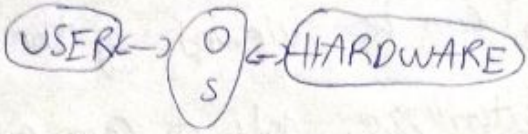
UNIT-I

27/10/21

Introduction to operating Systems

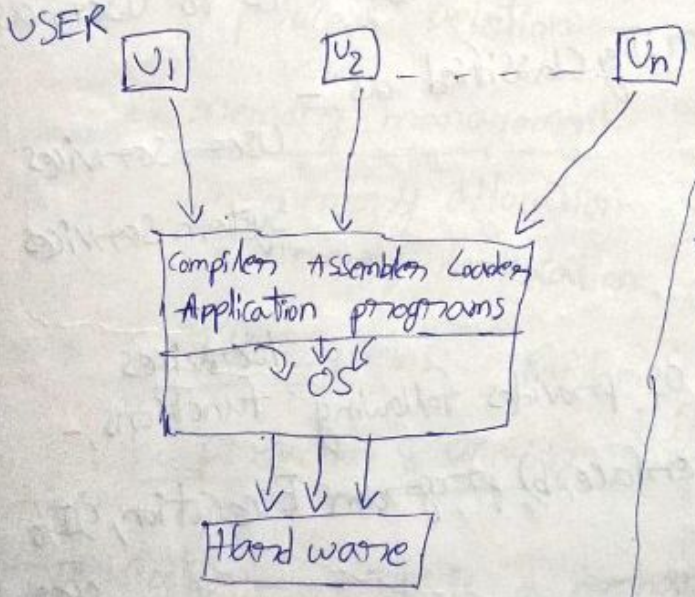
An (operating system) (OS) is defined as a Communication Bridge between users and Computers (Hardware).



=> Components of a Computer System:-

There are 4 components available in a Computer System. They are:-

1. USER
2. Hardware
3. OS
4. Application programs.



Compiler -> Instructions to machine-code

Assembler -> Low-Level Symbolic code to machine-code

Loader -> For loading programs & place program in memory for Execution.

Application programs -> E-mail, web browsers, Games, word processors Etc.

28/10/21

Operating System Role:-

Role of O.S can be viewed in 2 ways:-

i) User view, ii) System view

User view:-

User expects that once a task is given, he will expect only output, he never bothers how it produced.

System view:-

OS can be viewed by the system as a "Resource allocator", i.e., only OS can allocate resources like, memory, I/O devices etc to a process for it's Execution.

OS Services:-

OS provides lot of services to users as well as system. They classified as:-

User services
System services

User services:-

OS provides following functions:-

a) Interface, b) program Execution, c) I/O operation
d) File system manipulation, e) communication,
f) Error detection

System Services:-

OS provides following services:-

a) Resource allocator

b) Accounting
c) protecti

29/10/21
=)

OS Functi

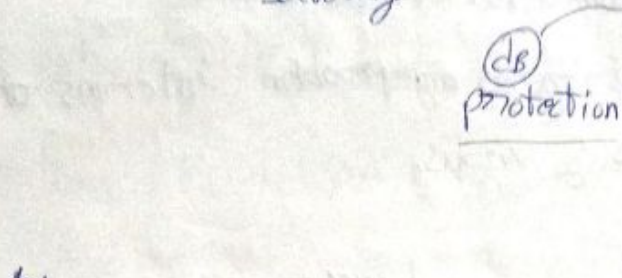
1.

2.

