PAGE NO.:
DATE: 17 / Jan / 2019

UNIT - 1

★ **Operating System** – An operating system is a system program that enables the computer hardware to communicate and operate with the computer software.

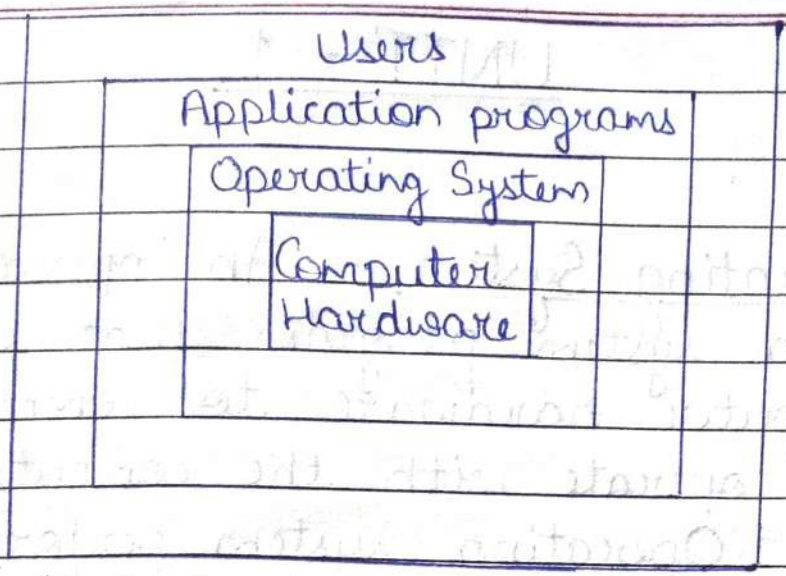
Operating system perform basic task such as recognising input from the keyboard and sending output to the display screen, keeping track of files and directories on the disk and controlling peripheral devices such as disk driver and printer.

The operating system is also responsible for security ensuring that unauthorise users do not access the system.

Some examples of operating system are – Unix, MS-DOS, MS-Windows, Windows-NT, VMS and UM.

"An operating system is a program (system software) which act as an interface between user and computer hardware."

A computer system can be logically divided into 4 components –



(Logical Computer Hierarchy)

* Functions of operating system :-

1.) Memory management function -

- Primary (main) memory - Find free space in memory and allocate it to different process. Provide direct access storage for CPU. Process must be in main memory to execute.

- Operating system must -

(i) Decide when to load each process into memory.

(ii) Decide how much memory space to allocate each process.

(iii) Decide when a process should be remove from memory.

PAGE NO:
DATE: / /

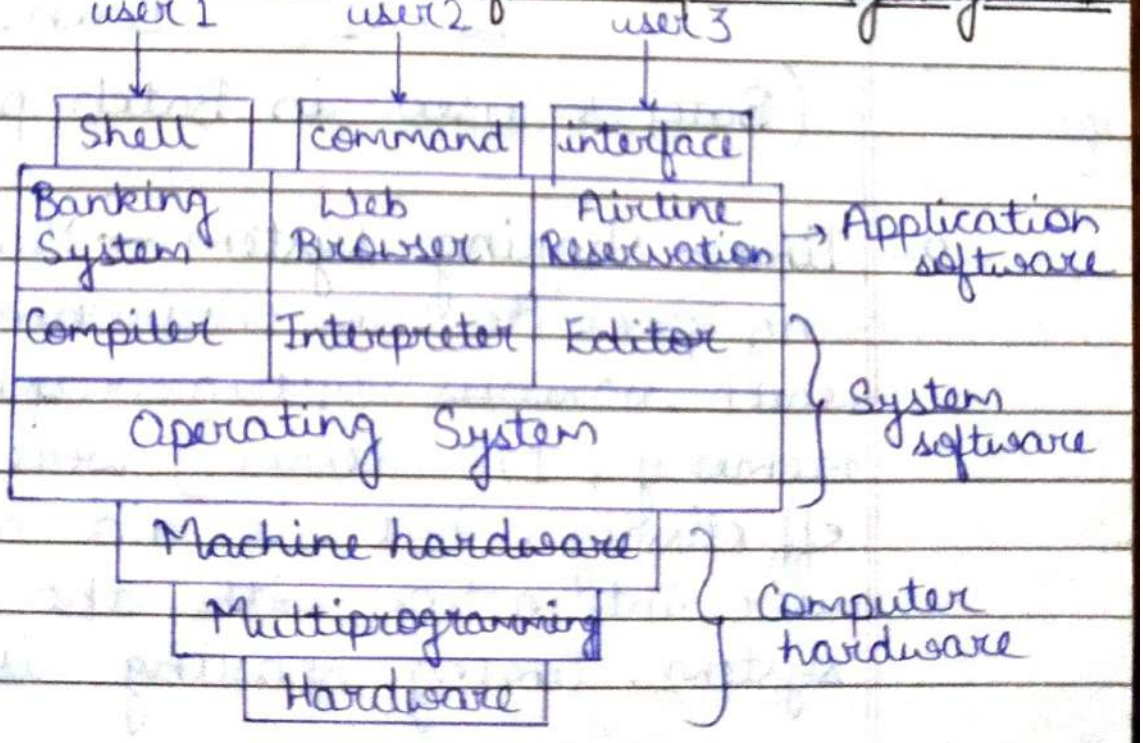
(iv) Protect memory space.
(v) Allocate - deallocate space for process.

2.) Process management function – Allocates the processor to execute a selected process.

3.) Input-Output device management function- It allocates a device to process.

4.) File management function – Keep track of all information about files, how they are open and close.
A file system supports directories which contain file and other directories name, size, data created, last modified etc.

* Layered abstraction of an operating system-



Multiprogramming → Context switch

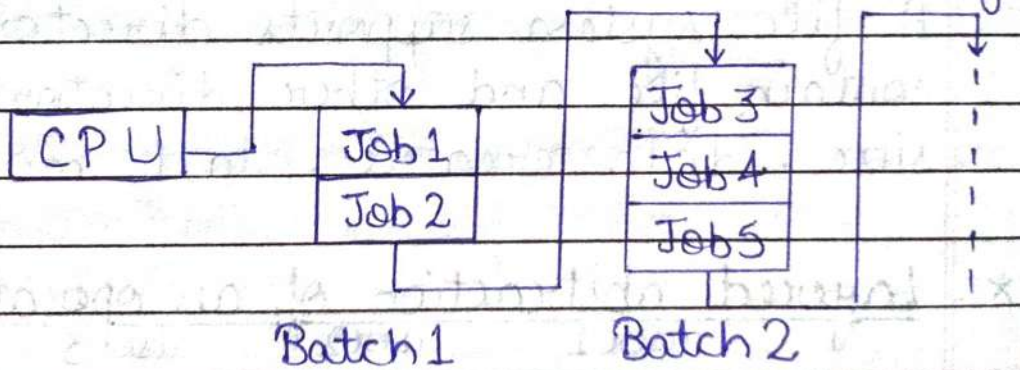
Multitasking → Time-sharing

PAGE NO :

DATE : 19 / Jan / 2019

1- Batch system
2- Time sharing system
3- Real time system
4- Parallel system
5- Distributed system

1- **Batch processing system** – Batch is define as a group of jobs with similar name.
Computer executes each batch sequentially processing all jobs of a batch considering them as a single process is called batch processing.



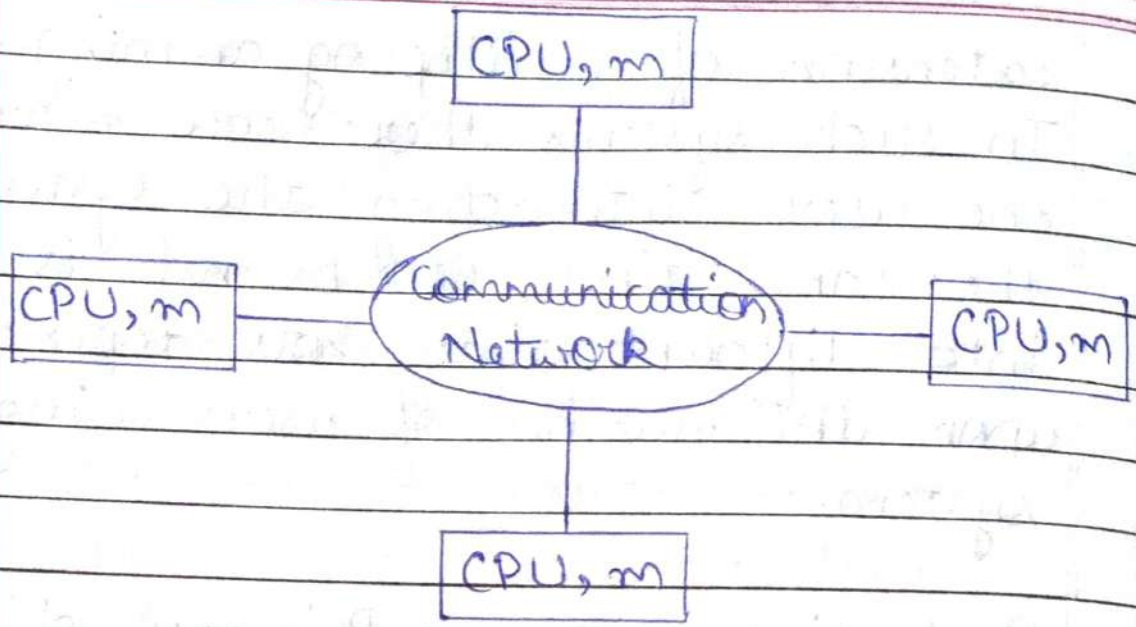
(Batches used in Batch processing)

2- **Time-sharing system** – Multiprogramming provide an environment in which various system resources (CPU, memory, I/O device) were utilize effectively, but it did not provide user interaction with the computer system. Timing sharing is a logical

extension of multiprogramming.
In such systems there are more than one user interacting the system at the same time. CPU bound is divided into different time slots depending upon the number of users using the system.

