Operating System

⇒ operating System :- An operating system is a program that manages the computer hardware.

- It also provides a basis for application programs and acts as intermediary between the computer user and the computer hardware.

⇒ Generations of operating System :-

a.) 1st Generation (1945-1955)
Vaccum Tubes and Plug boards.

b.) 2nd Generation (1955-1965)
Transistors and Batch Systems.

c.) The 3rd Generation (1965-1980)
Integrated Circuits and Multi programming.

d.) 4th Generation (1980- Current)
Personal computer.

* Types of operating System

1.) Simple Batch System :- In this, there is no

and the computer.

⇒ The user has to submit a job (written on cards or tape) to a computer operator.

* Advantages of Batch System

1.) No interaction between user and computer.

2.) No mechanism to priortise the processes.

2. Multi-processor system

A system consist of several processors that share a common physical memory. Multi-processor provides high computing power and speed. In multiprocessor system all processors operates under single operating system

* Advantages of Multiprocessor system

1.) Enhanced performance.

2.) ~~the~~ Execution of several task by different processors concurrently, increases the system's throughout speeding up the execution of a single task.

3. Desktop System

Earlier, CPUs and PCs lacked the features needed to protect an operating system from user programs. PC operating system therefore were neither multi-user nor

operating systems changed with time, instead of maximising CPU and phiripheral utilization the systems opt for maximising user convenience and responsiveness. These systems are called Desktop systems. for ex:- Apple Macintosh

4) Distributed Operating System

The motivation behind developing distributed operating system is the availability of powerful and inexpensive microprocessor. and advanced in communication technology. The main benefit of distributed system is its low price / performance ratio.

* Advantages Distributed operating System

1. Fast processing
2. less load on the Host machine.

* Types of distributed operating System:-

1. Client server System
2. Peer to Peer Systems

5.) Real time Operating System

The Real time operating system which gaurentees the maximum time for critical operations and complete them on

* Operating System Services

a.) User Interface :- ~~All all~~ almost all user interface (U.I). This interface can take several forms. One is a command-line interface (CLI), which uses text commands and a method for entering them.

b.) Program execution :- The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally (indicating error).

c.) I/O operations :- A running program may require I/O, which may involve a file or an I/O device. For specific devices, special functions may be desired (such as recording to CD or DVD drive or blanking a display screen). for efficiency & protection, user usually cannot control I/O devices directly. Therefore, the OS must provides a means to do I/O.

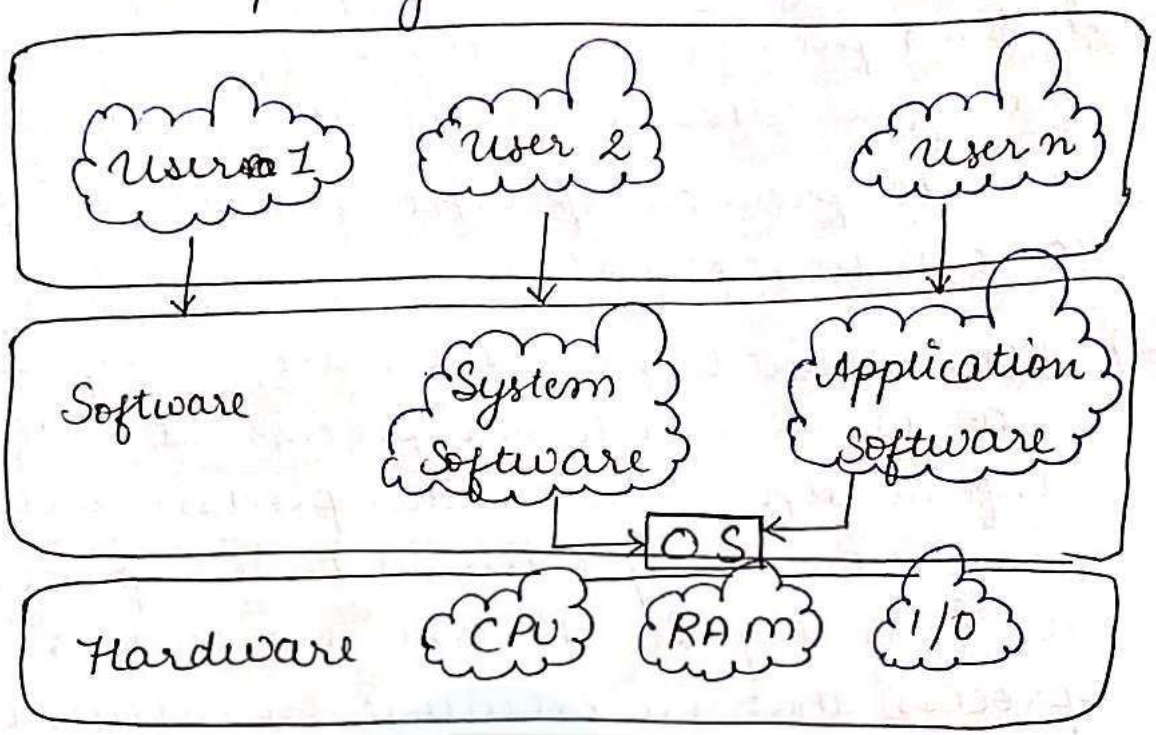
d.) file-System manipulation :- The file system of a particular interest, obviously need to read and write files and directories they also need to create and delete them by name, search for a given file, and list

systems include permissions management to allow or deny access to files or directories based on ~~life~~ file ownership. Many operating systems provide a variety of file systems, sometimes to allow personal choices and sometimes to provide specific features or performance characteristics.

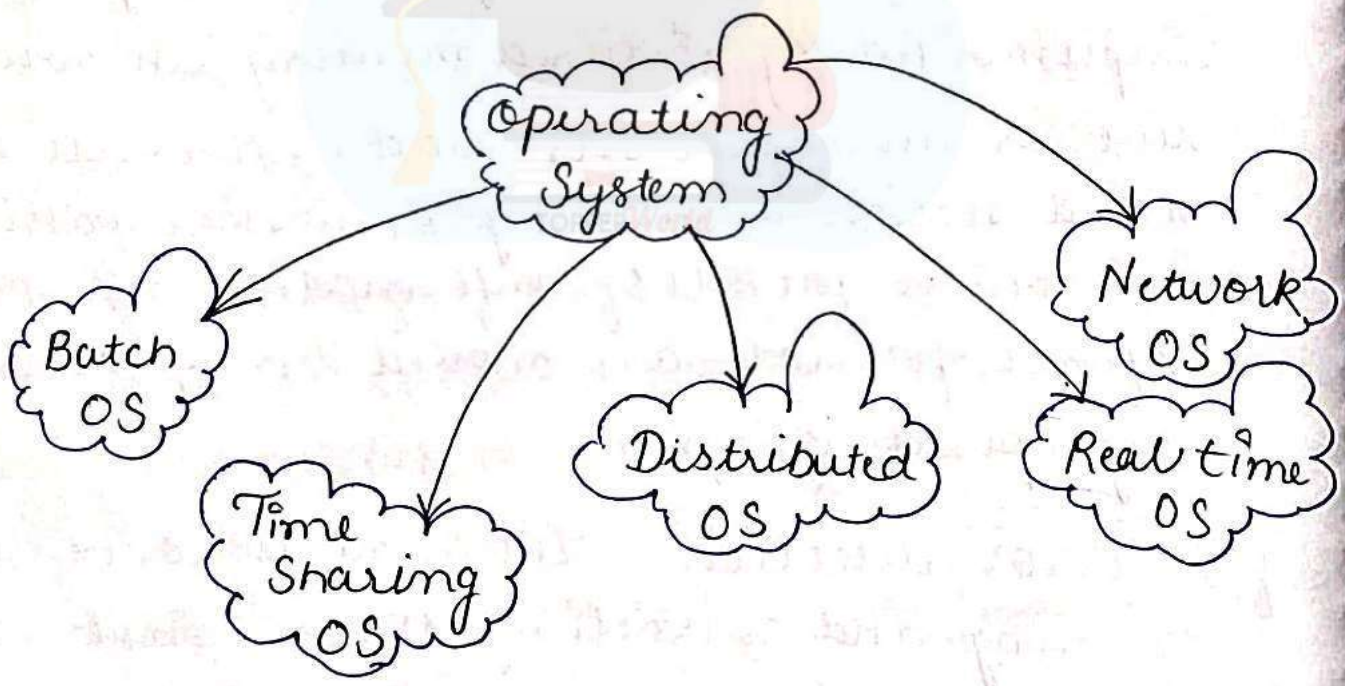
e.) Communications :- There are many circumstances in which one process need to exchange information with another process. Such communication may occur between processes that are executing on the same computer or between process that are executing on different computer systems tied together by a network. Communication may be ~~shared by a network~~ implemented by a "shared memory" in which two or more processes read and write to a shared section of memory, or message passing, in which packets of information in predefined formats are moved b/w processes by operating system.

f.) Error detection :- The OS needs to be detecting and correcting error constantly. Error may occur in CPU & memory hardware (such as a memory error or a power failure), in I/O devices (such as

· for each type of error, the OS should take the appropriate action to insure correct and consistent computing.



* Types of operating System :-



* Batch OS :- The user of this OS does not interact with the computer directly. Each user prepares his job with similar on an offline device called punch

→ lack of Batch OS :-

1. lack of interaction b/w the user & job.
2. CPU is often idle, because the speed of I/O mechanical device is slower than CPU.
3. Difficult to provide desired priority.

2.) Time sharing OS.

It is a technique which enables many user, located at various terminals, to use a particular computer system at the same time. Processor's time is simultaneously (among shared among multi-users is termed as time sharing.

Advantages :-

- Quick response
- Avoid duplication of software
- Reduce CPU idle time.

Disadvantages :-

- Problem of Reliability
- Problem of data communication.

3.) Distributed operating System

It use multiple central processors to serve multiple real time applications and multiple user. Data processing jobs are distributed among processors accord-

Advantages :-

1. Better service to the customer.
2. Reduction of the load on the host computer.
3. Reduction of delays in data processing.
4. Speedup the exchange of data with one another via electronic mail.

4.) Network OS

A Network OS runs on a server and provide the server capability to manage data, user, groups, security, applications, and other networking function

Advantages :-

1. Highly stable centralized server.
2. Security concerns are handled through server
3. New technologies and hardware upgradation are easily integrated to the system.
4. Server access are possible remotely from different location and types of systems.

Disadvantages :-

1. Servers are costly.
2. Maintainence and updates are required regularly

* <u>Processes</u> :- A program in execution is called process.

* States of process :- The state of process is defined in part by the current activity of that process.

New :- The process is being created.

↓

Running :- Instructions are being executed.

↓

waiting :- The process is waiting for some event to occur (such as an I/O completion or receptions of signal).