Scanned By Camera Scanner

b) Accounting

c) protection & Security



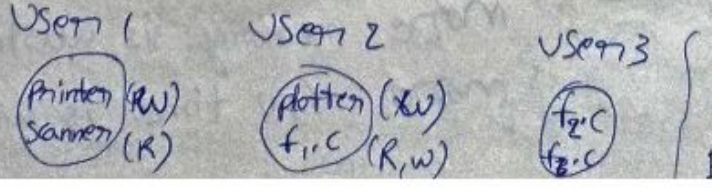
29/10/21

=)

OS Functions :-

1. process management :-
   - process creation :-
   - process Scheduling :-
   - process Execution
   - process Deletion
2. Memory management :-
   - memory allocation
   - memory deallocation
3. File Systems :- Random & Sequential
4. protection & security :- File permissions

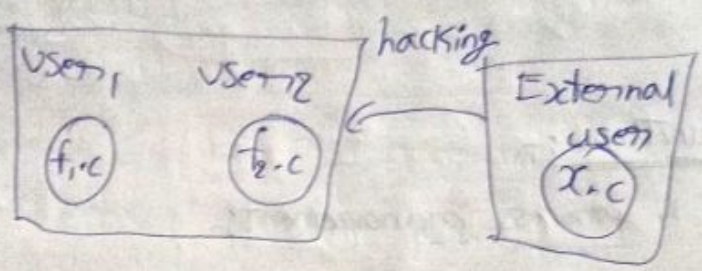
- protection :- It is a mechanism which safeguards the data & resources (h/w & s/w resources) internally i.e. from other users with in the LAN.



· Security:-

Security is safeguarding the data and resources from external users.

The security faces big problem interms of worms & virus.

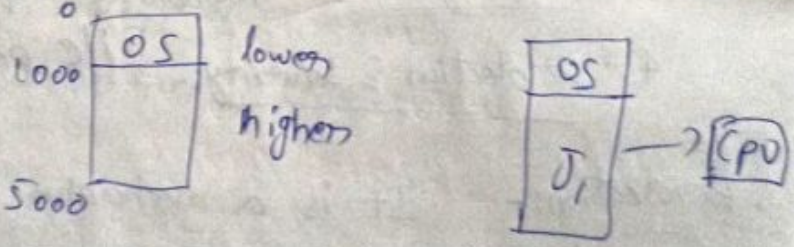


1/11/21

Special purpose (operating) Systems:-

1. Uni-programming:- Centralized / Distributed

According to uniprogramming concept, memory is divided into only one part i.e. maximum one user is accomodated.



Adv:-

more memory is available.

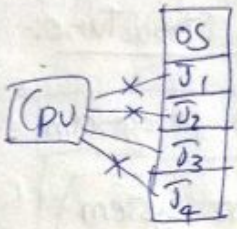
Dis Adv:-

More memory is wasted.

more CPU time is wasted.

2. multi-programming:-

multiple users are allowed by dividing the memory into multiple partions. Each part is allowed to one user/Job.



Advantages:-

memory isn't wasted

CPU time isn't wasted

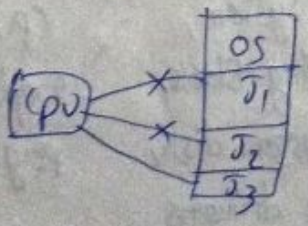
Dis advantages:-

memory is fragmented.

3. Time sharing system:-

Logical extension of multiprogramming concept.

Each Job is fixed with a particular time, within that time, Job must Execute.



Completion Time

J_1 - 10sec

J_2 - 7sec

J_3 - 12sec

2/11/21

=>

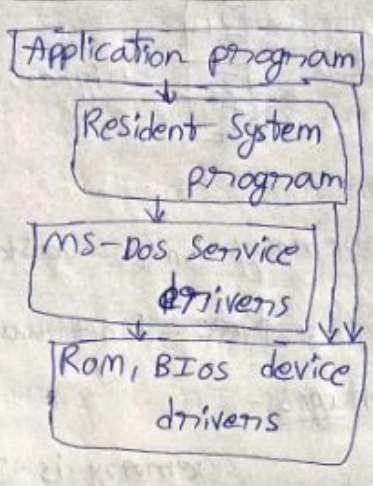
OS Structures

There are no. of structures designed for OS. Two popular structured designs are:-