3- Real-time system — Primary objective of real time system is to provide quick response time and meet a scheduling dead line.
Real time system has many events that must be accepted and process in a short time or within certain deadline.

4- Distributed system — Distributed operating system are the operating system for a network of computers connected by communication network through a message passing mechanism.
A distributed operating system manage the hardware and software resources of a distributed system.



5- Parallel system - Parallel operating system are the interface between parallel computers and the applications that are executed on them. They translate the hardware capabilities into concepts usable by programming language.

Client - server system - A client - server application is a distributed system made up of both client and server software.
Client - server communication involves two components, namely a client and a server. There are usually multiple clients in communication with a single server. The clients send requests to the server and the server

responds to the client request.

* **Memory management :-** It is the functionality of an operating system which manage primary memory and move process back between main memory and disk during execution.

Functions of memory management -

1- Keep track of every memory location.
2- Track of how much memory is allocated.
3- Takes the decision, which process will get memory and when.
4- It updates the memory status when it is allocated or free.

* **Physical and logical address space -**

Logical address space - It is generated by the CPU. It is also known as virtual address.

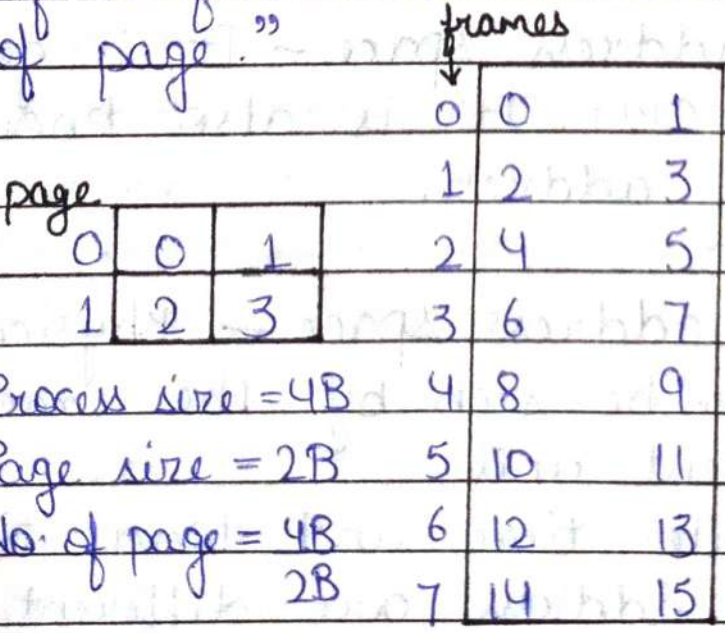
Physical address space - Physical address space can be seen by the memory management unit.
In compile time and load time memory address are different in

execution time, physical address space are same.

* **Paging :-** Paging is a memory management scheme that remove the requirement of contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non-contiguous.

The physical memory is divided into a number of fix size blocks called frames and the logical address space is also divided into fix size block called page.

"When a process is to be executed its pages are loaded into any available memory frames. The size of a frame is same as the size of page."



PAGE NO:
DATE: / /

If the capability of physical memory is m and size of each page is P, then the number of frames in physical memory will be

$$f = \frac{m}{P}$$

Ex- Consider a logical address space of 8 page of 1024 words. Each map onto a physical memory of 32 frames, then -

1.) How many bits are there in physical address?

2.) How many bits are there in logical address?

Sol: Let the number of bits in the physical memory = m

Then the size of physical memory = 2^m

Given

no. of pages = $8 = 2^3$

no. of frames = $32 = 2^5$

Size of each frame = size of each page

$= 1024$

$= 2^{10}$

$$f = \frac{m}{P}$$

$$2^5 = \frac{2^m}{2^{10}}$$

$2^m = 2^{10} \times 2^5$

$\boxed{m = 15}$

no. of page = $\frac{\text{size of logical memory}}{\text{size of each page}}$

$2^3 = \frac{2^n}{2^{10}}$

$2^n = 2^3 \times 2^{10}$

$\boxed{n = 13}$

★ **Segmentation** – Segmentation is another technique for the non-contiguous storage allocation. It is different from paging as pages are physical in nature and hence are of fix size, whereas segments are logical division of a program and hence are of variable size.

It is a memory management scheme that support the user view of memory rather than system view of memory. Each segment has a name and length which is loaded into physical memory.

★ **Virtual memory** – It is a virtual resource of a computer. It is an

illusion that a computer system resources process more memory that is actually having. This makes a process independent of the size of real memory (main memory). It also permits large number of process to share a computer system without constraining each other.

* Demand Paging :- In demand paging, a page is break into the memory for its execution only when it is demanded otherwise it is remain in backing storage (disk).
This method is the combination of paging and swapping method. The main requirement of this method is that the complete program should be present in the backing storage in the form of page.

- Swapping - Swapping is used to swap the content of the program from disk to main memory when it is required.

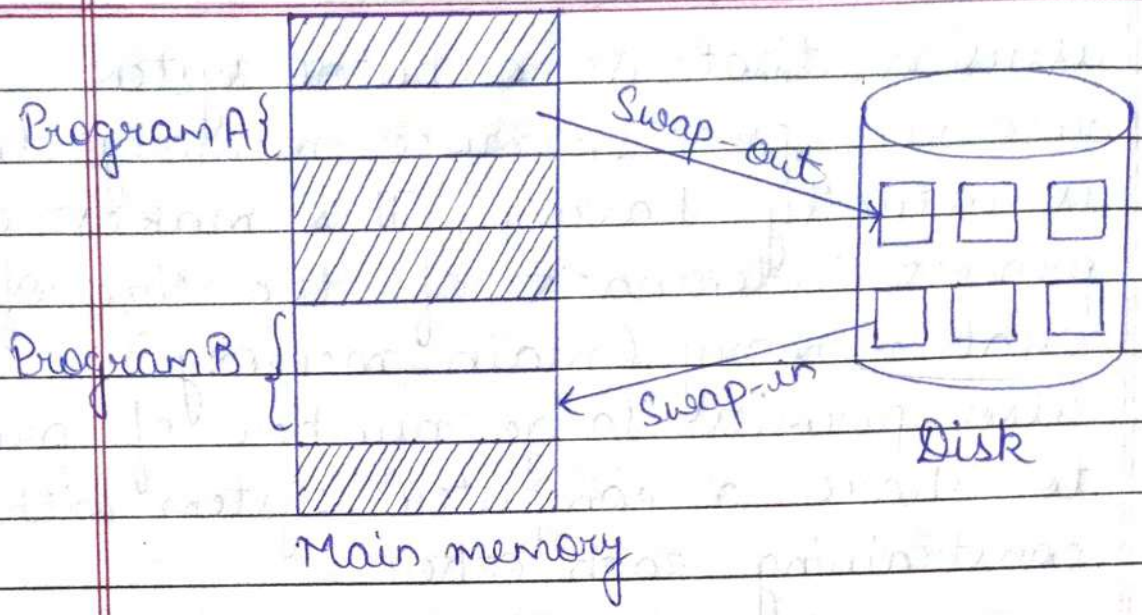


fig:- Swapping with paging

Imp.

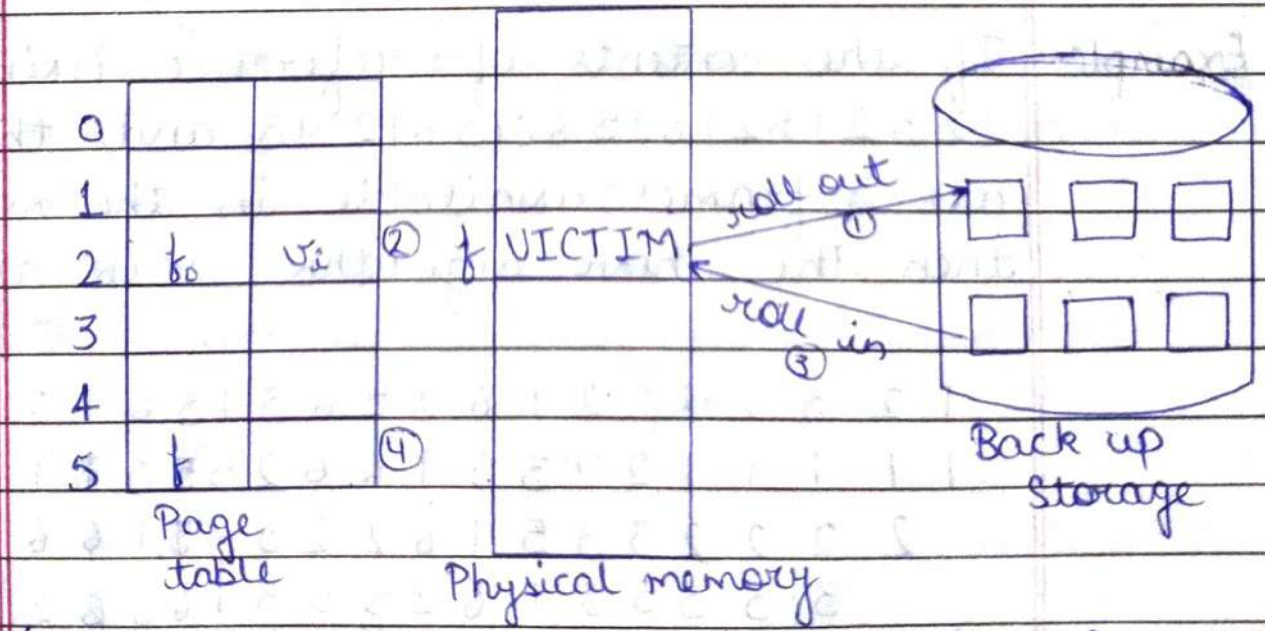
★ Page Replacement Algorithms :-

Principle of page replacement – The page replacement is done by swapping. The required page from back-up storage to main memory and vice-versa. This swapping is done by checking the content of physical memory if there is a free frame in the memory then swap-in the required page into the frame is free.