↓

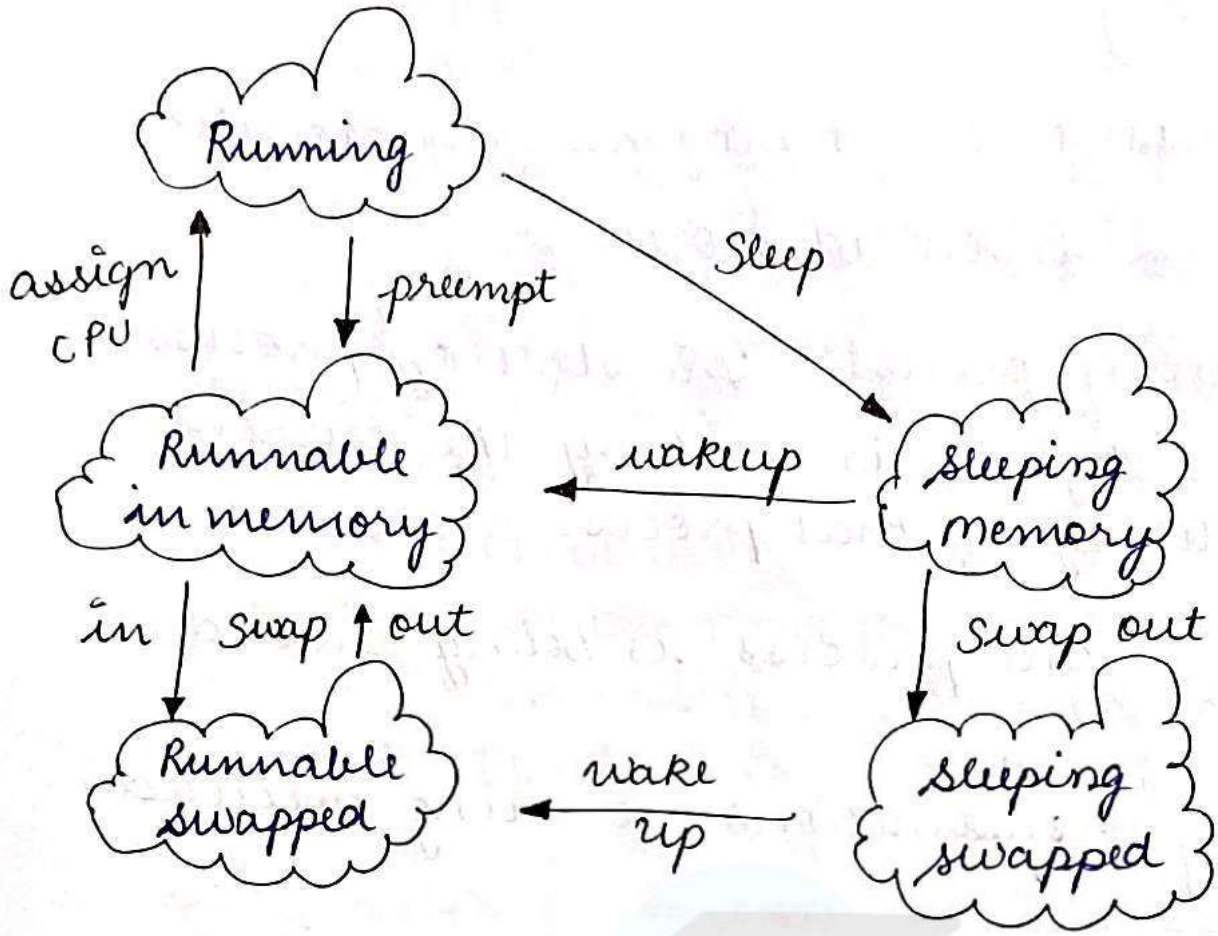
Ready :- The process is waiting to be assigned to a processor.

↓

Terminated :- The process has finished execution.

* Process State Transition :-

applications that have strict real-time constraints might need to prevent process from being swapped or paged out to secondary memory. A simplified overview



Process State Transition Diagram

* Process Control Block

Each process is represented in the operating system by a process control block (PCB) — also called a Task Control Block. It contains many pieces of information associated with a specific process, including these :-

1. Process state
2. Program counter
3. CPU Registers
4. CPU-Scheduling information
5. Memory Management information

Process State
Process Number
Program Counter
Registers
Memory limits
list of open files
. . .

Process control Block

* Context Switch :- Switching CPU to another process Requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch. When a context switch occurs, the kernal saves the context of the old process in its PCB. and load the saved (~~process~~) context of the next process schedule to run. Context-switch time is pure overhead, because the system does not useful work while switching. Switching speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the exist-ence of special instruction (such as single instruction to load or store all

* Threads in OS

Threads :- A thread is a path of execution within a process. A process can contain multiple threads. It is also known as light weight process.

Multi-threading :- The idea is to achieve parallelism by dividing a process into multiple threads. for example :- In a browser, multiple tabs can be different threads. ~~for Example~~ MS word uses multi-ple threads: one thread to format the text, another thread to process inputs, etc.

* Types of threads :-

1. User level thread.
2. Kernal level thread.

User level	Kernal level.
1.] It is implemented by user.	1.] It is implement by OS.
2.] OS doesn't recognize user level threads.	2.] Kernal threads are recognized by OS.
3.] Implementation of user threads is easy.	3.] Implementation of kernal thread is complicated.
4.] Context switch time	4.] Context switching

5.] context switch require no hardware support	5.] Hardware support is needed.
6.] Example :- Java thread, POSIX threads.	6.] Example :- window Solaris.
7.] If one user level thread perform blocking operat-ion then entire process will be blocked	7.] If one kernal thre-ad perform blocking operation then another thread can continue Execution

* Benefits / Advantages of threads

1. threads minimize the context switching time
2. Use of threads provide concurrency with a process.
3. efficient communication.
4. It is more economical to create and context switch threads.
5. Threads allow utilization of multiprocessor architectures to a greater scale and effici-ency.

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

* Main objectives of Scheduling :-

1.) Make the system fast.

4.) Maximize the number of users recieving acceptable responses time.

* Scheduler :- The main task is to select the jobs to be submitted into the system and to decide which process to run.

It has three types :-

1.) long term Scheduler
2.) Short-term Scheduler
3.) Medium term scheduler

long term	short term	medium term
1.) It is job schedular	1.) It is CPU Schedular	1) It is process swapping
2.) Speed is lesser than short term scheduler	2.) speed is fastest among other two	2.) speed is in b/w short and long term schedular.
3.) It controls the degree of multiprogramming	3.) It provides lesser control over degree of multiprogramming	3.) It reduces the degree of multi-programming
4.) It is almost absent or minimal in time sharing system.	4.) It is also minimal in time sharing system.	4.) It is a part of Time sharing System.
5.) It selects processes from pool and loads them into memory for execution	5.) It select those processes which are ready to execute.	5.) It can re-introduce the process into memory and

Process Scheduler :- A process scheduler different processes to be assigned to the CPU based on particular scheduling algorithms.

Scheduling Algorithms

- First come, First-Served (FCFS)
- Shortest-JOB-Next (SJN) Scheduling
- Priority Scheduling
- Short Remaining Time
- Round Robin (RR) Scheduling
- Multi-level Queues Scheduling

These algorithms are either preemptive or Non-preemptive.

- It is the responsibility of the CPU scheduler to allot a process to CPU whenever the CPU is in the idle state. The CPU scheduler selects a process from ready queue and allocates the process to CPU.

The scheduling which takes place when a process switches from running state to ready state or from waiting state to ready state is called Preemptive Scheduling

The scheduling which takes place when a process terminates or switches from running to waiting for state this kind of CPU scheduling is called Non-Preemptive Scheduling.

- In preemptive the process can be scheduled
- In Non-preemptive the process can not be scheduled.

* <u>Scheduling criteria</u>:-

1.) CPU utilization :- Keep the CPU as busy as possible (from 0% to 100%)

2.) Through put :- # of processes that complete their execution per time unit.

3.) Turn around time :- Amount of time to execute a particular process.

4.) Waiting time :- Amount of time a process has been waiting in the ready queue.

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

6.) Burst time :- Time required by a process for CPU execution.

7.) Arrival time :- Time at which the process arrives in the ready queue.

- Turnaround Time = Completion - Arrival

Unit :- 2

Inter Process Communication

* What is Inter Process Communication (IPC)?
It is a process that involves communication of one process with another process.

Communication can be of two types :-

1.) Between related processes initiating from only one process, such as parent and child process.

2.) Between unrelated processes, or two or more different processes.

Some Important terms :-

- Pipes :- communication b/w two related processes. The mechanism is half duplex meaning the first process communication with the second passes. To achieve full duplex i.e for the second process to communicate with the first process another process is required.

- FIFO :- Communication b/w two unrelated processes. FIFO is a full duplex, meaning the first process can communicate with the second process and vice verse at the same time.

- Message Queues :- Communication b/w two or more processes with full duplex capacity. The processes will communicate with each other by posting a message and retriving it out of the queue. Once, Retrieved, the message is no longer available in

- Shared memory:- communication b/w two or more processes is achieved through a shared piece of memory among all process. The shared memory needs to be protected from each other by synchoronizing access to all the process.
- Semaphores:- Semaphores are meant for synchronizing access to multiple processes. When one process wants to access the memory (for reading & writing), it needs to be locked (or protected) and released when the access is removed. This needs to be repeated by all the processes to secure data.
- Signals:- It is a mechanism to communication b/w multiple process by way of signaling.

* <u>Critical section</u> :- consider a system consisting of n processes $\{P_0, P_1 ---, P_{n-1}\}$. Each process has a segment of code, called a critical section. in which the process may be changing common variables, updating a table, writing a file, and so on. The important feature of the system is that, when one process is executing in its critical section, no other process is allowed to execute in its critical section.

↳ The critical section problem is to design a protocol that the processes can use to cooperate. Each

critical section. The section of the code implementing this request is the entry section.

- The critical section may be followed by an exit section.
- The remaining code is the Remainder section.

▶ A solution to the critical-section problem must satisfy the following three requirements :-

1.) Mutual exclusion:- If one process is executing in its critical section, then no other process can be executing in their critical sections.

2.) Progress:- If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.

3.) Bounded-waiting:- there exists a limit or bound on the no. of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical sections and before that request is granted.

- Two general approaches are used to handle critical sections in OS:-
  1.) preemptive kernals
  2.) Non-preemptive kernals.

* <u>Race Condition</u> :- It is a situation that may occur inside a critical section. This happens when the result of multiple thread execution in critical section ~~different from~~ differs according to the order in which the thread execute.

↳ Race condition in critical section can be avoided if the critical section is treated as an atomic instruction.

Also, proper thread synchronization using locks or atomic variables can ~~be~~ prevent race conditions.

* <u>The Producer-Consumer Problem</u> :- It is a classical problem which is used for multi-process ~~synchr~~ synchronization between more than one processes.

In the producer-consumer problem, there is one producer that is producing something and there is one consumer that is consuming the products.

- The producer and consumers share the same memory buffer that is of fixed size.
- The job of the producer is to generate the data, put it into the buffer, and again start generating data. while the job of the consumer is to consume the data from the buffer.

* Problems :- The ~~procedure~~ producer should produce data only when the buffer is not full. If the buffer is full, then the producer shouldn't be allowed to put any data into the buffer.

- The consumer should consume data only when the buffer is not empty. If the buffer is empty, then the consumer shouldn't be allowed to take any data from the buffer.
- The ~~procedure~~ producer and consumer should not access the buffer at the same time.

* Solutions :- the above three problems can be solved with the help of Semaphores.

In the producer - consumer problem, we use three semaphores variables :-

1) Semaphore S.
2) Semaphore E.
3) Semaphore F.

By using the above three semaphores variables and using the wait( ) and signal( ) function, we can solve our ~~prof~~ problem (the wait( ) function decreases the semaphore variable by 1 and the signal( ) function increases the semaphores variable by 1). So. let's see how.

1.) Semaphore S :- This semaphore variable is used to achieve mutual exclusion between processes. By using this variable, either producer or consumer will be allowed to use or accessed the shared buffer at a particular time. This variable is set to 1 initially.

2.) Semaphore E :- This semaphore variable is used to define the empty space in the buffer. Initially, it is set to the whole space of the buffer i.e. "n" because the buffer is initially empty.

3.) Semaphore F :- This semaphore variable is used to define the space that is filled by the producer. Initially, it is set to '0' because there is no space filled by the producer initially.

The following is the pseudo-code for the producer :-

```
void producer() {
    while (T) {
```

```
        wait(S)
        append ( )
        signal (S)
        signal (F)
    }
}
```

The above code can be summarised as:-

- while ( ) is used to produce data, again and again, if it wishes to produce, again and again.
- produce( ) function is called to produce data by the producer.
- wait(E) will reduce the value of semaphore variable "E" by one i.e when the producer produces something then there is a decrease in the value of the empty space in the buffer. If the buffer is full i.e the value of the semaphore variable "E" is "0", then the program will stop its execution and production will be done.
- wait(S) is used to set the variables "S" to "0" so that no other process can enter into the critical section.
- Append( ) is used to append the newly produced data in the buffer.
- signal(S) is used to set the semaphore variable "S" to "0" so that other processes can come

production is done and the append operation is also done.