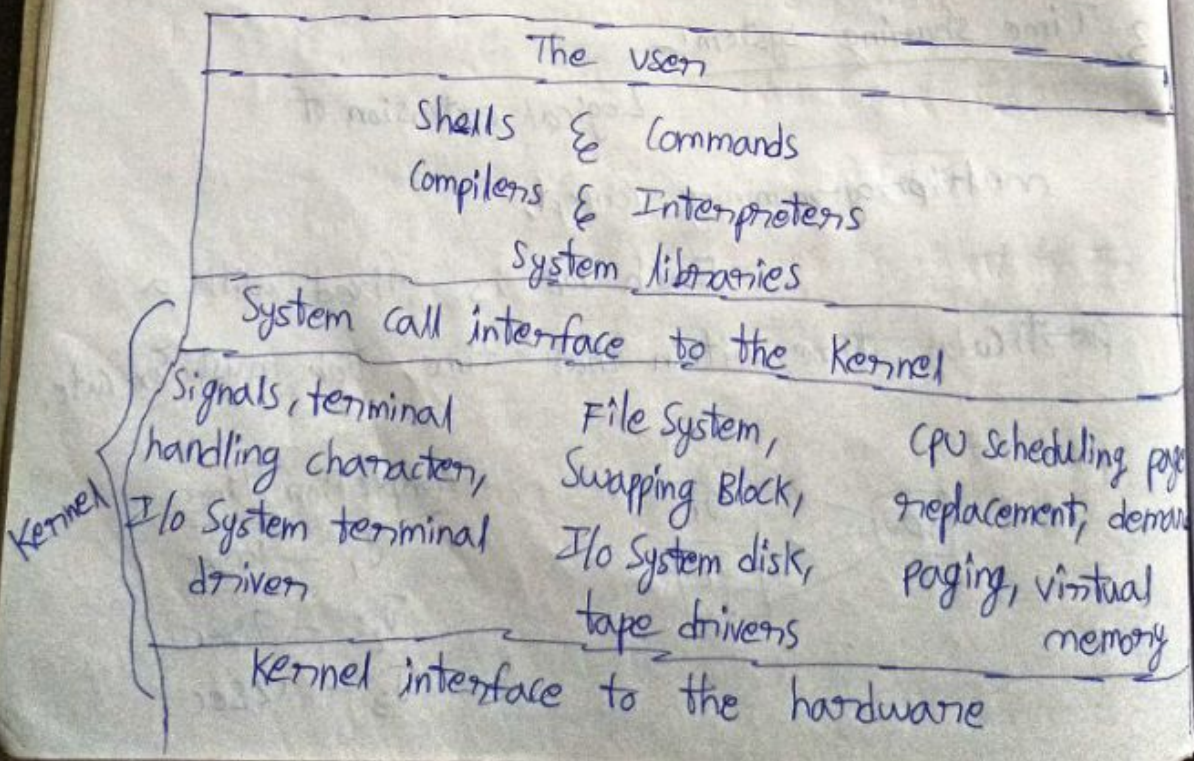
1. Simple structure
2. Layered structure

1. Simple Structure:-

i) MS-DOS Structure:-



ii) UNIX Structure:-



terminal Contr
termina

2. Layered Struc

a) The
of

b) In
int

c) The

d) Th

=> System
De

S

terminal Controllers terminals	device Controllers disks & tapes	memory Controller physical memory

2. Layered Structure:-

a) The Advantage of Layered approach is Simplicity of Construction & debugging.

b) In this Layered approach, the os is broken into no. of Layers (levels).

c) The bottom layer is hardware.

d) The uppermost layer is user interface.

=) System calls:-

Def:- A request to the os by a process (or) user is called System calls.