In case there is no free frame in physical memory then first find the frame which is not currently in use. The content of this frame is

swap-out from memory to back-up storage, then bring the required page in the frame which is now free.
Following steps are performed in page fault routine for page replacement –

1- Find the location of desired page on back-up storage.
2- Find the free frame.
(a) If the frame is free, use it.
(b) Otherwise find the frame which is not currently being used this is called as victim frame.
(c) Write the content of victim frame on back-store and change the page-table entries to indicate that the page is no longer in the main memory.
3- Read the desired page into free frame, change page table and frame entries.

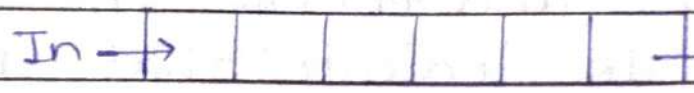


(fig:- Example of page replacement)

PAGE NO :
DATE : / /

1- Swap out the victim frame.
2- Update its page table entry.
3- Swap in the desired page.
4- Update its page table entry.

* 1.) FIFO Replacement Algorithm – In this method, the oldest page present in the memory is selected for replacement. For this the page at the head of the queue is replace and when the page is break into the main memory then it is place at the end tail of the queue.



(fig:- Queue used in Page Replacement Algorithm)

mple– If the contents of reference using is 1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3 and there are 3 frames available in the memory then the above algorithm work as –

1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
1	1	1	1	1	2	2	3	5	1	6	6	2	5	5	3	3	1	6	2
	2	2	2	2	3	3	5	1	6	2	2	5	3	3	1	1	6	2	4
		3	3	3	5	5	1	6	2	5	5	3	1	1	6	6	2	4	3
*	*	*			*		*	*	*	*		*	*	*	*		*	*	*

PAGE NO :
DATE : 2nd / Feb / 2019

Total no. of page fault = 14

2.) Optimal Replacement Algorithm – This algorithm has lowest page fault rate for all algorithm of page replacement.
"This algorithm states that replace that page which will not be use for longest period of time."
Future knowledge of reference string is required to understand this algorithm following example is consider –

1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	2	2	2
	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	4	4
		3	3	3	5	5	5	5	5	5	5	3	3	3	3	3	3	3	3
*	*	*			*			*				*	*				*	*	

Total no. of page fault = 9

3.) LRU (Least Recently Used) Algorithm – In this algorithm, the page has been not used for longest period of time is selected for replacement.
To understand this concept consider the following example –

PAGE NO :
DATE : / /

1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3	
1	1	1	1	3	2	1	5	2	1	6	2	5	6	6	1	3	6	1	2	
	2	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	
		3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3	
*	*	*			*			*		*		*	*				*	*	*	

Total no. of page fault = 11

Ques- 7012030423O3

- FIFO Replacement -

7	0	1	2	0	3	0	4	2	3	0	3	
7	7	7	0	0	1	2	3	0	4	2	2	
	0	0	1	1	2	3	0	4	2	3	3	
		1	2	2	3	0	4	2	3	0	0	
*	*	*	*		*	*	*	*	*	*		

Total no. of page fault = 10

- Optimal Replacement -

7	0	1	2	0	3	0	4	2	3	0	3	
7	7	7	7	7	0	0	2	2	2	3	3	
	0	0	0	0	2	2	3	3	3	4	4	
		1	2	2	3	3	4	4	4	0	0	
*	*	*	*		*		*			*		

Total no. of page fault = 7

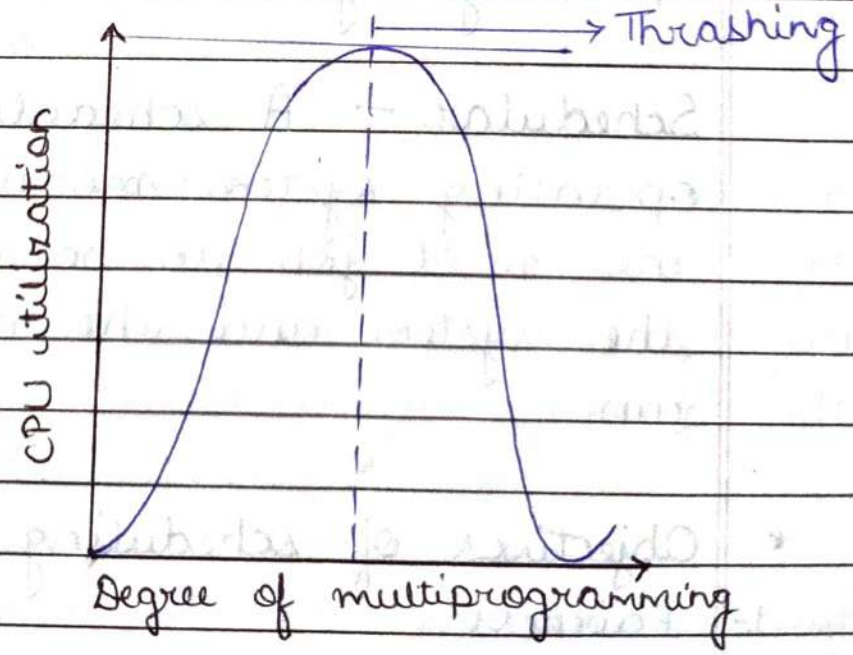
PAGE NO:
DATE: / /

- LRU –

7	0	1	2	0	3	0	4	2	3	0	3
7	7	7	0	1	2	2	3	0	4	2	2
	0	0	1	2	0	3	0	4	2	3	0
		1	2	0	3	0	4	2	3	0	3
*	*	*	*		*		*	*	*	*	

* **Thrashing :-** The phenomena of moving page from primary to secondary storage or vice-versa consume a lot of compute energy but accomplishes very few results. This situation is called thrashing.

The process is in thrashing if it is spending more time in paging instead of their execution.



(fig :- Thrashing)

PAGE NO :
DATE : 6 / Feb / 19

UNIT – 2

CPU Scheduling

* **CPU Scheduling :-**
Almost all computer resources are require to be schedule before their uses. CPU is one of the primary resource. So, its scheduling is necessary before its uses. The scheduling done by CPU is the basis of multiprogram. Operating system scheduling refered to set of policies and mechanism built into the operating system.

Schedular – A schedular is an operating system module that select the next job to be admitted into the system and the next process to run.

* **Objectives of scheduling –**
1- Fairness
2- Throughput
3- Resources
4- Priorities

PAGE NO :
DATE : / /

* Scheduling Criteria :-
Many criteria have been suggested for comparision of CPU scheduling algorithm. The criteria includes the following -

1- CPU utilization - We have to keep the CPU as busy as possible. It may ranged 0 to 100%. In real time system it should range from 40 to 90% for lightly loaded and heavily loaded system.

2- Throughput - It is the measure of work in terms of number of process completed per unit time for long process. This range may be 1 process per hour. For short transaction, throughput may be 10 process per second.

- Turn around time - It is the sum of time periods spend in waiting to get into memory, waiting in ready queue, executing on the CPU and doing I/O time.

- Waiting time - The CPU scheduling algorithm does not affect the amount of time during, which a process execution waiting time. It is the sum of time periods

spend waiting in ready queue.

- **Responsed time** – It is the amount of time, it takes to start responding not the time taken to output that response.

* Types of Scheduling :-

1- Preemptive scheduling schemes.
2- Non-preemptive scheduling schemes.

CPU scheduling decisions may take place under the following four circumstances –

(1) When a process switch from the running state to the ~~ready~~ waiting state (For example I/O wait).
(2) When a process switch from the running state to the ready state (For example when an interrupt occur).
(3) When a process switch from the waiting state to the ready state (For example completion of I/O).
(4) When a process terminates.

In circumstances (1) and (4) there is no choice in terms of scheduling. When scheduling takes place under (1) and (4)

circumstances, we say the scheduling scheme is non-preemptive.
However, in circumstances (2) and (3) there is a choice in terms of scheduling. process may or may not be selected for its execution. This type of scheduling is called preemptive.

* Scheduling Algorithm :-

CPU scheduling deals with the problem of deciding which of the process in the ready queue is to be allocated to the CPU. Now, we describe several of the many CPU scheduling algorithm exist.

(1)- First Come First Serve (FCFS) -

FCFS is the simplest of all scheduling algorithm. The key concept of this algorithm is allocated the CPU in the order in which the process arrive.

Ques - Consider the following set of process having their burst time mention in milliseconds. CPU burst time indicates that how much time the process need the CPU.

Process	Burst time (ms)
P_1	20
P_2	7
P_3	5

PAGE NO :
DATE : / /

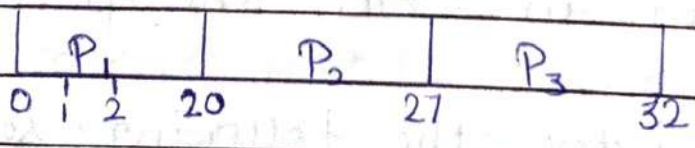
Calculate the average waiting time if the process arrive in the order of –

1- P_1, P_2, P_3 – (0, 1, 2)
2- P_2, P_3, P_1 – 0
3- P_3, P_1, P_2 – 2
4- P_3, P_2, P_1 – 0

Solution:- (1) If process arrive in order P_1, P_2, P_3. Let us assume at 0 ms, 1 ms and 2 ms respectively.

Process	Burst time	Arrival time
P_1	20	0
P_2	7	1
P_3	5	2

Since, FCFS is a non-preemptive algorithm so when a process execution is started, it is not preempt~~ive~~ until its execution is completed.



Waiting time for process P_1 = 0 ms
Waiting time for process P_2 = 20 – 1 = 19 ms
Waiting time for process P_3 = 27 – 2 = 25 ms

Average waiting time = $\frac{0+19+25}{3}$ = 14.67 ms.

PAGE NO:
DATE: / /

Consider a case if the process P_1, P_2, P_3 arrive at the same time say at 0 ms in the system in the order of P_1, P_2, P_3.

