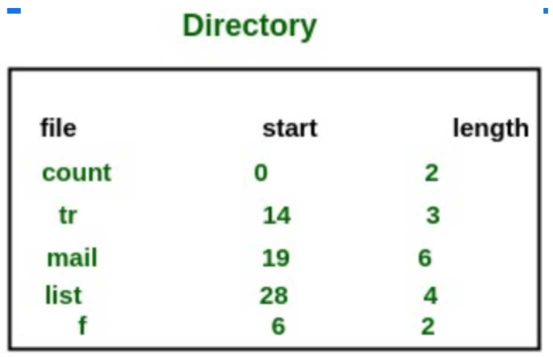
- Efficient disk space utilization.
- Fast access to the file blocks.

1. Contiguous Allocation

Each file occupies a contiguous set of blocks on the disk.

The directory entry for a file with contiguous allocation contains

- Address of starting block
- Length of the allocated portion.



Directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Advantages:
1.Easy to implement
2. Excellent read performance

Disadvantages:
1.Internal and external fragmentation
2.Difficult to store large sized file

2. Linked list allocation:

File blocks are distributed randomly on the disk and can only be accessed through pointers.
The directory entry contains a pointer to the starting and the ending file block.Each block contains a pointer to the next block occupied by the file.

Advantages:
1.No external fragmentation
2.Large size file can be stored

Disadvantages:
1.Very large seek time (as have to travel to different blocks for 1 file)
2.Random access to blocks not possible
3.Overhead of pointers.

3. Indexed Allocation

In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file

Advantages:
1.No external fragmentation
2.Random access to blocks possible

Disadvantages:
1.The pointer overhead for indexed allocation is greater than linked list allocation.
2.Even for a file with only 2-3 blocks,there will also be an entire index block resulting in wastage of memory.