There are different types of System calls Some of them are as follows:

1. process management:-

a) end :

The program Execution is made to end the Execution.

b) abort :

when a program is on Execution the program is abnormally made to end.

c) Load : Loading the program to memory.

d) Execute : The program will be Executed.

e) Create process : A process will be Created.

f) Terminate process : The process will be deleted when an interrupt occurs (or) a resource is not available.

g) Get process attributes : Viewing attributes like name, length, type of the process.

h) Set process attributes : we have to set the attributes for the program.

i) Signal (event) : one process gives a Signal to the waiting process to get ready for Execution

j) wait (event) : The process has to wait when another process is on Execution.

2. memory management :-

a) Allocate memory :- Allocation of memory for a Created process

b) Deallocate (free) memory:-

If a process is identified as unwanted, then the process is deleted and the memory is reclaimed.

3. File management:-

a) Create file:- Creating a file
b) Delete file:- Deleting a file
c) open file:- opening the contents of a file
d) close file:- closing a file
e) Read, write, reposition of file; Reading a file, writing into a file & the position is regained.
f) Get file attributes: viewing file attributes like file name, file type, file size etc.

4. Device management:-

a) Request device:- A command which asks OS to allocate a device.
b) Release device:- After the usage of the device the user can release the device by informing the os.
c) Read, write, reposition:-

d) get device attributes:- viewing the device attributes like status, idle, allocated etc.

e) Set device attributes;- we have to set the attributes for the device.

f) Attach device;- Allocating a device by OS to the user.

g) Detach device;- Deallocating the device from the user by OS.

5. Communications;-

a) Create Connection;- Establishing connection for between 2 two processes.

b) Delete Connection;- Disconnecting the path which is already Exists.

c) Send message
Sending a message

d) Receive message
Receiving a message

e) Attach Device;- Allocating a device to the user.

f) Detach Device;- Deallocating the device from the user.

=) Examples

b. Process

2. File

3. In

4. G

=) 8

=> Examples of windows and UNIX system calls:-

	Windows	UNIX
1. Process Control:-	Create process()	→ fork()
	Exit process()	→ exit()
	wait	→ wait()
2. File manipulation:-	Create file()	→ open()
	read file()	→ read()
	write file()	→ write()
3. Information maintainence:-	Get current process ID()	→ getpid()
	set timer()	→ alarm()
	Sleep()	→ sleep()
4. Communication:-	Create pipe()	→ pipe()

=> 8/11/21

Process management:-

Def:-

A ~~Program~~ which is on Execution is called a process.

(or)

A task / Job which is on Execution is called process.

process states:-

A process can be represented in various States during it's lifetime. They are;-

a) new state:- The process is just created.

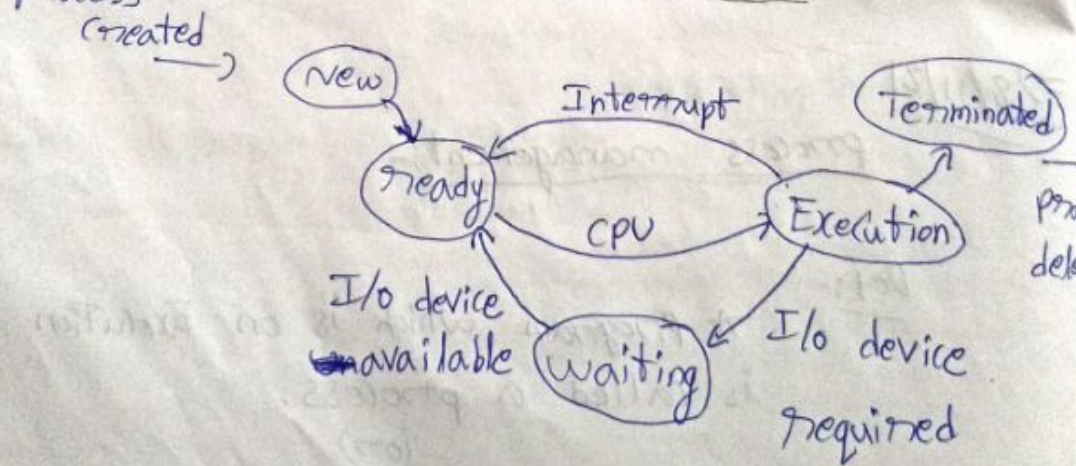
b) Ready State:- The process waiting to be assigned to processor, (CPU).

c) Execution (or) Running State:- The process is being Executed.

d) waiting state:- The process should be waiting for some event of occur.

e) Terminated State:- The process has finished Execution.