Waiting time for process P_1 = 0 ms
Waiting time for process P_2 = 20 ms
Waiting time for process P_3 = 27 ms

Average waiting time $= \frac{0+20+27}{3} = \frac{47}{3}$

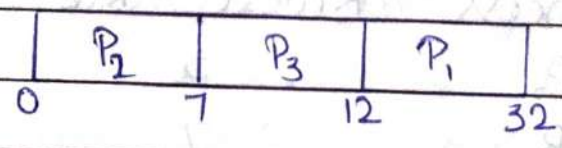
$= 15.6$ ms

Sol 2:- If process arrive in order P_2, P_3, P_1. Let us assume at 0 ms all process.

Process	Burst time	Arrival time
P_2	7	0
P_3	5	0
P_1	20	0

Since, FCFS is a non-preemptive algorithm so when a process execution is started, it is not preemptive until its execution is completed.

The grant chart will be-



Waiting time for process P_2 = 0 ms
Waiting time for process P_3 = 7 ms
Waiting time for process P_1 = 12 ms

PAGE NO :
DATE : / /

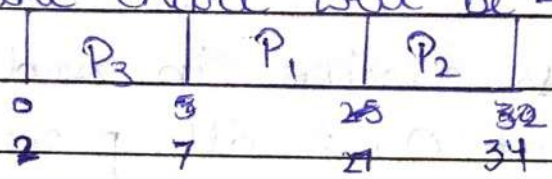
Average waiting time = $\frac{0+7+12}{3} = \frac{19}{3}$

= 6.34 ms.

(3) If process arrive in order P_3, P_1, P_2 and arrival time is 2 ms for all process.

Process	Burst time	Arrival time
P_3	5	2
P_1	20	2
P_2	7	2

The grant chart will be –



Waiting time for process P_3 = 2 ms

Waiting time for process P_1 = 5–2 = 3 ms

Waiting time for process P_2 = 25–2 = 23 ms

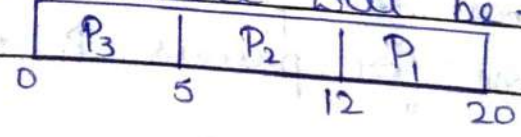
Average waiting time = $\frac{2+3+23}{3} = \frac{28}{3}$

= 9.34 ms.

(4) If process arrive in order P_3, P_2, P_1 and arrival time is 0 ms for all process.

Process	Burst time	Arrival time
P_3	5	0
P_2	7	0
P_1	20	0

The grant chart will be –



PAGE NO:
DATE: 11 / Feb / 2019

Waiting time for process P_3 = 0 ms
Waiting time for process P_2 = 5 ms
Waiting time for process P_1 = 12 ms
Average waiting time = $\frac{0+5+12}{3} = \frac{17}{3} = 5.67$ ms

(2)- **SJF (Shortest Job First) Algorithm -**

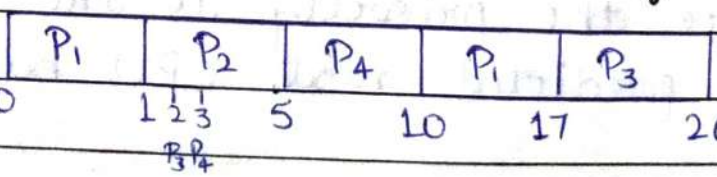
Shortest job first scheduling is a different approach to CPU scheduling. This algorith associate with each process the length of the next CPU burst. When the CPU is available it is assigned to the process that has the smallest next CPU burst. If the two process have same length of CPU burst then FCFS scheduling algorithm is follow.

Example:-

Process	Arrival time	Burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Calculate average waiting time in
1- Preemptive SJF scheduling
2- Non-preemptive SJF scheduling

Sol 1-

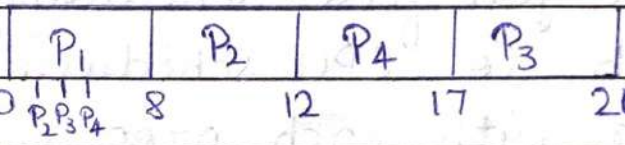


PAGE NO :
DATE : / /

Waiting time for process $P_1 = 0 + (10-1) = 9$ ms
Waiting time for process $P_2 = 1-1 = 0$ ms
Waiting time for process $P_3 = 17-2 = 15$ ms
Waiting time for process $P_4 = 5-3 = 2$ ms
Average waiting time $= \frac{9+2+15}{4} = \frac{26}{4}$

$= 6.5$ ms.

Solution 2-



Waiting time for process $P_1 = 0$ ms
Waiting time for process $P_2 = 8-1 = 7$ ms
Waiting time for process $P_3 = 17-2 = 15$ ms
Waiting time for process $P_4 = 12-3 = 9$ ms
Average waiting time $= \frac{7+15+9}{4} = \frac{31}{4}$

$= 7.75$ ms.

(3)- **Priority Scheduling** -

A priority is associated with each process and the CPU is allocated to the process with the highest priority. Equal priority process are scheduled in FCFS order.

A SJF is simply a priority algorithm where the priority is the inverse of the predicted next CPU burst.

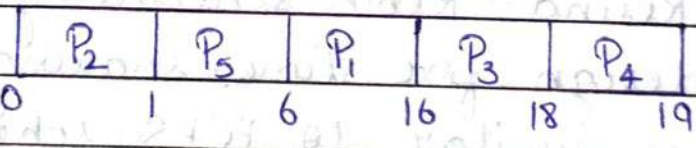
PAGE NO:
DATE: / /

Ex- Consider the following set of process assume to have arrive at time 0 in the order P_1, P_2, P_3, P_4, P_5 with the length of the CPU burst time given in milliseconds.

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

using Priority scheduling, calculate average waiting time.

Solution- Using priority scheduling we would schedule the given process according to the following grant chart -



Waiting time for process P_2 = 0 ms
" " " " P_5 = 1 ms
" " " " P_1 = 6 ms
" " " " P_3 = 16 ms
" " " " P_4 = 18 ms

Average waiting time = $\frac{1+6+16+18}{5} = \frac{41}{5}$

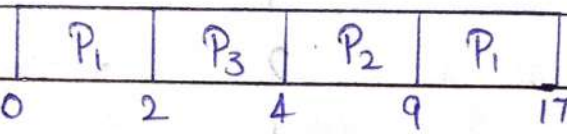
= 8.2 ms.

PAGE NO :
DATE : / /

Ques- Preemptive Priority Scheduling -

Process	Burst time	Priority	Arrival time
P_1	10	3	0
P_2	5	2	1
P_3	2	1	2

olution:- The grant chart will be -



Waiting time for process $P_1 = 0 + (9-2) = 7$ ms

" " " " $P_3 = 2 - 0 = 0$ ms

" " " " $P_2 = 4 - 1 = 3$ ms

Average waiting time = $\frac{0+7+3}{3} = \frac{10}{3} = 3.33$ ms.

(4)- **Round-Robin scheduling Algorithm (RR):-**

The Round-Robin scheduling algorithm is design for time-sharing system. It is similar to FCFS scheduling algorithm but preemption is added to switch between process.

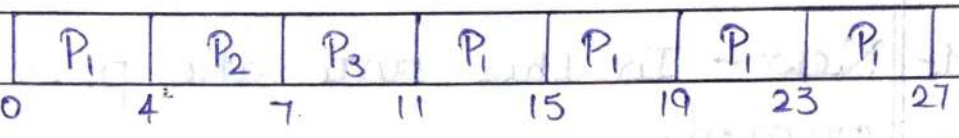
A small unit of time called a time quantum (or time slice) is define where range is kept between 10 to 100 ms.

Example- Consider the following set of process that arrive at time 0 ms

Process	Burst time
P_1	20
P_2	3
P_3	4

If we use time quantum of 4 ms then calculate the average waiting time using RR scheduling.

Sol:- The grant chart will be—



Waiting time for process $P_1 = 0 + (11-4) = 7$

" " " " $P_2 = 4$

" " " " $P_3 = 7$

Average waiting time $= \frac{7+4+7}{3} = \frac{18}{3} = 6$ ms.

★ **Process :-**

A process is an instance of a program in execution. A process in the smallest unit of work individually schedulable by an operating system. Such a system consist of a collection of process. Operating system process execute system code and user process execute user code. All these process execute concurrently.

Process is not same as program. A program is a passive entity while a process is an

active entity.

Imp:

* **Process States** – When a process comes in execution it change its state. The state of a process is defined in part by the current activity of that process. Each process may be one of the following state during the time of its execution.

1- **New** – In this state the process is being created.

2- **Ready** – In this state the process is waiting to be assign to a processor through a short term scheduler.

3- **Waiting** – In this state the process is waiting or blocked for some event to occur such as input – output completion or reception office signal.

4- **Running** – In this state instructions are being executed.

5- **Terminated** – In this state the process has finished the execution.

The state transition diagram corresponding to these states is represented in figure

below —

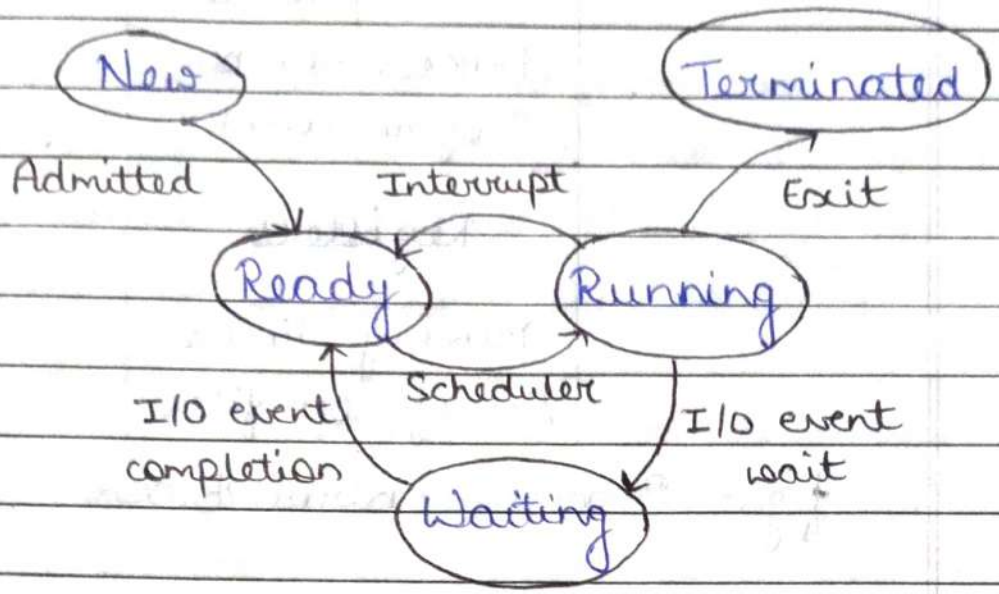


Fig:- Process states Transition Diagram

★ Process Control:-

Process control block — In operating system, each process is represented by a process control block or a task control block.