- Signal(F) is used to increase the semaphore variable "F" by one because after adding the data into the buffer, one space is filled in the buffer and the variable "F" must be updated

↳ This is how we solve the produce part of the produ-er/consumer-problem. Now, let's see the consumer solution. the following is the code for the con-sumer:

```
void consumer () {
while(T) {
    wait (F)
    wait (S)
    take ( )
    signal (S)
    signal (E)
    use ( )
  }
}
```

The above code can be summarised as:-

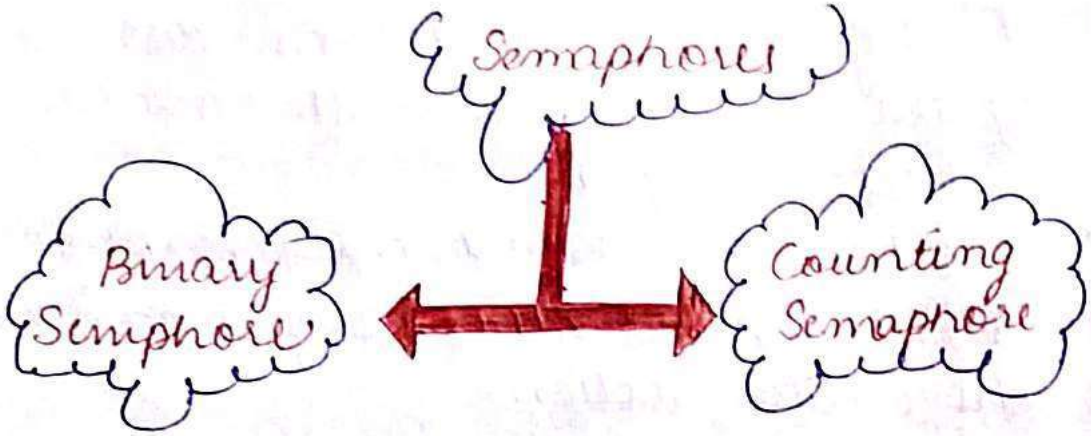
- while ( ) is used to consume data, again & again, if it wishes to consume, again and again.

variable "F" by one because if some data is consumed by the consumer then the variable "F" must be decreased by one.

- wait(S) is used to set the semaphore variable "S" to "0" so that no other process can enter into the critical section.
- take() function is used to take data from the buffer by the consumer.
- Signal(S) is used to set the semaphore variable "S" to "1" so that other processes can come into the critical section now because the consumption is done and the operation is also done.
- signal(E) is used to increase the semaphore variable "E" by one because after taking the data from the buffer, one space is freed from the buffer and the variable "E" must be increased.
- use() is a function that is used to use the data taken from the buffer by the process to do some application.

* Semaphore

Semaphore is simply a variable which is non-negative and shared between threads. This variable is used to solve the critical section problem and to achieve process synchronization



1.) Binary Semaphore :- It is also known as Mutex lock. It can have only two values - 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problem with multi processes.

2.) Counting Semaphore :- Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

↳ characteristics of Semaphores :-

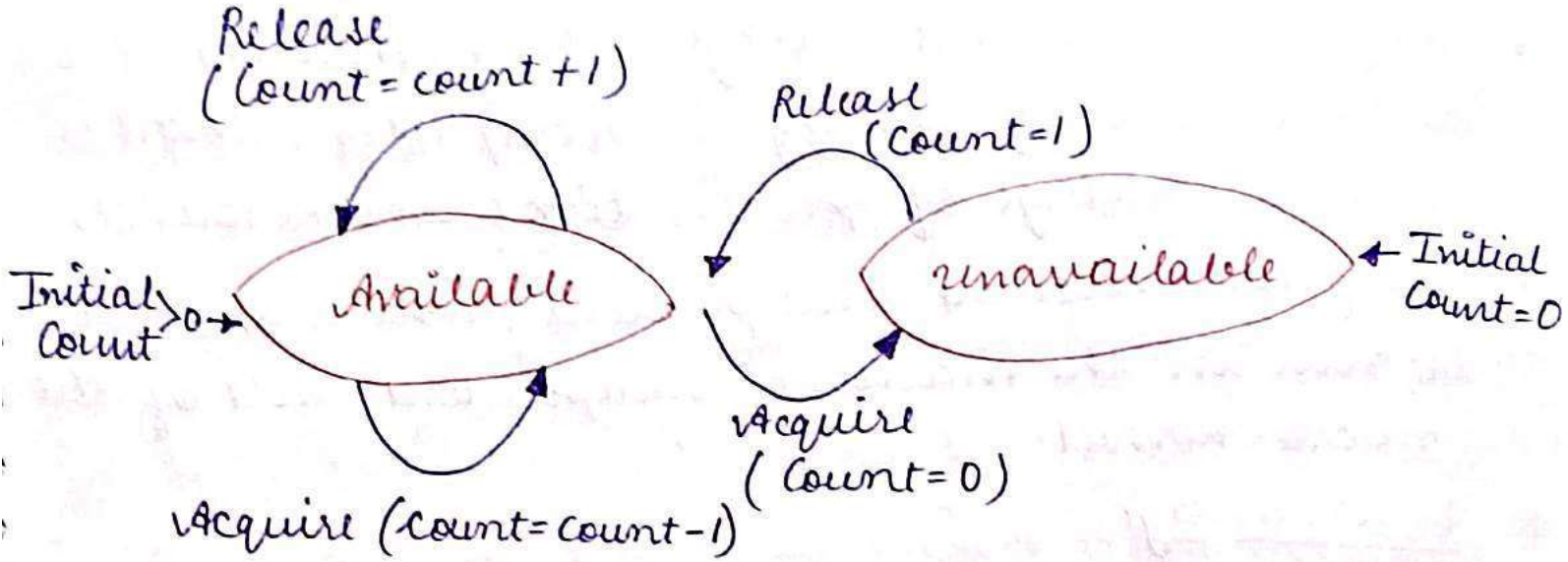
1.) It is a mechanism that can be used to provide synchronization of tasks.

2.) It is low-level synchronization mechanism

3.) Semaphore will always hold a non-negat-ive integer value.

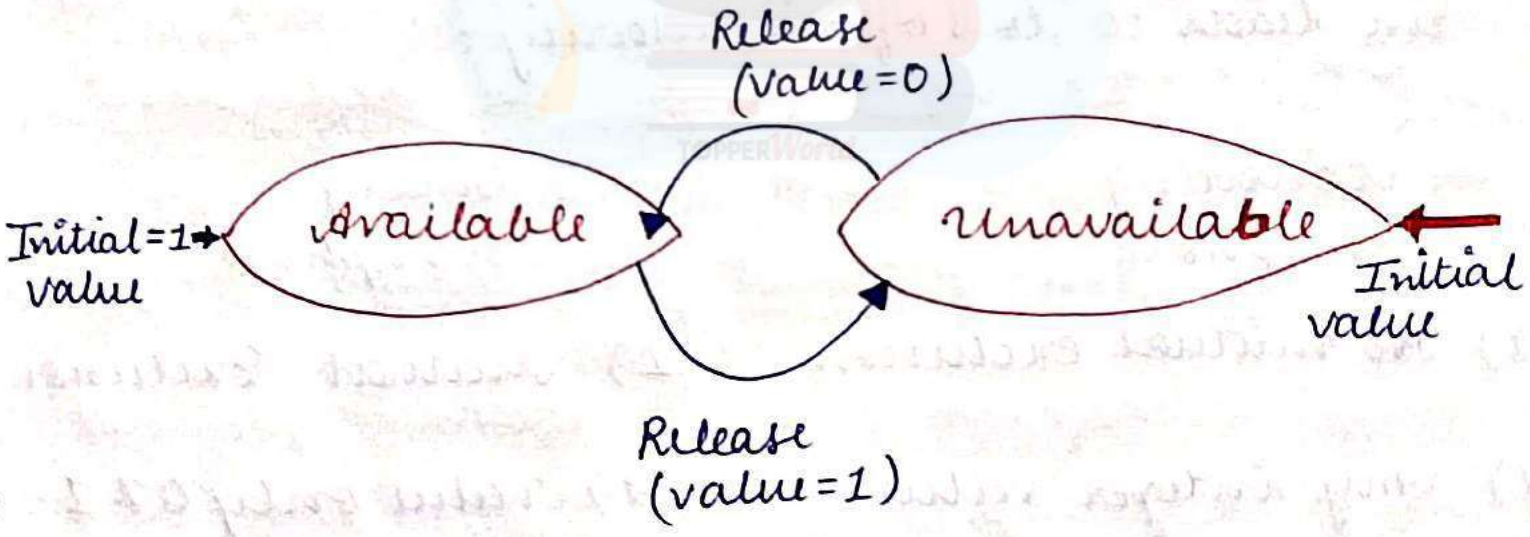
4.) Semaphore can be implemented using test operations and interrupts, which should be executed using file descriptors.

Counting Semaphore



However, if the count is > 0, the semaphore is created in the available state, and the number of tokens it has equal to its count.

Binary Semaphore



* Advantages of Semaphores

- It allows more than one thread to access the critical section.

- They do not allow multiple-processes to enter the critical section.
- They allow flexible management of resources.
- As there is busy waiting in semaphores, there is never a wastage of process time and Resources.
- They are machine-independent, which should be run in the machine-independent code of the micro-kernal.

* <u>Disadvantages</u>

- One of the biggest limitation of the semaphore is priority inversion.
- The OS has to keep track of all calls to wait & signal semaphores.
- Semaphore is prone to programmer error.
- It is also not practical for large scale use as their use leads to loss of modularity.

Counting Semaphore	Binary Semaphore
1.) No mutual exclusion	1.) Mutual Exclusion
2.) Any integer value	2.) Value only 0 & 1.
3.) More than one slot	3.) Only one slot
4.) Provide a set of Processes	4.) It has a mutual exclusion mechanism

Monitor in Process Synchronization

- The monitor is one of the ways to achieve process synchronization. The monitor is supported by programming languages to achieve mutual exclusion between processes.

  for ex:- JAVA synchronized methods. JAVA provide wait( ) and notify ( ) construct.

1.) It is a conditional variables and procedures combined together in a special kind of module or a package.

2.) The processes running outside the monitor can't access the internal variable of the monitor but can procedure of the ~~no mirror~~ monitor

3.) Only one process at a time can execute code inside monitors.

Syntax :

```
Monitor Demo //Name of Monitor
{
variables;
conditional variable;
procedure p1 {....}
procedure p2 {....}
}
```

Syntax of Monitor

* Advantage of Monitor:-

1.) Making parallel programming easier and less error prone than using techniques such as semaphores.

* Disadvantage of Monitor :-

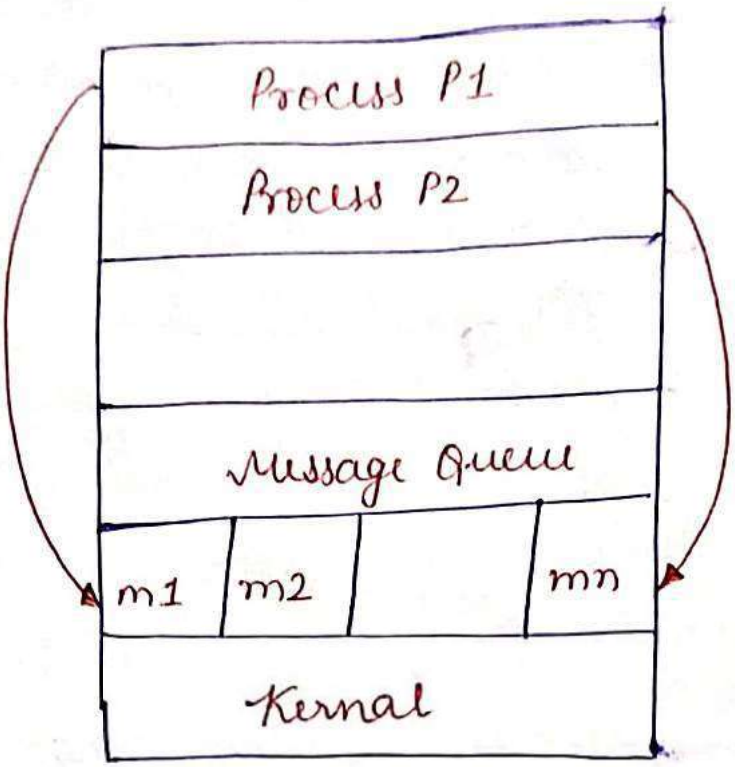
- Monitor have to be implement as part of the programming language. The compiler must generate code of them. This gives the complier the additional burden of having to know what operating system facilities are available to control access to critical sections in concurrent processes. Some languages that do support monitors are JAVA, C#, Visual Basic, Ada and concurrent euclid.