=> Different States of a process:-



9/11/21

=>

is sh
Proces
a (n
A pr
one
PCB

=>

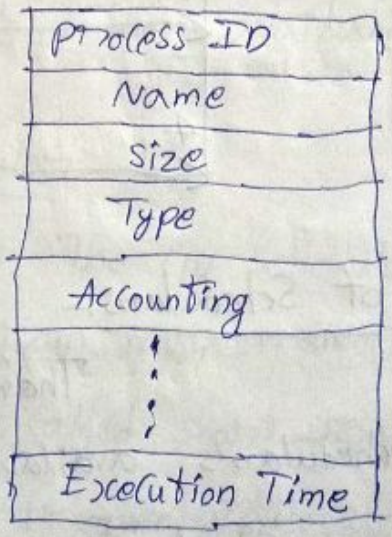
9/11/21

=> Process Control Block (PCB):-

A process control Block is shortly represented as PCB. For Every Process available in the System, There will be a corresponding PCB available in the system. A PCB Contains, all the Informations about one particular process. The process no. & the PCB no. are same.

Process P_0 – PCB_0

A (process) PCB can be represented as follows;



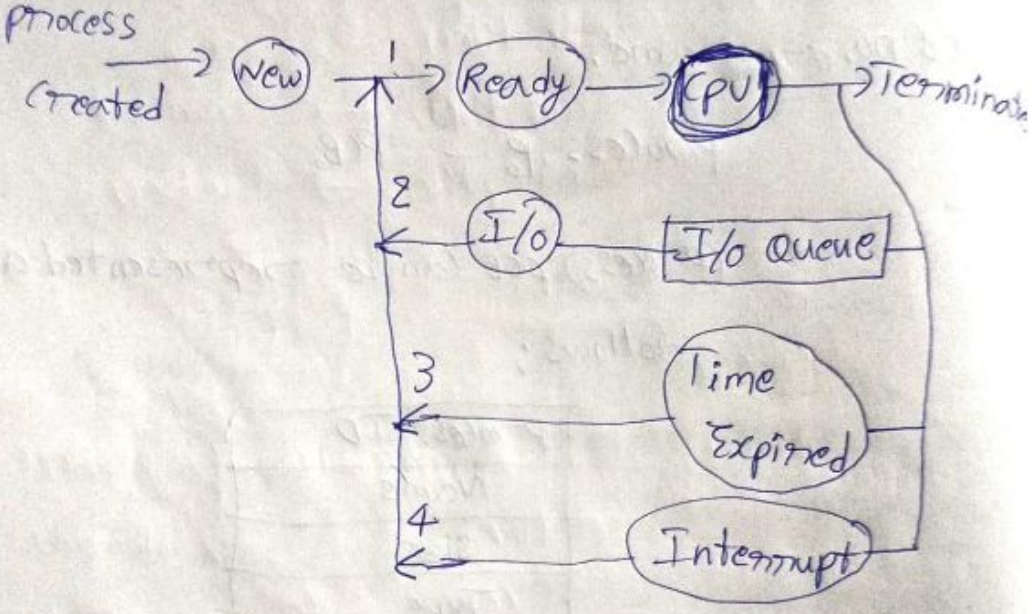
=> Schedular:-

Schedular is a program which Schedule various processes for Execution.

=> Process Scheduling:-

The various processes will be scheduled for Execution by the schedulars program is called "process scheduling."