It is a data-structure that physically represent a process in the memory of a computer system. It contains many pieces of information associated with a specific process that include the following —

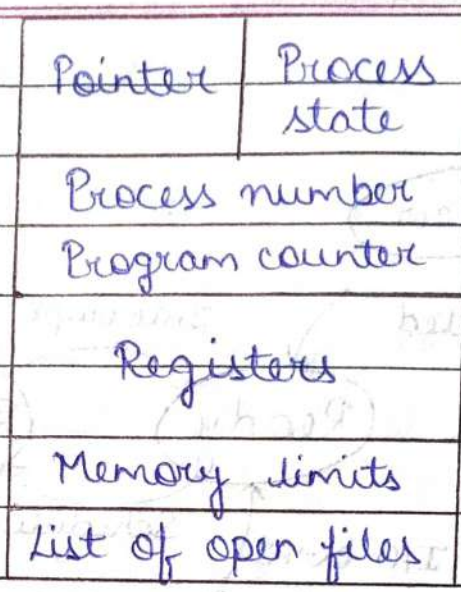


fig:- Process Control Block

1- Process state — The state may be new, ready, waiting, running or terminated.

2- Program counter — The counter indicates the address of the next instructions to be executed for this process.

3- CPU registers — The registers vary in numbers, size and type depending upon the computer architecture. They include accumulators, index registers, flags, stack pointers and other general purpose registers and any condition called information.

4- CPU scheduling information — This information include a process priority, pointers to scheduling queue and any

other scheduling parameters.

5- Memory management information — This information include the value of base and limit registers, the page of segmentable use by the operating system.

6- Input - output information — This information include the list of input - output device allocated to this process, a list of open files and so on.

* **Critical section :-**

The critical section in a code segment where the shared variable can be access. Atomic action is required in a critical section. Only one process can execute in its critical section at a time, all other process have to wait to execute in their critical sections.

The critical section is given as follow -

```
do
{
  Entry section
  Critical section
  Exit section
  Remainder section
```

PAGE NO :
DATE : / /

} while (TRUE);

In the above diagram the entry section enters the entry into the CS. It aquire the resources needed for execution by the process. The exit section handles the exit from the critical section. It release the resources and also inform the other process that CS is free.

The CS problem need a solution to synchronize the different process. The solution to the CS problem must satisfy the following conditions :-

1- **Mutual exclusion** - It implies that only one process can be inside the CS. at any time. If any other process require the CS, they must wait until it is free.

2- **Progress** - Progress means that if a process is not using the CS then it should not stop any other process from accessing it. In other words, any process can enter a CS if it is free.

3- **Bounded waiting** - Bounded waiting

means that each process must have a limited waiting time, it should not wait endlessly to access the CS.

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

Process synchronization was introduced to handle problems that arose while multiple process executions. Some of the problems are critical section problem, mutex locks.

Ex- What be happen if at the same time, same process use the same resource?

```
P()
{
R(a)
a = a + 1;
W(a)
}
```

a = 10, Process P_1, P_2

Systematic manner	Distributed manner
$P_1 = 11$	$P_1 = 10$
$P_2 = 12$	$P_2 = 10$

PAGE NO :
DATE : 7 / Mar/2019

* **Race Condition :-** Race condition is a special condition that may occur during of same data at same time.

Race condition is an undesirable that occurs when a device or system attempt to perform two or more operations at same time but because of the nature of the device or system the operations must be done in the proper sequence to be done correctly.

When order of execution can change the result now that is nothing, it's a race condition.

* **Semaphores :-** A semaphor is an integer variable int s that a part from utilization is access only through two standard atomic operations - wait and signal.

1. **Wait** - It is an atomic operation. It reduces the value by 1.

2. **Signal** - It is an atomic operation. It increases the value by 1.

PAGE NO:
DATE: / /

P_i	P_1, P_2, P_3	
do {	int s = 1;	
entry section	wait (s)	signal (s)
// critical section	{	{
exit section	while (s <= 0);	s = s + 1;
// remainder section	s = s - 1;	}
} while(T)	}	

PAGE NO :
DATE : 8 / Mar / 2019

UNIT – 3
Deadlocks

Imp
* **Deadlock :-** A deadlock is a situation in which some process wait for each other actions indefinitely. It can arise in a synchronisation situation.
Example- When process wait for the other process to send them message or to release resources that they need.

- **Definition of deadlock –** Deadlock means a lock having no keys.
A set of process is in deadlock if each of them wait for an event that can be cause only by process in the set. Thus each process wait for an event that cannot occur. This situation arises if the following conditions are satisfied –

1- Process P_i in a set of processes D is blocked on some event E_j.
2- Event E_j can be cause only by actions of other process in D.
Each process P_i in D must satisfy above two conditions for D to be in deadlock

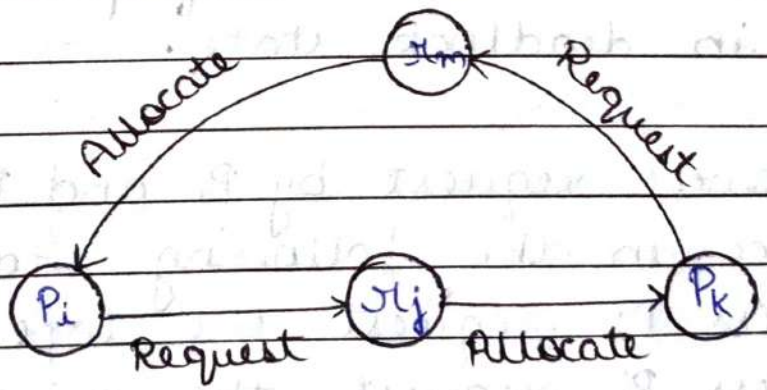
state.

* **Deadlocks in Resource allocation :-** Three events concerned the resource allocation can occur in a system.

1. Request for a resource.
2. Allocation for a resource.
3. Release of a resource.

A request event occurs when process P_i makes a request for a resource r_s. If r_s is currently allocated to some process P_K, process P_i gets locked on an allocation event for r_s. In effect P_i is waiting for P_K to release r_s.

A release event by P_K free resource r_s. It can cause the allocation event for which P_i is waiting. Process P_i will face an indefinite wait if P_K is release of r_s is indefinitely delayed.



Deadlock

There may be another resource r_m which is allocated to the process P_i and process P_k request for it to accomplish its task then same will occur that is until P_i release the resource r_m. P_k can't complete its task so both process P_i and P_k are locked. No (process) task is perform in the system.

xample- A system contains one tape and one printer and two process P_i and P_j that use these resource as follow -

Process i	Process j
Request tape	Request Printer
Request printer	Request tape
Use tape and printer	Use tape and printer
Release tape	Release Printer
Release Printer	Release tape

Show that the set of process P_i, P_j is in deadlock state.

tion:- Resource request by P_i and P_j take place in the following order -

1- Process P_i request the tape.
2- Process P_j request the printer.
3- Process P_i request the printer

PAGE NO :
DATE : / /

4- Process Pj request the tape.
The first two requests are granted immediately because a tape and printer exist in the printer. Now process Pi hold the tape and Pj hold the printer. When Pi ask for the printer it is blocked until Pj release the printer.
Similarly Pj is locked until Pi release the tape. So the set of process Pi, Pj is in deadlock state.

* Necessary conditions for a resource deadlock –

1. Mutual exclusion (Non-sharable resources) – Resources cannot be shared a process need exclusive access to a resource. If another process request that resource, it is block until the resource is release by the first process.

2. Hold and wait – A process continue to hold the resources allocated to it while waiting for other resources.

3. No preemption – OS preempt a resource from one process in order to allocate it to another process. A resource can be

release after that process completed its task.

4. Circular wait – A circular chain of hold and wait conditions exist in the system. A set of waiting process $P_0, P_1, \ldots P_n$ must exist in the system such that P_0 is waiting for the resource held by P_1, P_1 is waiting for the resource held by P_2 and so on. P_{n-1} is waiting for the resource held by P_n and P_n is waiting for the resource held by P_0.

* **Handling deadlocks :-**
There are 3 fundamental approach use in deadlock handling. Each approach has different implications for user process and for the operating system.

1- Deadlock Prevention – The kernal uses a resource allocation policy that ensure that the four conditions for resource deadlock do not arise simultaneously. This approach makes deadlock impossible.

2- Deadlock Avoidance – The kernal analyz

allocation state to determine whether granting a resource request might lead to be a deadlock later. Only request that cannot lead to a deadlock are granted, others are kept pending until they can be granted.

3- Deadlock detection and resolution – The kernal analyze the resource state to check whether a deadlock exist, if so it abort some process and allocates the resources held by them to other process. So that the deadlock classes to exist.

* **Deadlock Avoidance :-**

Given a prior information about the maximum number of resources of each type that may be requested for each process. It is possible to construct an algorithm that ensure that the system will never enter a deadlock state. This approach is called as deadlock avoidance.

- Safe state – A state is safe if the system can allocate resources to each process in some order and avoid deadlock.