* Message Passing Model of Process Communication

Process mechanism is a mechanism provided by the operating system that allows processes to communicate with each other.

Message passing model allows multiple processes to read and write data to the message queue without being connected to each other. Message are stored on the queue until their ~~resp~~ recipient retrieves

communication used by most operating system. and are



Message Passing Model

* Advantage of Message Passing Model

- It is much easier to implement than the shared model.
- It is easier to build parallel hardware using message passing model as it is quite tolerant of higher communication latencies (the delay before a transfer of data begins)

* Disadvantage of Message Passing model.

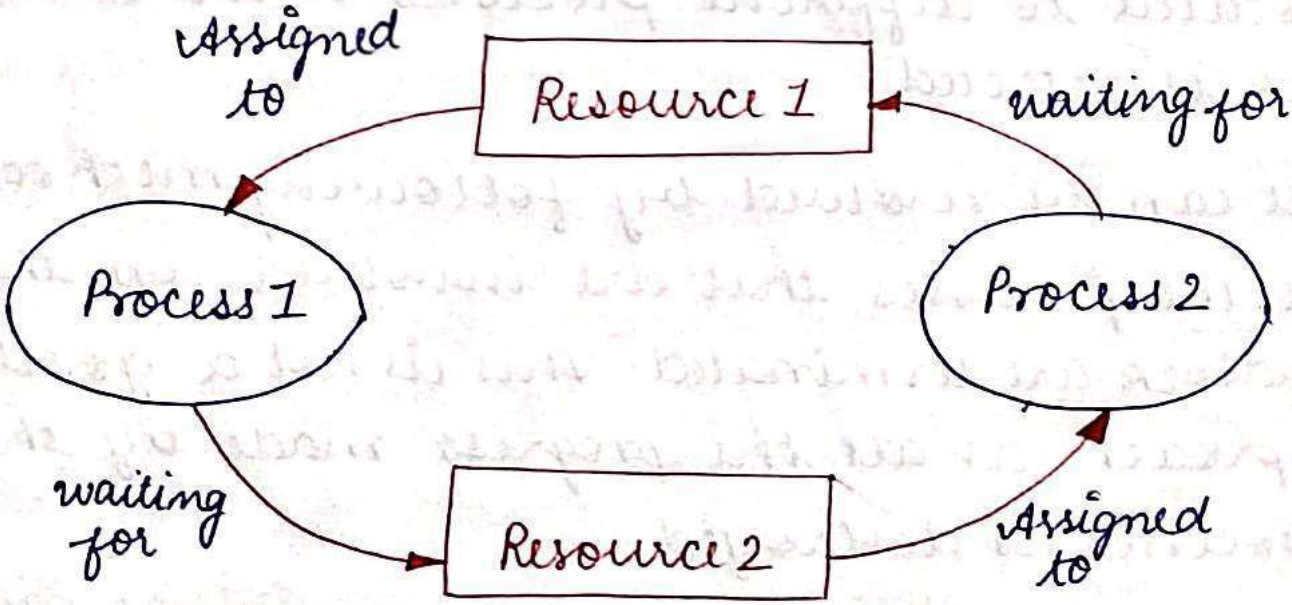
- It has slower communication than the shared memory model Because the connection setup takes time.

Deadlock

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

Necessary condition for Deadlock

- Mutual Exclusion :- One or more than one resource are non-starable (only one process can use at a time)
- Hold and wait :- A process is holding atleast one resource and waiting for resources.
- No-preemption :- A resource can be taken from a process unless the process releases the resources.
- Circular wait :- A set of processes are waiting for each other in circular form.



* Deadlock Prevention

It is very important to prevent deadlock before it can occur So, the system check each transaction before it is executed to make sure it does not lead to ~~deadlock~~ deadlock. If there is even a slight chance that a transaction may lead to dead-lock in the future, it is never allowed to executed.

* Deadlock Avoidance

It is better avoid a deadlock rather than takes measures after the deadlock has occured. The wait for graph can be used for deadlock avoidance.

* Deadlock detection

A deadlock can be detected by a resource sched-ular as it tracks all the resources that are allocated to different processes. After a dead-lock is detected,

↳ It can be resolved by following methods:-

- All the processes that are involved in the deadlock are terminated. This is not a good approach as all the progress made by the processes is destroyed
- Resources can be preempted from some proce-ses and given to others till the deadlock

* Banker's Algorithm

The Banker's Algorithm is a Resource allocation and deadlock avoidance algorithm that tests for safety by stimulating the allocation for predetermined maximum possible amounts of all resources, then makes an "S-State" check to test for possible activities, before deciding whether the allocation should allowed to continue.

Unit - 3

Memory Management

Main memory refers to a physical memory that is the internal memory to the computer. The word main is used to distinguish it from external mass storage devices such as disk drives Main memory is also known as RAM. The computer is able to change only data that is in main memory Therefore every program we execute and every file we access must be copied from a storage device into main memory.

All the programs are loaded in the main memory for execution. Sometimes complete program is loaded into the memory, but sometimes a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called Dynamic loading, this enhance the performance

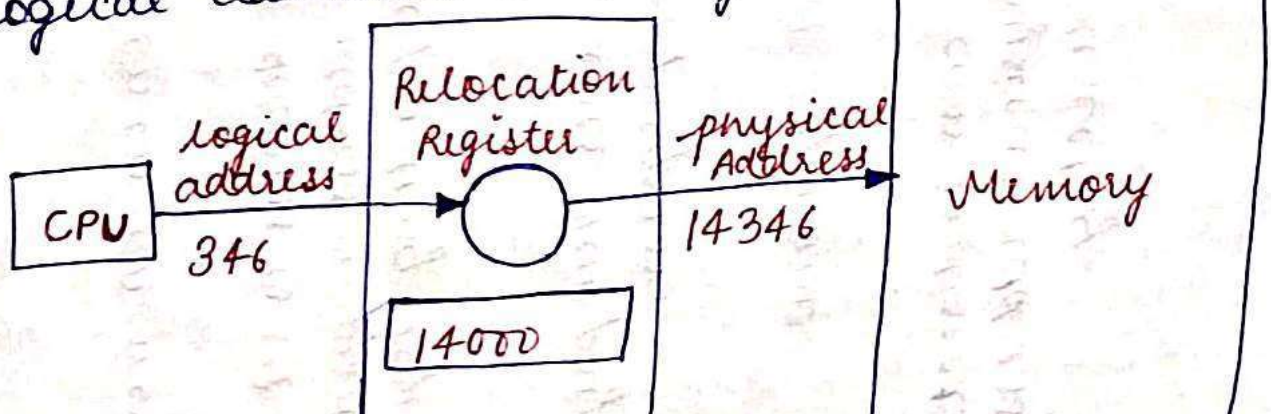
* Logical and Physical Address

Logical Address is generated by CPU while a programming is running. The logical address is virtual address as it does not exist physically, therefore, it is also known

as a reference to access the physical memory location by CPU. The term logical Address space is used for the set of all logical address generated by a program's perspective.

The hardware device called Memory-Management unit is used for mapping logical address to its corresponding physical address.

Physical Address :- It identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a logical address space.



* Difference between logical Address and Physical Address.

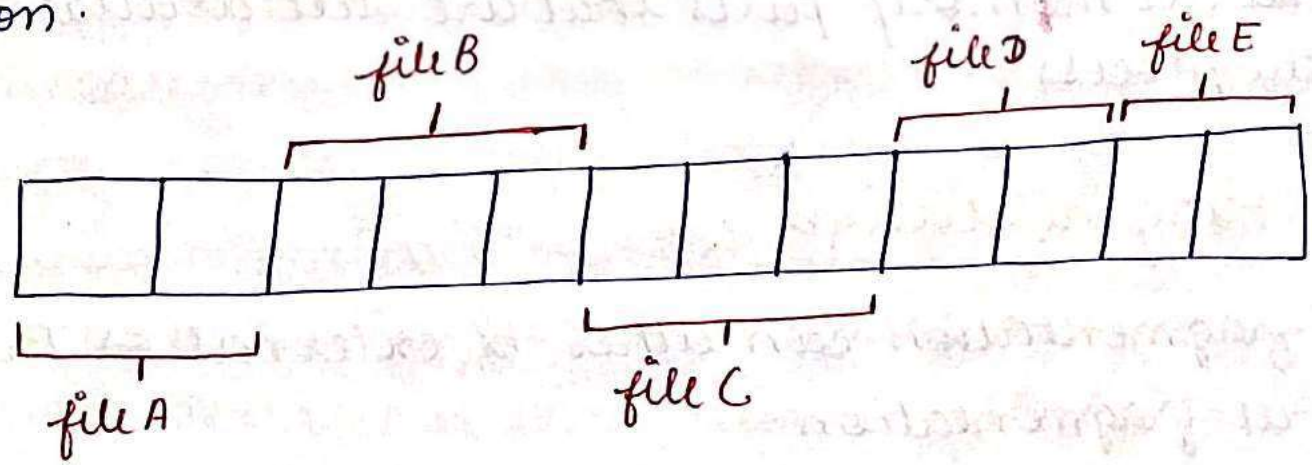
Parameter	logical Address	Physical Addre
BASIC	Generated by CPU	location in a memory
Address Space	logical Address space is set of all logical addresses generated by CPU in reference to a program.	Physical Address is set of physical address mapped the corresponding logic address.
Visibility	user can view the logical address of a program	user can never view phys address of program.
Generation	Generated by the CPU	computed by MMU.
Access	The user can use the logical address to access the phys-ical address.	The user can indirec access physical addre but not directly.

* Memory Allocation

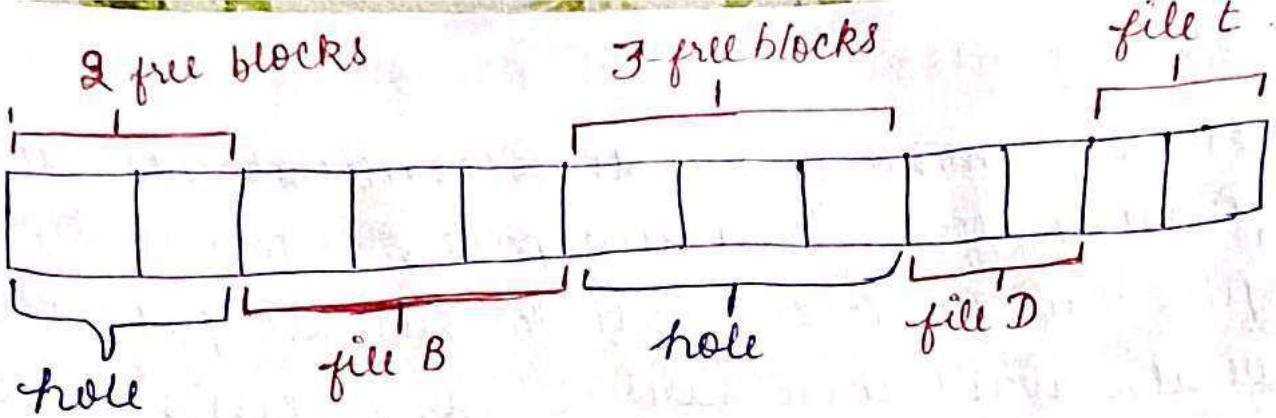
The main memory has to accomodate both the operating system and userspace. Now, here the space has to accomodate various user processes. We also want these several user processes must reside in the main memory at the same time.