The process scheduling can be represented as follows:-

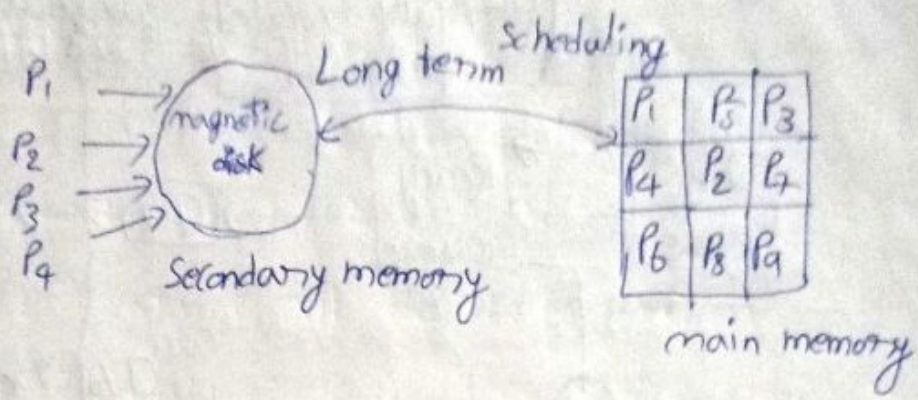


=> Types of Schedulars:-

There are 3 types of schedulars is available. They are;-

a) Long term scheduler (Job schedule
b) Short term scheduler (CPU schedu
c) medium term scheduler

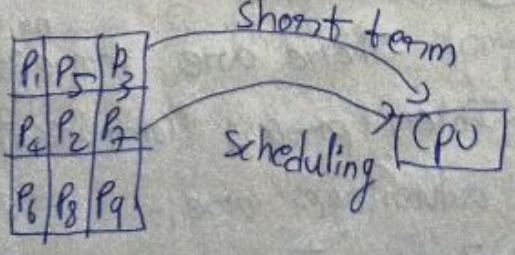
a) Long-term Scheduler:-



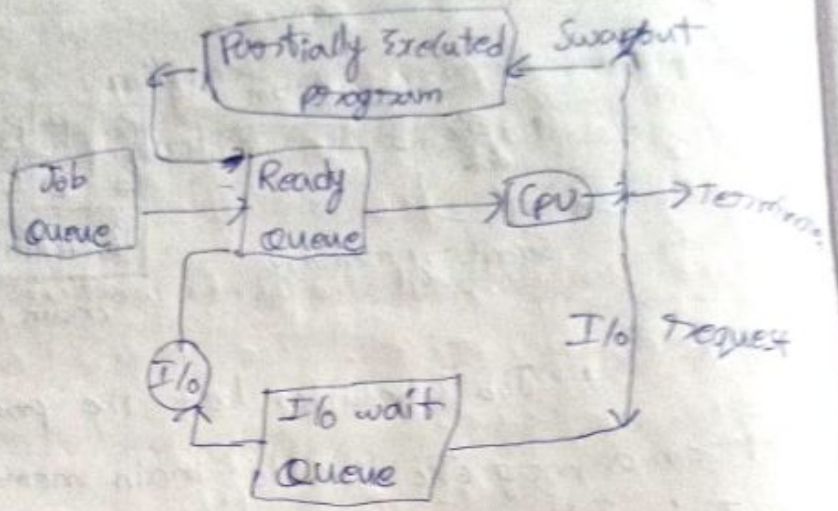
* Job scheduler loads the processes from a magnetic disk to main memory. This Job scheduler is called "long-term scheduler" because it takes more time to load a process from disk to main memory depends on the availability of main memory.

b) Short-term scheduler:-

The CPU scheduler assigns one process from main memory to CPU for Execution. It is called "short-term scheduler" because it very frequently allocates different processes to CPU in minimum time.



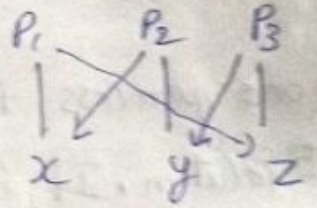
c) medium-term schedulers:-



=> Inter process communication:-

It is defined as sharing of Information among more number of processes.