Formally a system is said to be in safe state only if there exist a safe sequence.

PAGE NO :
DATE : 11 / Mar / 2019

- Unsafe state – If no such sequence exist then the system state is said to be unsafe.
A safe state is not a deadlock state conversely, a deadlock state is an unsafe state.

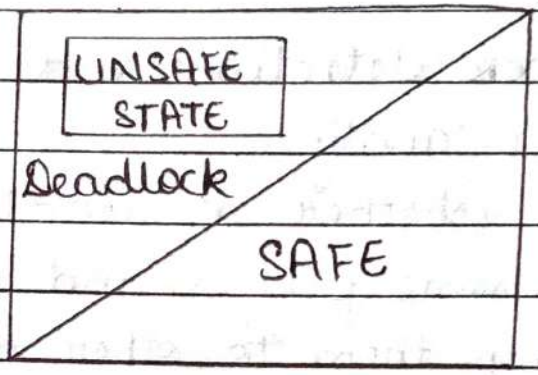


Fig:- Safe, Unsafe, Deadlock State

Example- Consider a system with 12 tape driver and 3 process as-

Process	Maximum needs	Current need	Available
P_0	10	5	12
P_1	4	2	
P_2	9	3	

Out of 12 available tape drives suppose at time t_0 process P_0 is holding 5 tapes, P_1 hold 2 tapes, P_2 hold 2 tapes and 3 tape drives are free. Whether this system is in deadlock state or not.

PAGE NO :
DATE : 13 / Mar / 2019

★ **Banker's algorithm :-** The resource allocation graph algorithm is not applicable to a resource allocation system with multiple instance of each resource type, the deadlock avoidance scheme that we describe next is applicable to such a system. But is less efficient than resource allocation graph scheme. This algorithm is commonly known as Banker's algorithm.

Several data structures are required to implement the Banker algorithm. Let n be the number of process in the system and m be the number of resource type. We need the following data structures —

1. **Available** — A vector of length M indicates the number of available resources of each type. If available[j] = k instance of resource type r_j are available. It is a one-dimensional array of length m.

2. **Max** — It is a 2-dimensional array (matrix) of length n x m that defines the maximum demand of each process in a system. If max[i,j] = k then

process P_i may request at most k instance of resource type r_j.

3- Allocation — It is also a 2-D array of length n X m that defines the number of resources of each type currently allocated to each process.
If allocation [i,j] = k then process P_i currently allocate k instance of resource type r_j.

4. Need — If need [i,j] = k then process P_i may need k more instance of resource type r_j to complete its task.
Need [i,j] = Max[i,j] − Allocation [i,j]

* Safety Algorithm :-
The algorithm for finding out whether or not a system is in a safe state can be describe as follow-

1. Let work n finish by vector m and n.
Initialize work = available
finish [i] = false;

2. Finish an i such that both -
(i) finish [i] = false
(ii) Need ≤ work
if no such i exist, go to step 4.

PAGE NO :
DATE : / /

3. Work = Work + Allocation
finish [i] = true
goto step 2

4. If finish [i] = true for all i, then the system is in safe state.

★ **Resource request algorithm** —
Let request$_i$ be the request vector for process P_i. If request$_i$[j] = k then process P_i wants k instance of resource type R_j. When a request for resources is made by process P_i, the following actions are taken :

(1) If Request$_i$ ≤ Need$_i$
goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

(2) If Request$_i$ ≤ Available
goto step (3) ; otherwise, P_i must wait, since the resources are not available.

(3) Have the system pretend to have

allocated the requested resources to process P_i by modifying the state as follows:

Available = Available − Request$_i$

Allocation$_i$ = Allocation$_i$ + Request$_i$

Need$_i$ = Need$_i$ − Request$_i$

If the resulting resource − allocation state is safe, the transaction is completed and process P_i is allocated its resources. However, if the new state is unsafe, then P_i must wait for Request$_i$ and the old resource − allocation state is restored.

Example:- Consider a system with 5 process - P_0, P_1, P_2, P_3, P_4 and 3 resources types A, B, C. Resource type A has 10 instance, B has 5 instance and type C has 7 instance. Suppose at time t_0 following snapshot of the system has been taken −

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

PAGE NO :

DATE : / /

(i) What will be the content of need matrix?

(ii) Is the system in safe state? If yes, what is the safe sequence.

(iii) If ~~request~~ (3,3,0)

(iii) What will happen if process P_1 request 1 additional instance of resource type A and 2 instance of resource type C.

(iv) If request (3,3,0) by process P_4 arrive in the state define by (iii), can it be granted immediately?

(v) If a request (0,2,0) by process P_0 arrive then check whether it is granted or not?

Solution:- (i) Content of need matrix -

Need = Max - Allocation

$$= \begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

(ii) Applying safety algorithm on the given system for P_i if Need $\leq$ Available then P_i is in safe sequence.

PAGE NO:
DATE: 19/ Mar/ 2019

Available = Available + Allocation

So for P_0

($i=0$) $Need_0 \rightarrow 7,4,3$

Available $\rightarrow 3,3,2$

Condition is false.

So P_0 must wait.

for P_1

($i=1$) $Need_1 \rightarrow 1,2,2$

Available $\rightarrow 3,3,2$

(Need < Available)

P_1 will be kept in safe sequence.

Now available will be updated as -

Available = Available + Allocation

$= 3,3,2 + 2,0,0$

$= 5,3,2$

for P_2

($i=2$) $Need_2 \rightarrow 6,0,0$

Available $\rightarrow 5,3,2$

Condition is false and P_2 must wait.

for P_3

($i=3$) $Need_3 \rightarrow 0,1,1$

Available $\rightarrow 5,3,2$

Now, Available = $5,3,2 + 2,1,1$

$= 7,4,3$

for P_4

($i=4$) $Need_4 \rightarrow 4,3,1$

Available $\rightarrow 7,4,3$

Need < Available

PAGE NO :
DATE : / /

Now Available = 7,4,3 + 0,0,2
= 7,4,5

Now we have two process P_0 and P_2 in waiting state. As current available either P_0 or P_2 is kept in safe sequence

Firstly we take P_2 whose need = 6,0,0
Available → 7,4,5
$Need_2$ < Available.
So P_2 now comes in safe state.
Available = 7,4,5 + 3,0,2
= 10,4,7
Next, P_0 whose Need → 7,4,3 and
Available → 10,4,7
P_0 now comes in safe state
and Available = 10,4,7 + 0,1,0
= 10,5,7
So the safe sequence is ←
$< P_1, P_3, P_4, P_2, P_0 >$

(iii) Since P_1 request some additional instance of resources such that
$Request_1 (1,0,2)$
To decide whether this request is immediately granted as we first check that Request ≤ Available which hold true. So the request may be granted

PAGE NO :
DATE : 22 / Mar / 2019

To confirm that this request is granted we check the new state by applying safety algorithm that our system is in safe state or not.

If the new state is in safe state then only this request is granted otherwise not.

To define the new state of the system because of the arrival of request P_1. We follow the resource request algorithm which result as-

Process	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

We must determine whether this new system state is safe. We again execute a safety algorithm and find the safe sequence as $<P_1, P_3, P_4, P_2, P_0>$ which satisfy our safety requirements hence we can immediately grant the request for process P_1.

(iv) The request for (3 3 0) by P_4 cannot be granted because Request$_4$ = 3 3 0

Available = 2 3 0

PAGE NO :
DATE : / /

In this situation the condition is false. So it is not granted since resource are not available.

(v) The request for (0 2 0) by process P_0.

Request$_0$ = 0 2 0

Available = 2 3 0

So the condition is true and the request may be granted.

If it is granted then the new state of the system is define as-

Available = Available − Request

= 2,3,0 − 0,2,0

= 2,1,0

Allocation = Allocation + Request

= 0,1,0 + 0,2,0

= 0,3,0

Need = Need − Request

= 7,4,3 − 0,2,0

= 7,2,3

Process	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 3 0	7 2 3	2 1 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

PAGE NO :

DATE : 22/ Mar/ 2019

Ques Consider a system with 5 process P_0 ---- P_4 and 3 resources types A, B and C. Resource type A has 7 instance, B has 2 and C has 6 instance. Suppose at time t_0 we have following state -

	Allocation	Request	Available
Process	A B C	A B C	A B C
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

(i) Is the given system is in deadlock state?

(ii) Suppose P_2 makes an additional request (0 0 1). What will be the effect of this request to the system?

Sol:- (i) For P_0 $\Rightarrow$ Request$_0$ = 0 0 0

Available = 0 0 0

Condition is true

Now available = Available + Allocation

= 0 0 0 + 0 1 0

= 0 1 0

for P_1 $\Rightarrow$ Request$_1$ = 2 0 2

Available = 0 1 0

(Request > Available)

Condition is false, so P_1 must wait.

PAGE NO:
DATE: / /

for P_2 ⇒ $Request_2 = 000$
$Available = 010$
Condition is true.
Now available = 010 + 303
= 313
for P_3 ⇒ $Request_3 = 100$
$Available = 313$
Condition is true.
Now available = 313 + 211
= 524
for P_4 ⇒ $Request_4 = 002$
$Available = 524$
Condition is true
Now available = 524 + 002
= 526
Now for the waiting process P_1 ⇒
$Request_1 = 202$
$Available = 526$
Condition is true.
So P_1 comes in safe state.
Now available = 526 + 200
= 726.

(ii) Since P_2 request some additional instance of resources such that
$Request_2 = 001$
$Available = 726$
$Request < Available$

PAGE NO :
DATE : / /

So the request may be granted.
The new state of the system will be –

	Allocation	Request	Available
Process	A B C	A B C	A B C
P_0	0 1 0	0 0 0	7 2 5
P_1	2 0 0	2 0 2	
P_2	3 0 4	0 0 1	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

PAGE NO :

DATE : 23 / Mar / 2019

* **Deadlock resolution or recovery from deadlock :-**

When a detection algorithm determines that a deadlock exist, several alternatives approach can be implemented. One possibility is to inform the operator that a deadlock has occure and to let the operator deal with the deadlock manually. The other possibility is to let the system to recoverer from the deadlock automatically. There are two options for breaking a deadlock -

1- Terminate some process P_j which is a subset of D to free the resource required by P_i.