Now, the question arises how to allocate the available memory space to the user processes that are waiting in a ready queue?

In contiguous memory allocation, when the process arrives from the ready queue to the main ~~memo~~ memory for execution, the contiguous memory blocks are allocated to the process according to its requirement. Now, to allocate the contiguous space to user processes, the memory can be divided either in the fixed-sized parition or in the variable sized partition.



a.) Contiguous memory Allocation of 5-files.



(b.) when the file A and C terminates and release the memory creating hole.

* <u>fixed sized partition</u> :- In this, the memory is divided into fixed-sized blocks and each blocks contains exactly one process. But, the fixed-sized partition will decide the number of processes.

* <u>Variable-Size partition</u>:- In the variable size parition method, the operating system maintains a table that contains the information about all memory parts that are occupied by the processes and all memory parts that are still available for the process.

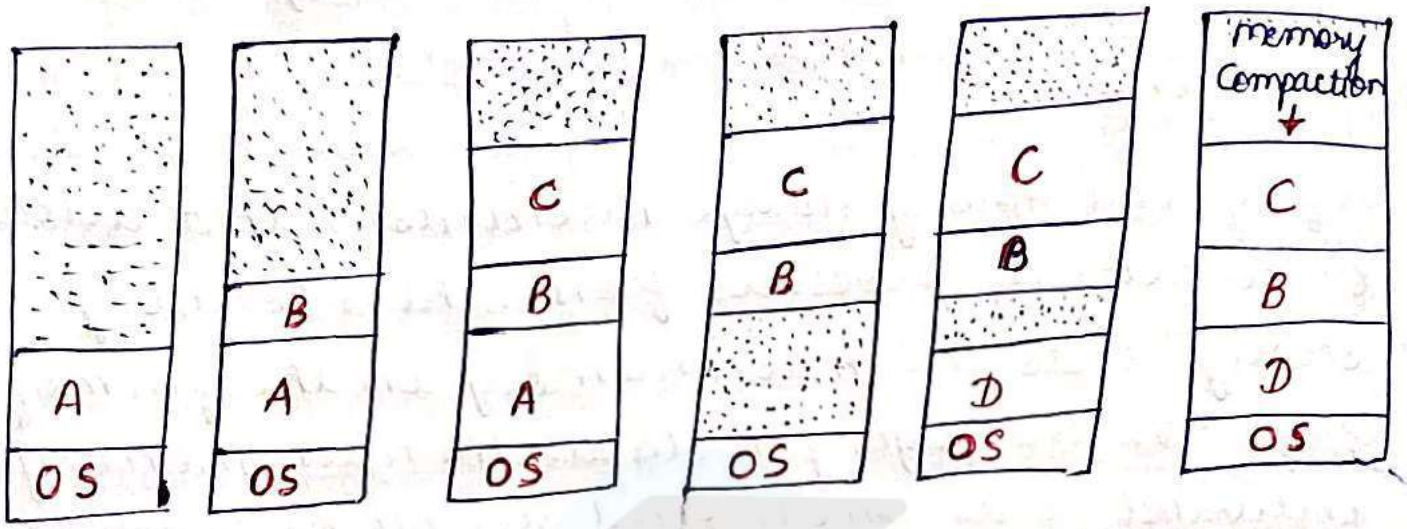
* <u>Fragmentation</u>

fragmentation can either be external or Internal fragmentation.

↳ External fragmentation :- when the free memory blocks available in memory are too small and even non-contiguous.

4. Internal fragmentation :- It occurs when the process does not fully utilize the memory allocated to it.

The solution to the problem of external fragmentation is called Memory compaction.



Memory compaction

* Advantages and Disadvantages of memory compaction.

The main disadvantage of contiguous memory allocation is memory waste and inflexibility. As the memory is allocated to a file or a process keeping in mind that it will grow during the run. But until a process or a file grows many blocks allocated to it remains unutilized. And they even they cannot be allocated to the other process leading to wastage of memory.

In case the process or the file grows beyond the

leading to inflexibility.

The advantage of contiguous memory allocation is it increases the processing speed. As the operating system uses the buffered I/O and reads the process memory blocks consecutively it reduces the head movements. This speed ups the processing.

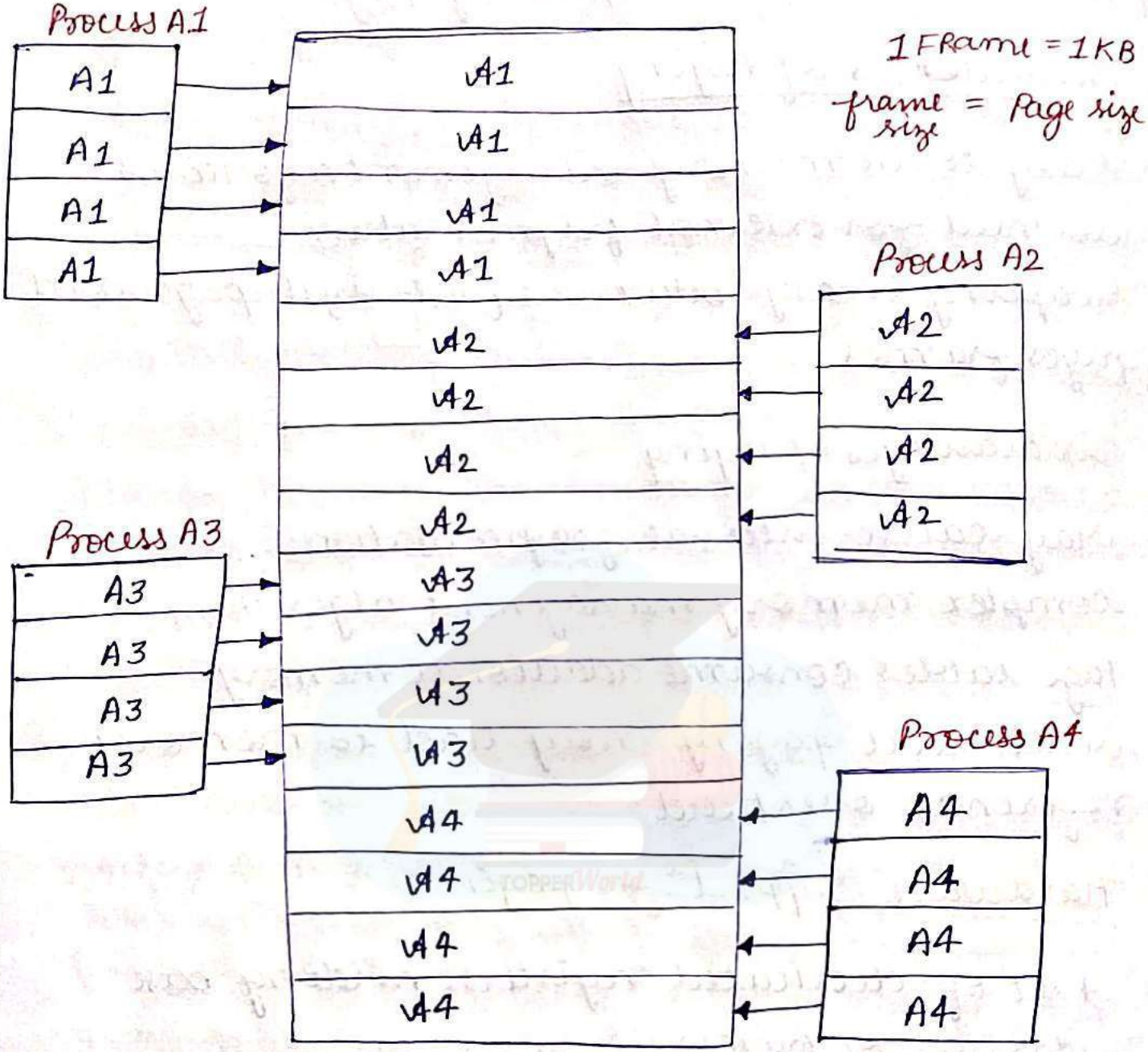
* Paging

Paging is a ~~storing~~ storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages. In the paging method, the main memory is divided into small fixed size blocks of physical memory, which is called frames. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept.

Example:- if the main memory size is 16 KB and frame size is 1 KB. Here, the main memory will be divided into collection of 16 frames of 1 KB each.

There are 4 separate processes in the system that is A1, A2, A3 and A4 of 4 KB each. Here, all the processes are divided into pages of

At the beginning of the process, all the frames remains empty so that all the pages of the processes will get stored in a continguous way.



* Paging Protection

The paging process should be protected by using the concept of insertion of an additional bit called Valid / Invalid bit. Paging memory

protection bits with each page. These bits are associated with each page table entry and specify protection on the corresponding page.

Advantages of Paging

- Easy to use memory management algorithms.
- No-need for external fragmentation.
- Swapping is easy between equal-sized pages and pages frames.

Disadvantages of Paging

- May cause internal fragmentation.
- Complex memory management algorithm.
- Page tables consume additional memory.
- Multi-level paging may lead to memory reference overhead.

Hardware Support for paging:-

- A set of dedicated registers, holding base addresses of frames.
- In memory page table with a page table base register (PTBR).

* Virtual Memory

It is a space where large programs can store themselves in form of pages while their execution and only the required pages or portions of processes are loaded into the main memory. This technique is very useful as large virtual memory is provided for user programs when a very small physical memory is there.

↳ Benefits of having Virtual memory

1.) large programs can be written, as virtual space available is huge compared to physical memory.

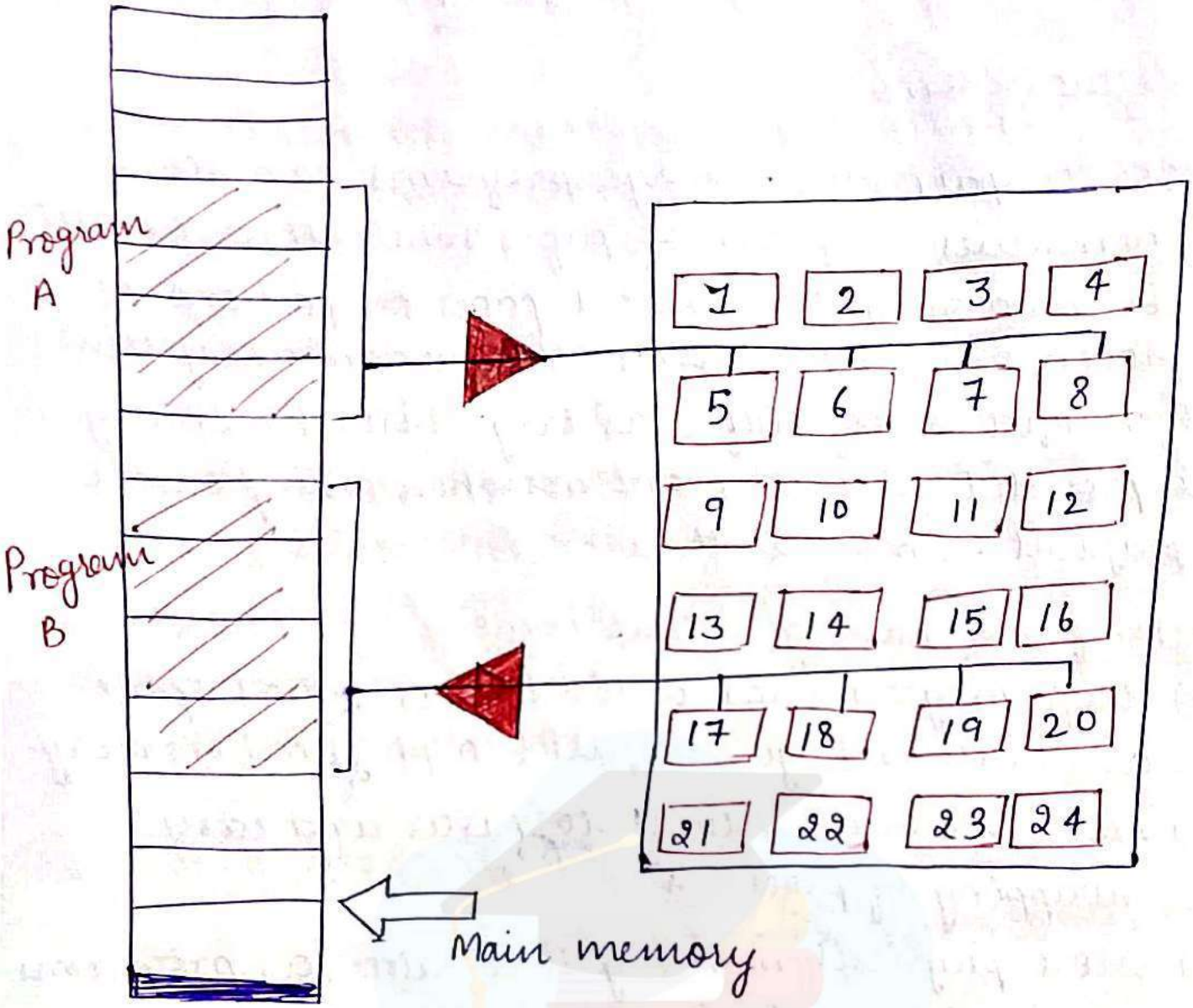
2.) less I/O required, leads to faster and easy swapping of processes.

3.) More physical memory available, as programs are stored on virtual memory, so they occupy very less space on actual physical memory.

↳ Demand Paging

The Basic idea behind demand paging is that when a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them (on demand). This is termed as lazy swapper, although a pager is a more accurate term.

Initially only those pages are loaded which will



The page that are not moved into memory are marked as invalid in the page table. for an an invalid entry the rest of the table is empty. In case of pages that are loaded in the memory, they are marked as valid along with the information about where to find the swapped outpage.

When the process requires any of the page that is not loaded into the memory, a ~~page~~ page fault trap is trigerred and following

1.) the memory address which is requested by the process is first checked, to verify the request made by the process.

2.) If it is found to be ~~valid~~ invalid, the process is terminated.

3.) In case the request by the process is valid, a free frame is located, possibly from a free frame list, where the required page will be moved.

4.) A new operation is scheduled to move the necessary page from disk to the specified memory location

5.) when the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to valid.

6.) The instruction that caused the page fault must now be restarted from the Beginning.

These are cases when no pages are loaded into the memory initially, pages are only loaded when demanded by the process by generating page faults. this is called Pure Demand Paging.

↳ The only major issue with demand Paging is, after a new page is loaded, the process starts execution from the Beginning. It is not a Big issue for small programs, but for larger programs it affects performance drastically.

↳ PAGE Replacement

This allow us to get more no. of processes into the memory at the same time. But what happens when a [illegible] pages and no

following steps can be taken to deal with this problem:-

1.) Put the process in the wait queue, until any other process finishes its execution thereby freeing frames.

2.) Or, remove some other processes completely from the memory to free frames.

3.) Or, find some pages that are being used right now, move them to the disk to get free frames. This technique is called Page Replacement and is most commonly used. We have some great algorithms to carry on Page replacement efficiently.

FIFO Page Replacement

- A very simple way of page replacement is FIFO.
- As new page are requested and are swapped in, they are added to tail of a queue and the page which is at the head becomes the victim.
- It is an effective way of page replacement but can be used for small systems.

* Optimal Page Replacement

The algorithm that has the lowest page-fault rate of all algorithms and will never suffer from Belady's Anomaly. Such algorithm does exist and has been

It is simply this :-

Replace the page that will not be used for the longest period of time.

Use of this page-replacement algorithm guarantee the lowest possible page-fault rate for a fixed number of frames.

Reference string

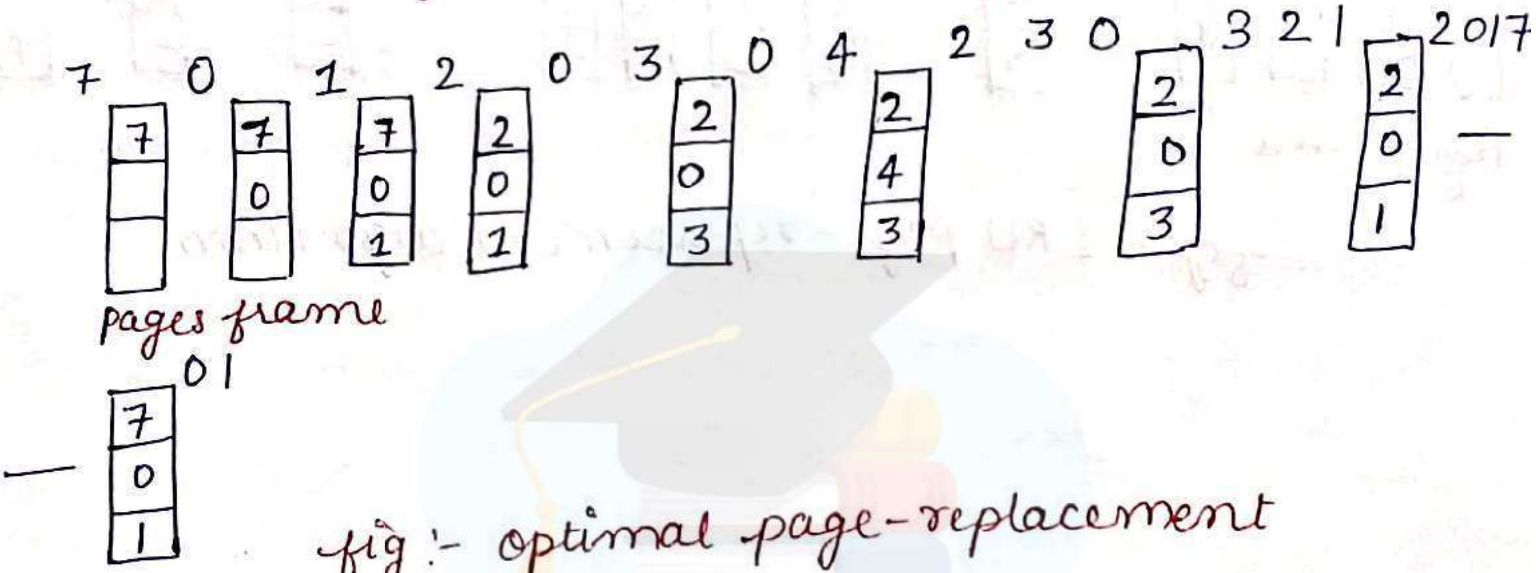


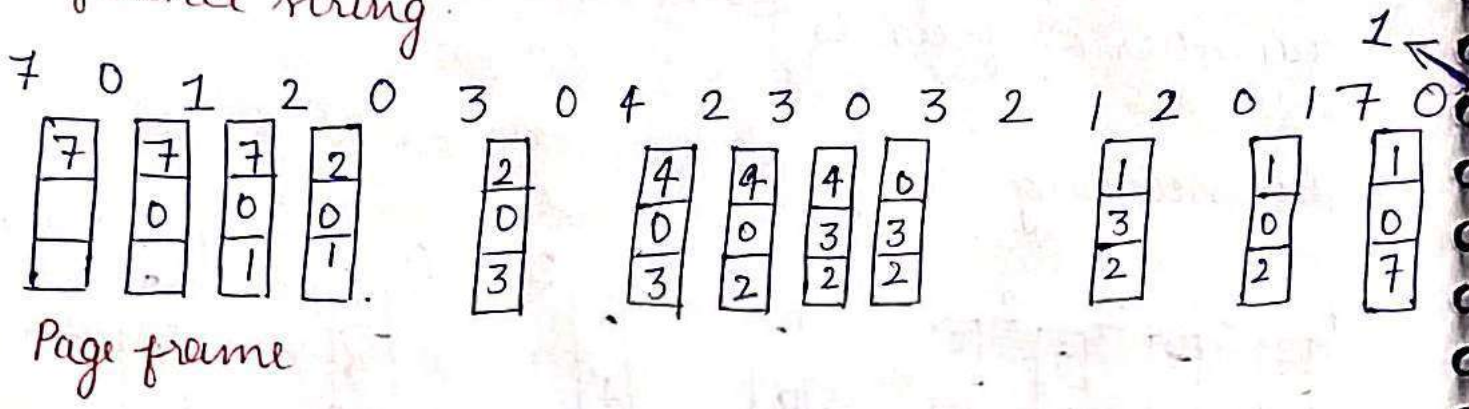
fig :- optimal page-replacement algorithm.

* LRU Page Replacement

If the optimal algorithm is not feasible, perhaps an approximation of the optimal algorithm is possible. The key distinct b/w the FIFO and OPT algorithm is ~~possible~~. ~~The key distinction b/w the FIFO~~ that the FIFO algorithm uses the time when a page was brought into memory, whereas the OPT algorithm uses the

If the recent past an approximation of the near future, then we can replace the page that has not been used for the longest period of time. This approach is the Least Recently Used (LRU) algorithm.

Reference string.



Page frame

fig:- LRU page-replacement algorithm

Unit :- 4

file management

* file :- A file is a named collection of related information that is recorded on second-ary storage such as magnetic disks, magnetic tapes and optical disk.

In general, a file is a sequence of bits, bytes, lines, or records whose meaning is defined by the file creator and user.

* file type :- file type refers to the ability of the operating system to distinguish different types of files such as text files source files and binary files etc. Many operating systems supports many types of files. operating systems like MS-DOS and UNIX the following type of files -

Ordinary files

- These are the files that contains user information.
- These may have text databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file

Directory files

- These files contains list of files names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks terminals, printers, networks, tape drive etc.

Special files

- Character special files
  [Data is handled character by character as in case of terminals or printers]
- Block-special files
  [Data is handled in blocks as in case of disks and tapes.]

File Access Mechanism

- Sequential Access
- Direct/ Random Access
- Indexed sequential Access

↳ Sequential Access:- Sequential access is that in which the record are accessed is some sequence i.e information in the file is

the other. The access method is most ~~prem~~ primit-ive one.

for ex:- Compiler usually access files in this fashion.

* Direct / Random Access.

→ Random access file organization provides, Access-ing the record directly.

→ Each Record has its own address ~~to~~ on the file with by the help of which it can be directly accessing for reading or writing.

→ The records need not be in any sequence within the file and they need not be in ~~adjacen~~ adjacent location on the storage medium.

* Indexed Sequential Access

→ This mechanism is built up on base of sequenti-al Access.

→ An index is created for each file which con-tains pointers to various blocks.

→ Index is searched sequentially and its pointer is used to Access the file directly.

* File structure

~~A file has a certain~~

A file structure should be according to a required format that the operating system can underst-and.

- A file has a certain defined structure according to its types.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organised into blocks that are ~~usted~~ understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

* Space allocation (Allocation methods)

files are allocated disk spaces by OS. OS deploy following three main ways to allocate disk space to files.

- Contiguous Allocation.
- linked allocation
- Indexed Allocation

↳ Contiguous allocation

- Each files occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement.

↳ Linked Allocation

- Each files carries a list of links to disk blocks
- Directory contains link/pointer to first block of a file.
- No external fragmentation.
- Effectively used in sequential access file.
- Inefficient in case of direct access files.

↳ Indexed allocation

- Provides solutions to problem of contiguous and linked allocations.
- A index block is created having all pointers to files
- Each file has its own index block which stores the address of space occupied by the file.
- Directory contains the addresses of index blocks of files.

* Efficiency and Performance

↳ Efficiency depends upon :-

- Disk allocation and directory algorithms
- types of data kept in file's directory entry

↳ Performance

- Disk cache :- separate section of main memory for frequently used blocks.
- free behind and read-ahead - technique to

- Improves PC performance by dedicating section of memory as virtual disk, or RAM disk.

Disk Management

* Disk
  - It is a secondary storage device that is used to store data
  - Disk provide a means to store a large amount of information for modern computer

    Ex :- Hard Disk, Solid State Drive, Floppy Disk.