2- Add new unit of resources requested by P_i or preempt some resources from one or more of the deadlock process.

PAGE NO:
DATE: 27 / Mar/2019

UNIT – 4
Device Management

★ **Disk Scheduling :-**

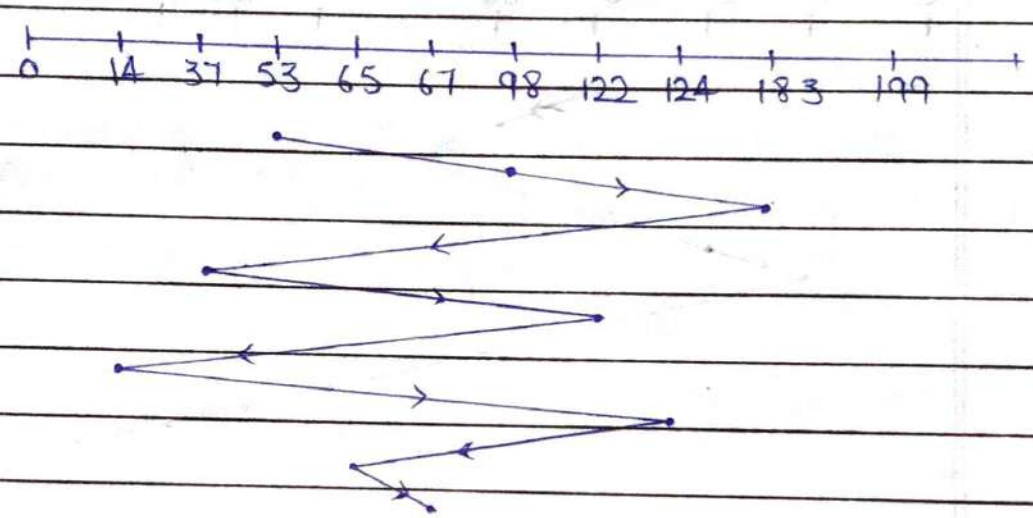
The major responsibility of the operating system is to use the hardware in an efficient manner. For the disk - drivers meeting this responsibility having a fast access time and disk bandwidth. We can improve both access time and bandwidth by scheduling the servicing of the disk input - output request in a good manner.

1. **First Come First Serve (FCFS) scheduling:**
FCFS is the simplest algorithm defined for disk scheduling. Program for this algorithm is easy to implement but it does not provide the fastest service for this consider following example –

Ques- Consider an order disk queue with request involving tracks 98, 183, 37, 122, 14, 124, 65 and 67. If the read / write head initially at track 53. What is the total distance that the disk move to satisfy all the pending request for FCFS.

PAGE NO :
DATE : / /

Sol:- Since the head is initially at track 53. We see that disk move from 53 to 98, then to 183, 37, 122, 14, 124, 65, 67.
Total head movement = 640 tracks
$|98-53| + |183-98| + |37-183| + |122-37| + |14-122| + |124-14| + |65-124| + |67-65|$
$= 45 + 85 + 146 + 85 + 108 + 110 + 59 + 2$
$= 640$ tracks



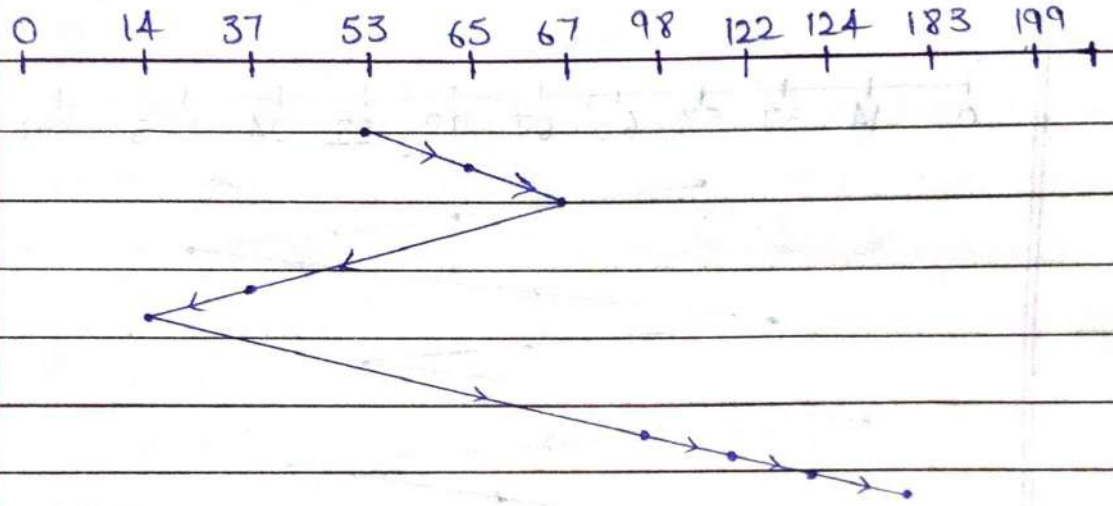
Queue :- 98, 183, 37, 122, 14, 124, 65, 67
Initial position of head - 53

fig :- FCFS Scheduling

2- **Shortest-Seek-Time-First (SSTF) Scheduling:**
It is reasonable to service all the request close to the current head position before moving the head far away to service other request.
In our example the closest request to

PAGE NO :
DATE : / /

the initial head position (53) is at track 65. Once we at 65 then next closest track is 67. Continuing in this manner as shown in figure below we finally reach at track 183. This algorithm result in total head movement of 236 track less than obtain in FCFS.



Queue = 98, 183, 37, 122, 14, 124, 65, 67
Initial position of head = 53

fig :- SSTF Scheduling

3 Scan - Scheduling :-

In this algorithm the disk arm starts at one end and move towards the other end servicing request as it reaches each track until it gets to other end of the disk. At the other end the direction of the head is reverse

and servicing continuous.
We can explain the scan algorithm as -
Before applying scan to schedule request on tracks 98, 183, 37, 122, 14, 124, 65 and 67.
The position of head is at 53.
The scan algorithm sometime called the elevator algorithm as the disk arm behave like an elevator in the building, first servicing all the request going up and then reversing to service request the other way.

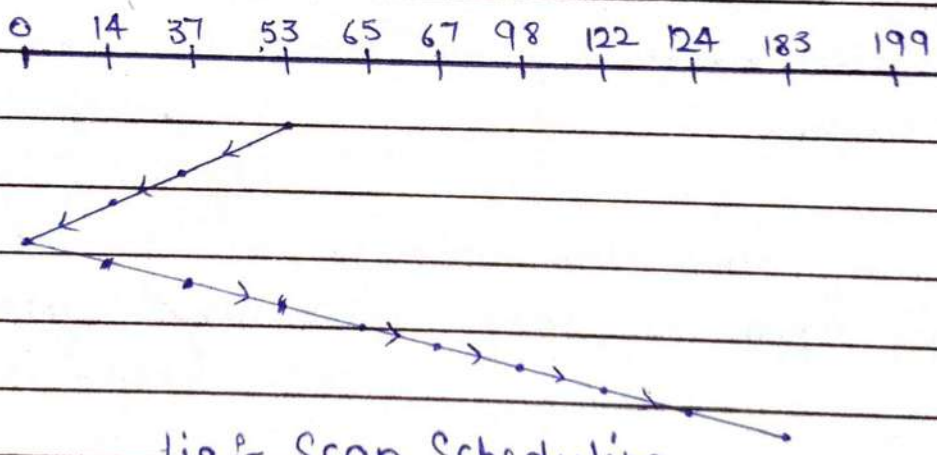


fig :- Scan Scheduling

4- **Circular Scan (C-Scan) Scheduling :-**
Circular scan scheduling is a variant of scan scheduling that provide more uniform waiting time like scan scheduling, circular scan move the head from one end of the disk to the other servicing

PAGE NO :

DATE : / /

request along the way.

Head Position = Large value

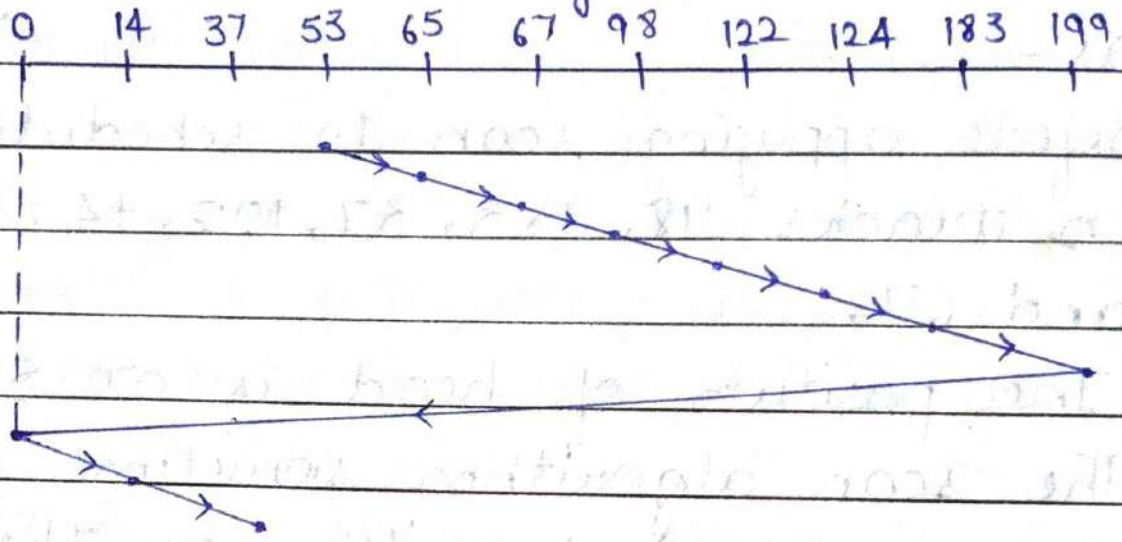


fig:- C-Scan Scheduling

PAGE NO.:
DATE : 12 / Apr / 2019

* **File-System Management** :- A file is a collection of related information, commonly files represent program and data. Data files may be numeric, alphabetic or alpha-numeric. In generated a file is a sequence of bits, bytes, lines or records.