* Disk structure
  - A disk is usually divided into Tracks, cylinders and sectors.
  - Hard disks drives are organised as a concentric stack of disks or "platters"
  - Each platter has 2 surfaces and two read/write heads for each surface.
  - Each platter has the same no. of tracks.
  - Platter is made from aluminium, ceramic, or class, coated with magnetic material such as Iron oxide.

* Disk Geometry

↳ Platters :- Platters resembles the phonograph records found in an old-fashioned Juke box.

equivalent increase in cost.

- Head:- Each platter is associated with read/write head.
  - They are energy converters: i.e; they transform electric signals into magnetic (write the disk) and vice-versa (read the disk)

- Tracks:- circular area of disk
  - length of a track one circumference of disk
  - Over 1000 on a hard disk.
  - Data first written to outer most track.

- Sectors:- Divide tracks selections.

- Cylinders:- logical groupings of the same track on each disk surface in a disk unit.

  OR

  All the track with the same radius are known as a cylinder.

- Clusters:- Several sectors form a cluster.
  - 64 sectors in one cluster.
  - groups of sectors used by OS

P.T.O →

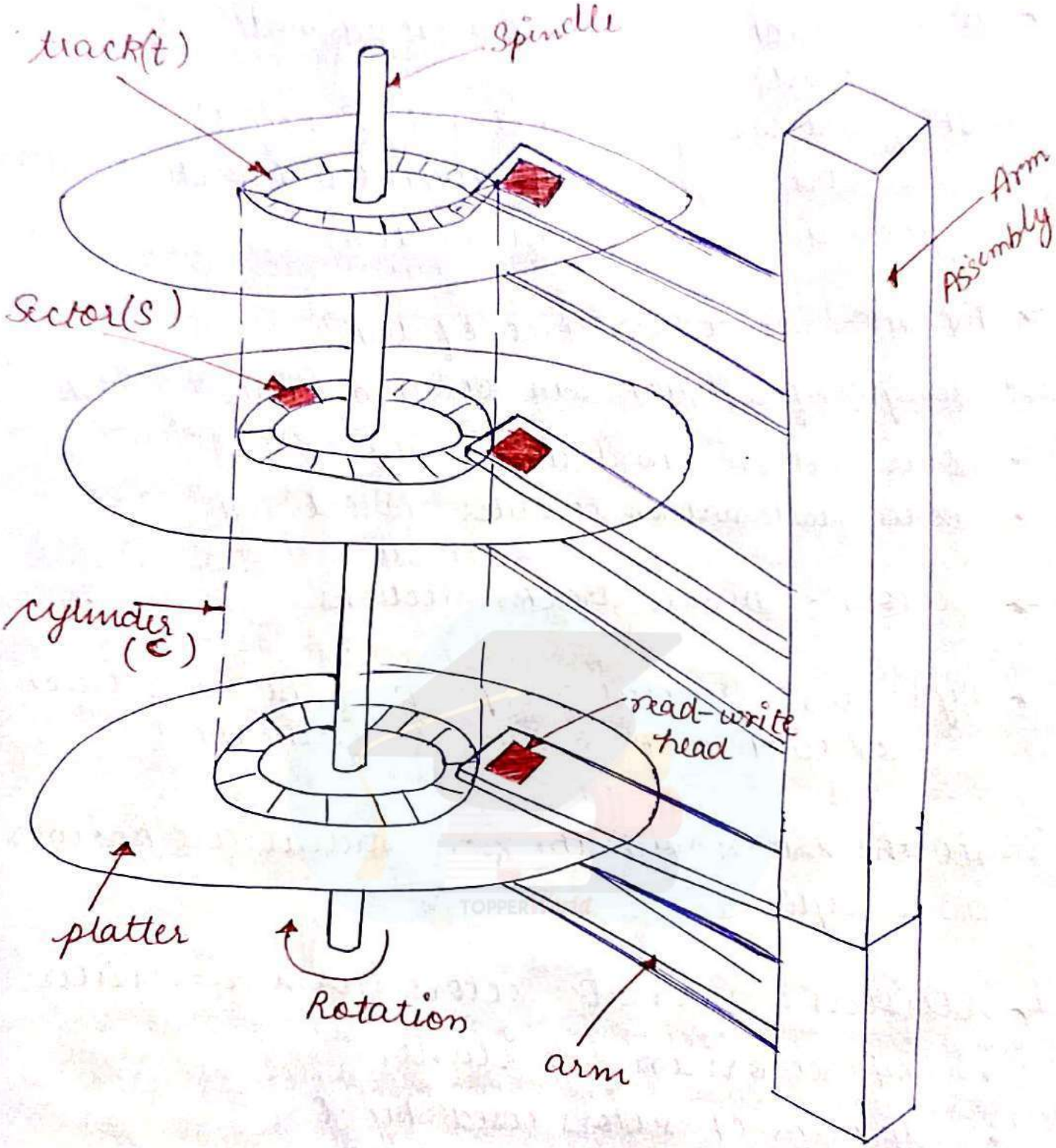


fig :- Disk structure

* Disk Scheduling :- The technique that operating system uses to determine the request which is to be satisfied next is called Disk Scheduling.

Important terms :-

1.) Seek Time :- It is the time taken in locating the disk arm to a specified track where the read and write request will be ~~specified~~ satisfied.

2.) Rotational latency :- It is the time taken by the desired sector to rotate itself to the position from where it can access the Read/write heads.

3.) Transfer Time :- Time taken to transfer the data.

4.) Disk Access Time :- Disk access time is given as,

Disk Access Time = Rotational latency + Seek Time + Transfer time

5.) Disk Respons Time = It is the average of time spend by each request waiting for the IO operation.

* Purpose of Disk Scheduling :-

To select a disk request from the Queue of IO requests and decide the schedule when this request will be processed.

* Goals of Disk Scheduling :-

- Fairness
- High throughout
- Minimal travelling head time.

* Disk Scheduling Algorithms

1. FCFS scheduling Algorithm
2. SSTF (Shortest seek time first) algorithm
3. SCAN scheduling
4. C-SCAN Scheduling
5. LOOK Scheduling
6. C-LOOK Scheduling

1.) FCFS Scheduling Algorithm

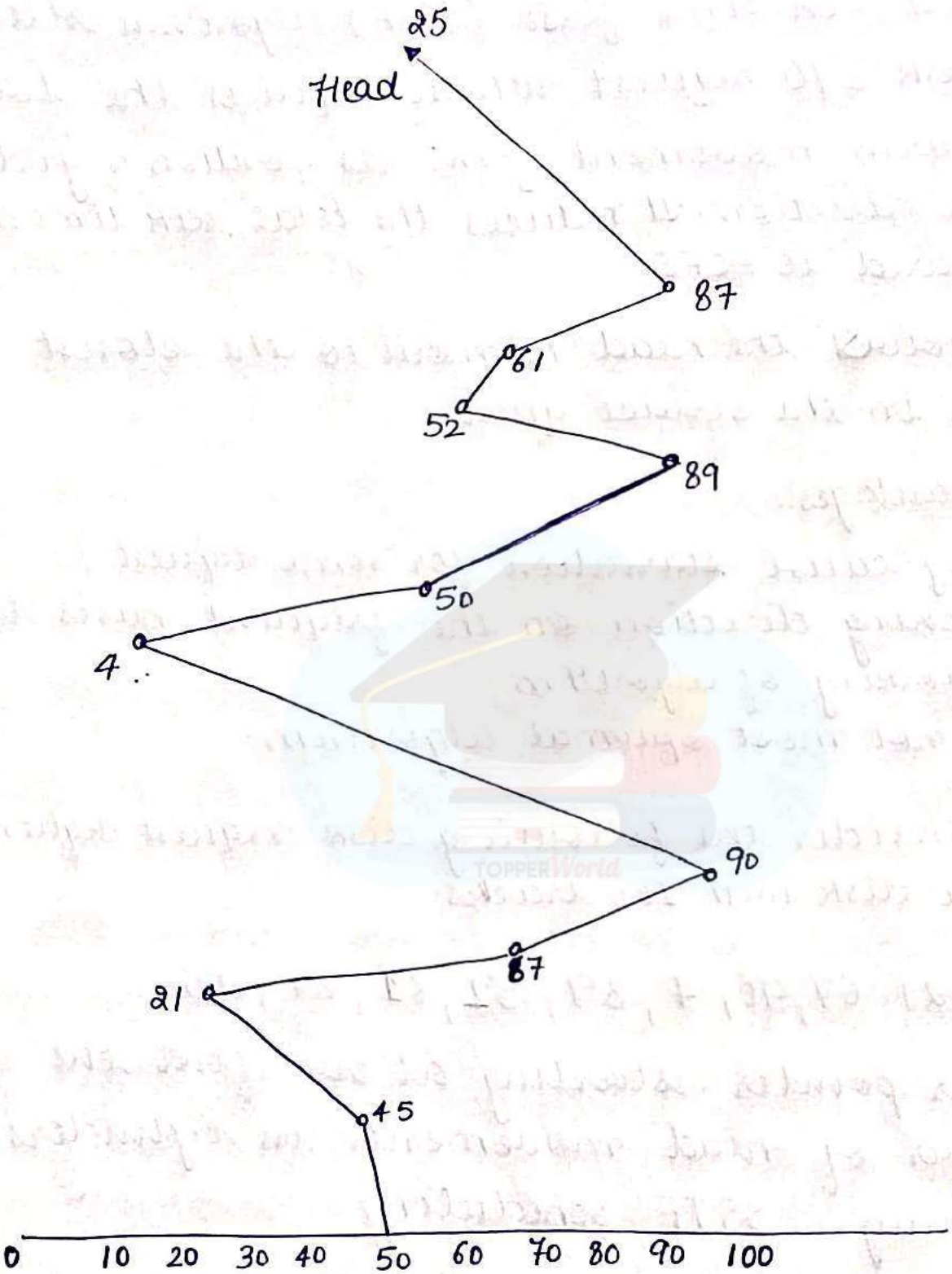
It is the simplest scheduling algorithm. It servic the IO requests in order in which they ~~have~~ arrive. There is no starvation in this algorithm, every request is serviced.

↳ Disadvantages

- The scheme does not optimize the seek time.
- The request may come from different processes therefore there is the possibility of inappropriate movement of the head.

Ex :- Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25

Head pointer starting 50 and moving in left direction find the number of head movements in cylinders using FCFS Scheduling.



No. of cylinders moved by the head.

$= (50-45) + (45-21) + (67-21) + (90-67) + (90-4) + (50-4)$

$= 5 + 24 + 46 + 23 + 86 + 46 + 49 + 9 + 26 + 62$

$= 376$

* SSTF Scheduling Algorithm

Shortest seek time first (SSTF) algorithm selects the disk I/O request which requires the least disk arm movement from its position regardless of the direction. It reduces the total seek time as compared to FCFS.

It allows the head to move to the closest track in the service queue.

Disadvantages.

- It may cause starvation for some request.
- Switching direction on the frequent Basis slows the working of algorithm
- It is not most optimal algorithm.

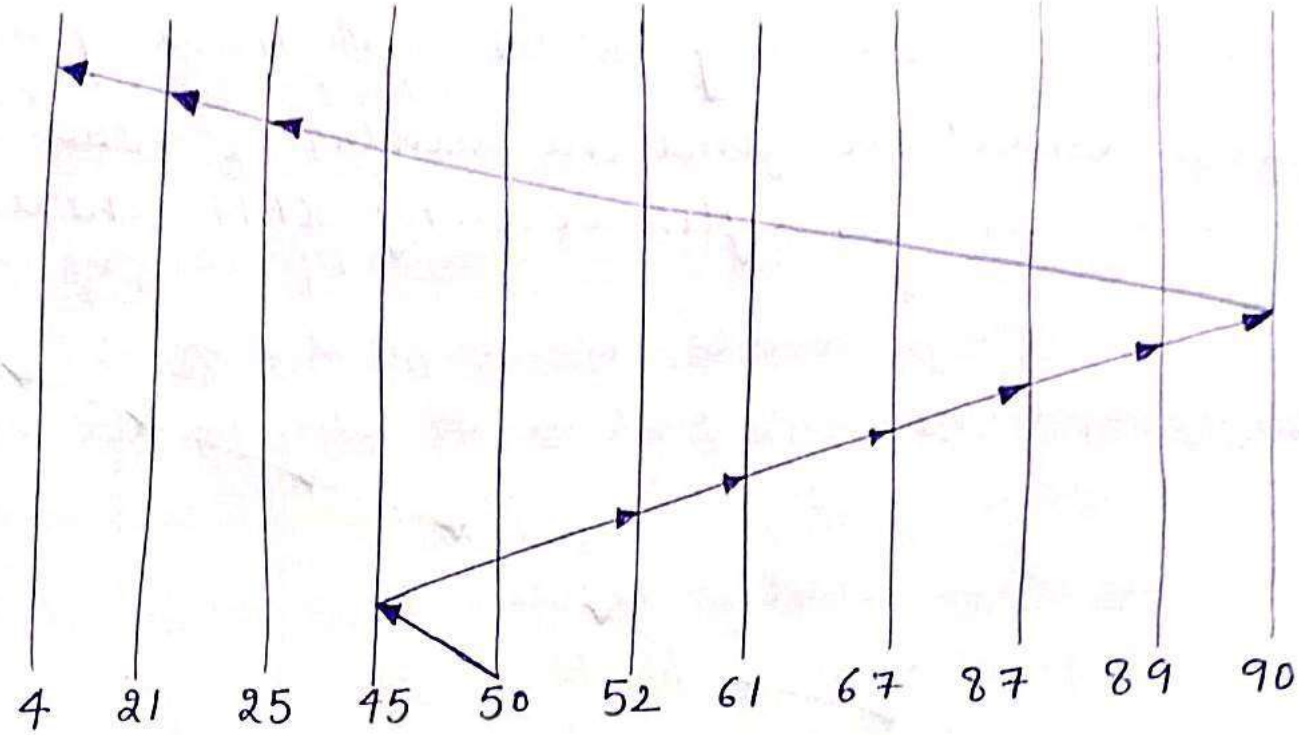
Ex :- Consider the following disk request sequence for a disk with 100 tracks.

45, 21, 67, 90, 4, 89, 52, 61, 87, 25

Header pointer starting at 50. find the number of head movements in cylinders using SSTF Scheduling

Solution

P. T. O



Number of cylinder = 5+7+9+6+20+2+1+65+4+17

= 136

* SCAN Algorithms

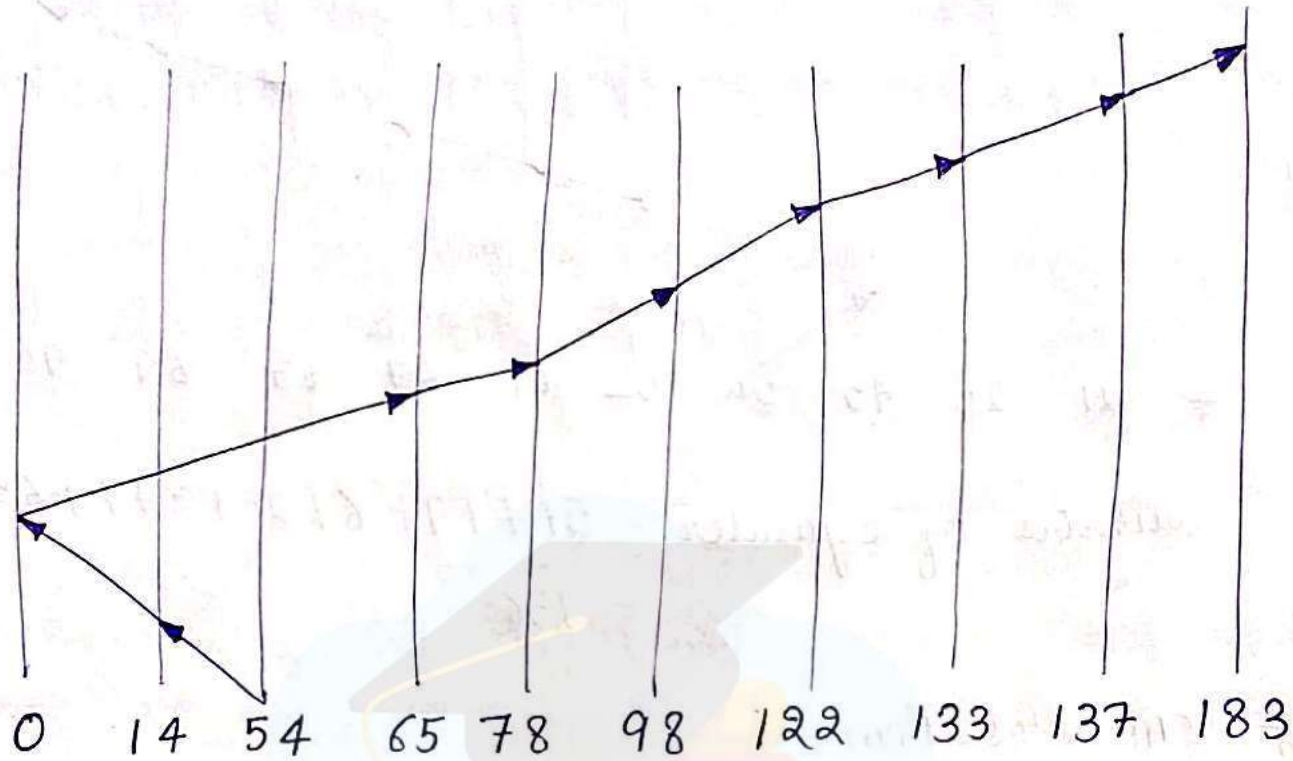
It is also called as Elevator Algorithm. In this algorithm the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path, and then it turns back and moves in the reverse direction satisfying requests coming in its path.

It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.

Example :- Consider the following disk request sequence for a disk with 100 ~~tracks~~ tracks

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction find the number of head movements in cylinder using SCAN scheduling



Number of cylinders = 40+14+65+13+20+24+11+4+46
= 237

* C-SCAN Algorithm

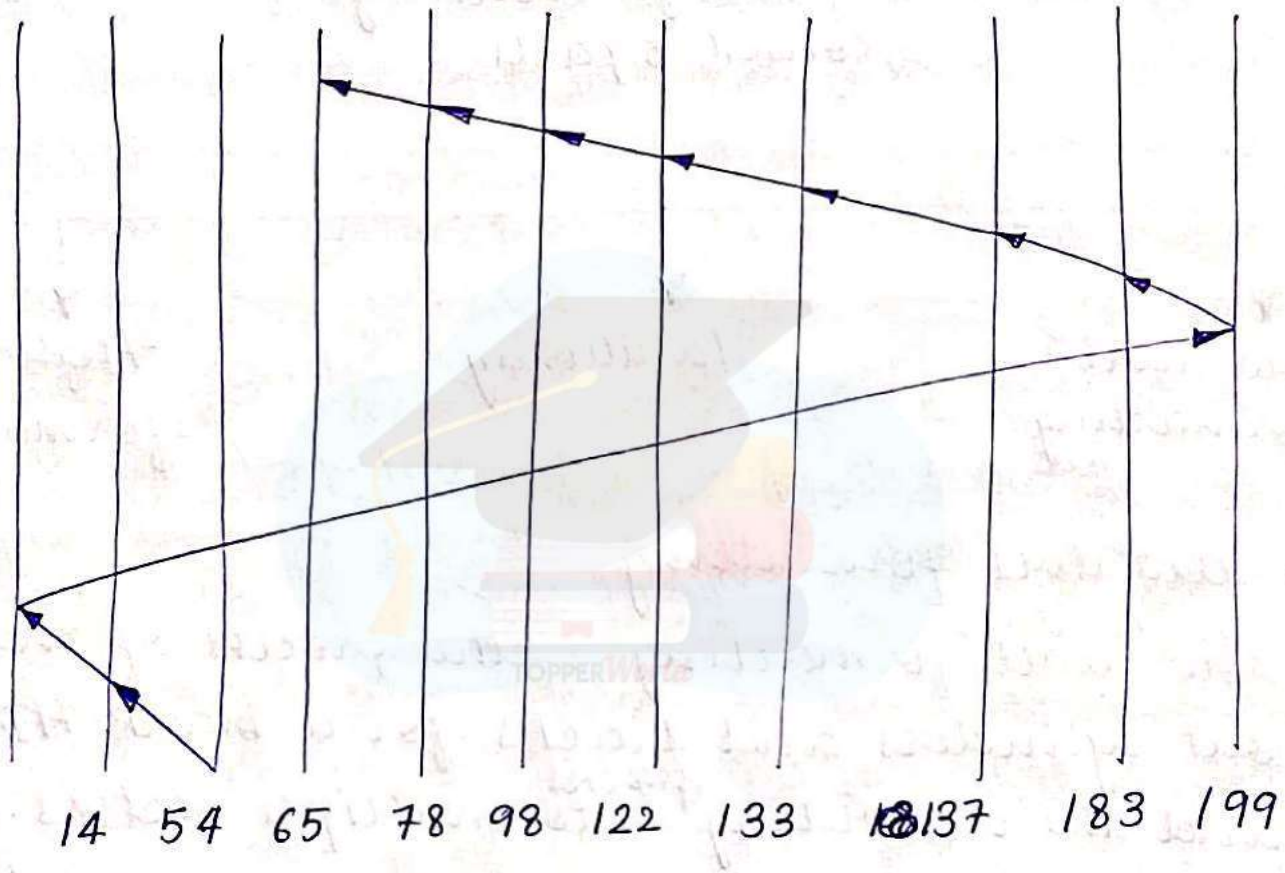
In C-SCAN algorithm, the arm of the disk in a particular direction servicing request until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns Back and start

the remaining request.

Example :- Consider the following disk request sequence for a disk with 100 tracks

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in the left direction. find the number of head movements in cylinders using C-SCAN Scheduling.



Number of cylinders crossed = 40 + 14 + 199 + 16 + 46 + 4 + 11 + 24 + 20 + 13 = 387

* Disk Reliability

Reliability is the ability of the disk system to accomodate a single-multiple disk failure and still

is the ability of the disk to efficiently provide information to the user. Adding redundancy almost always increases the reliability of the disk system.

* Disk formatting :- It is a process of configuring data storage device such as hard disk, SSD, floppy disk or USB flash drive for initial use.

Disk formatting consist 3 parts

- Low-level formatting
- Partitioning
- High-level formatting.

↳ low level formatting

low level formatting is the process of marking out cylinders and tracks for a blank HDD, and then dividing tracks into multiple sectors with sector markers. So, this is process is actually a kind of physical formatting. And it is now often performed by HDD manufacturer.

If user perform low-level formatting when data have been install, all existing files will be erased and it is almost impossible

make such a formatting to avoid privacy leakage. Never the less, performing low-level formatting will cause damage to hard-disk and shortenits service life. Therefore, this formatting is not suggested for users.

- Partition

It is the process of dividing a disk into one or more regions, the so called partition. Disk partition can be operated by users and it will affects the disk performance.

- High-level formatting

It is a process of writing a file system, cluster size, partition label, and so on for a newly created partition or volume. Its is physically done to erase the hard disk and reinstall the operating System back on the disk drive.

* Boot-Block / Boot Sector

It is a region of HDD, floppy disk, optical disk, or other storage devices that contains machine code to be loaded into ~~sectors~~ RAM by a computer system's Built in firmware. The purpose of Boot sector is to allow the boot process of a computer to load a program (usually, but not necessarily, an operating system) stored on the

corresponding to a logical disk sector) is specified by the design of the computing platform.

* Bad-Block.

A Bad sector is a sector on a computer disk drive or flash memory that cannot be used due to permanent damage (or an OS inability to successfully access it) such as physical damage to the disk surface (or sometimes sector become stuck in a magnetic or digital state that cannot be reversed) or failed flash memory transistors. It is usually detected by a disk utility software such as CHKDSK or SCANDISK on Microsoft systems, or bad blocks on UNIX-like Systems. when found, these programs may mark the sectors unusual (all the systems contain provision for bad sector marks) and the OS skips them in the future.