A file is name and is referred by its name. It has certain other properties such as its type, the time of its creation.

Attributes of a file –

1. Name
2. Identifier
3. Type
4. Location
5. Size
6. Protection
7. Time, date and user identification

1. Name – The symbolic file name is the only information that is kept in human readable form.

2. Identifier – This is a unique tag that is represented by the number which identify the file within the file system.

3. Type – This is the information about the file which is required for the system having different type of files.

PAGE NO:
DATE: / /

4- Location – This information is a pointer to a device and to the location of the file on that device.

5- Size – The current size of file (in bytes, word or block) and possibly the maximum allow size are included in this attribute.

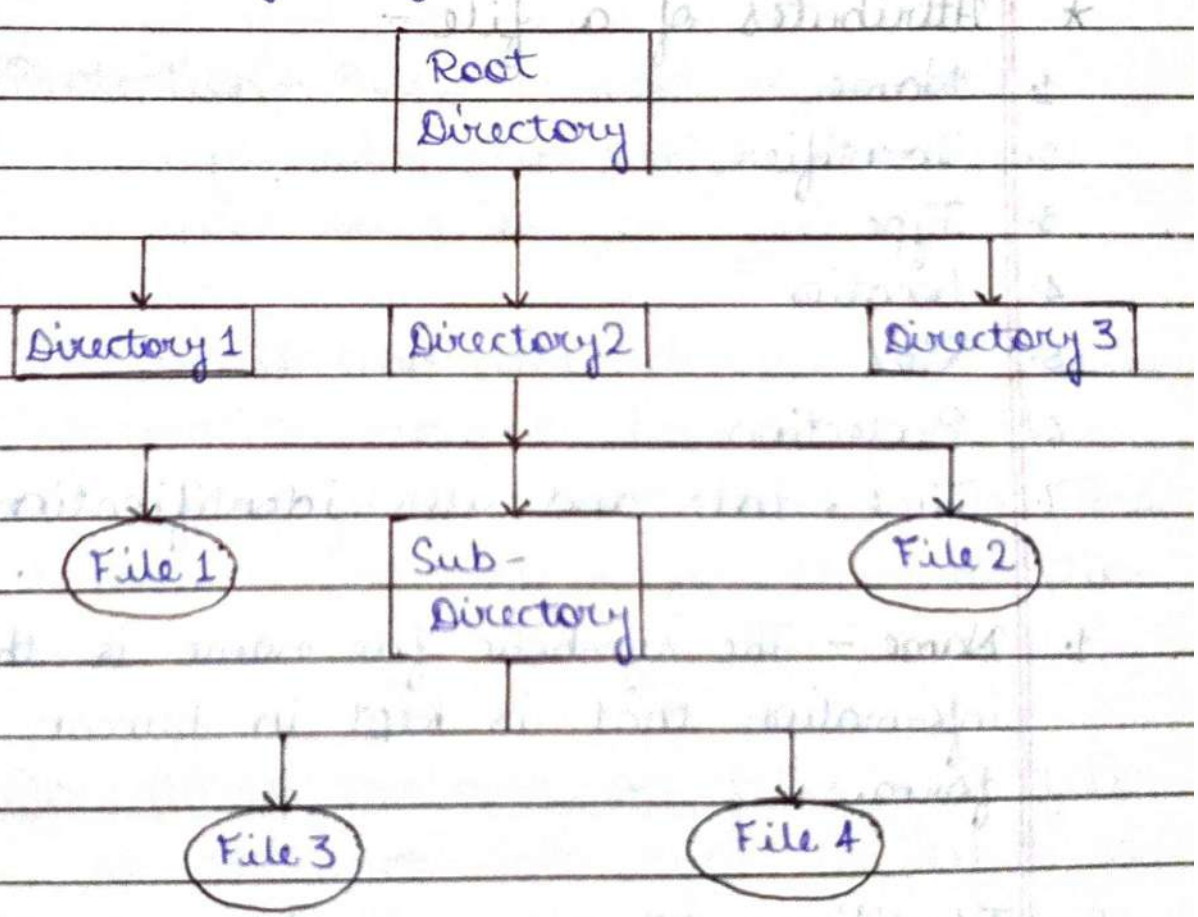
6- Protection – Access control information determine who can perform various operations on file.

7- Time, date and user identification – This information may be kept for creation, last modification and last uses. These data can be useful for the protection and security.

PAGE NO :
DATE : / /

* Structure of file system management:
File management is one of the basic and important features of operating system. Operating system is used to manage files of computer system. All the files with different extension are managed by operating system.

The below figure shows the general hierarchy of the storage in an operating system :



PAGE NO :
DATE : 12 / Apr / 2019

* File-System Management :-

A file is a collection of related information. Commonly files represent program and data. Data files may be numeric, alphabetic or alphanumeric. In generated a file is a sequence of bits, bytes, lines or records.

A file is named and is referred by its name. It has certain other properties such as its type, the time of its creation

* Attributes of a file -

1. Name
2. Identifier
3. Type
4. Location
5. Size
6. Protection
7. Time, date and user identification

1. **Name** - The symbolic file name is the only information that is kept in human readable form.

2. **Identifier** - This is a unique tag that is represented by a number which identify the file within the file system.

3. **Type** – Type is the information about the file which is required for the system having different type of files.

4. **Location** – This information is a pointer to a device and to the location of the file on that device.

5. **Size** – The current size of file (in bytes, word or block) and possibly the maximum allow size are included in this attribute.

6. **Protection** – Access control information determine who can perform various operations on file.

7. **Time, date and user identification** – This information may be kept for creation, last modification and last uses. These data can be useful for the protection and security.

* **Operations perform on files** :- A file is an abstract data type, to define a file property we need to define various operations that can be perform on files. The OS provide various system calls to create, write, read, delete files.

PAGE NO :
DATE : 15 / Apr / 2019

* **Single-level directory** – It is the simplest directory structure, All files are contain in the same directory which is easy to support and understand.

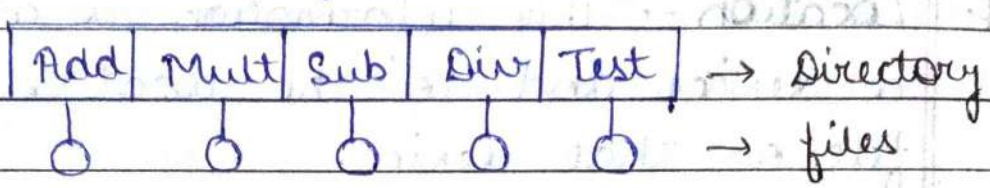


fig :- Single-level directory

A single level directory has certain limitations when the number of files increases or when the system has more than one user.
If two users call their data file then the unique name rule is violated.

* **Two-level directory** – The disadvantage of single-level directory is a confusion of file names between different users. The solution for this problem is to create a directory for each-other.

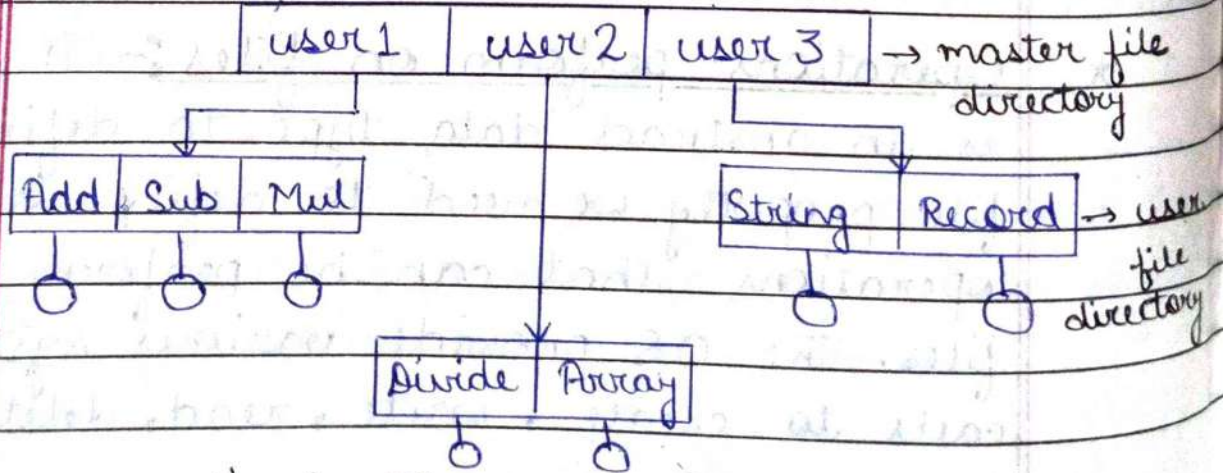


fig :- Two-level directory

In two-level directory structure each user has his own user file directory. Each user has similar structure but least only the files of a single user. When user login the system master file directory is search.

* **Access control verification :-**

Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the business or organisation.

Access control system perform identification authentication and authorization of users and entities by evaluating required login credentials that can include passwords, PINs, biometric scans.

These security control work by identifying an individual or entity, verifying that the person or application is who or what it claims to be, and authorizing the access level and set of actions associated with the username or IP address.

PAGE NO :
DATE : / /

- **Physical file system** – Physical files contain the actual data that is stored on the system and a description of how data is to be presented to or received from a program. They contain only one record format, and one or more members.
A physical file can have a keyed sequence access path. This means that data is presented to a program in a sequence based on one or more key fields in the file.

- **Logical file system** – Logical files do not contain data. They contain a description of records found in one or more physical files. A logical file is a view or representation of one or more physical files. Logical files that contain more than one format are referred to as multi-format logical files.