Ex:-



Info. sharing
computation speed up
modularity
convenience

11/11/21

=> Benefits/Advantages of IPC:-

There are more no. of advantages available in Inter process communication.
Some of the advantages are:-

a) Information sharing
b) Compatation speed up

c) modularity

d) convenience

Information Sharing

a) Info sharing

If a process requires a resource which is available with some process, it can be shared through IPC.

Ex:-

process P Q

Resource printer keyboard

b) Computation Speed up:-

P — Read
write
display

process P Q R

function Read display write

The work of a process can be splitted into no. of modules & these modules can be given to some existing procedures. So that all the procedures parallely execute the work. So that the computation time is reduced.

c) modularity:-

one of the most important use of Ipc is the task to be Executed can be divided into modules.

d) Convenience:-

The process can conveniently Execute the different modules of one task parallely.

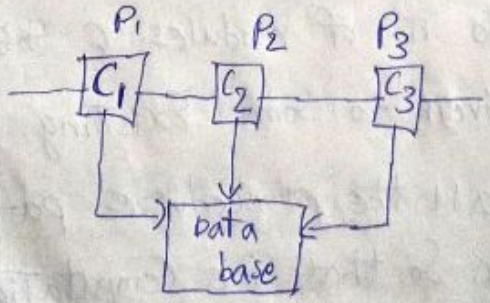
Ex:- P = Read, display, write

Types of Ipc :-

There are 2 types of Ipcs

a) Ipc through shared memory.
b) Ipc through message passing.

a) Ipc through shared memory:-



Ipc can be done through Shared memory concept, i.e., all the computers connected in the network will be sharing a common memory. so, any process which is Executing

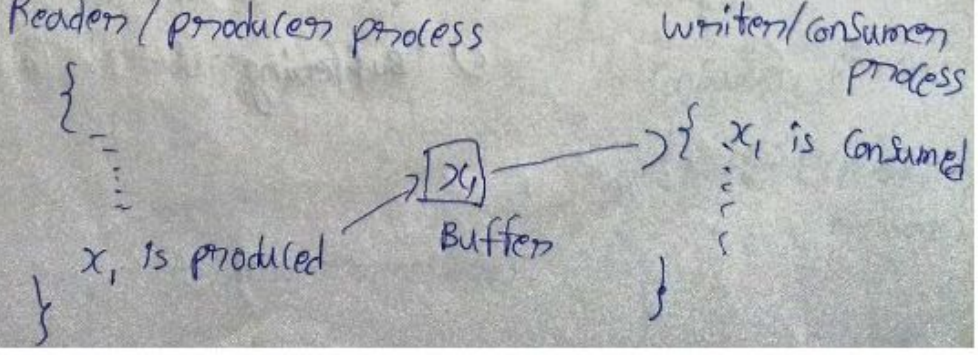
in any computer can access any data from the shared memory.

Ex:-

Producer - Consumer problem
(or)
Reader - writer problem
(or)
Bounded - Buffer problem

Producer - Consumer problem:-

(Producer) It is a process which produces the input for consumer process. The consumer process is also a process, which consumes the producer process output. The producer process is also known as Reader process, & the consumer process is also known as writer process. There is a small memory called Buffer, which stores only one data. The producer process data is stored in ~~that~~ Buffer. If the next data is produced, the previous stored data will be consumed by consumer process.



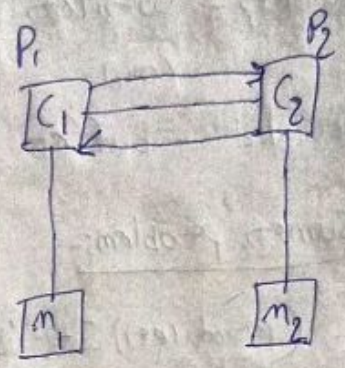
0th sec – x_1 – 1st sec

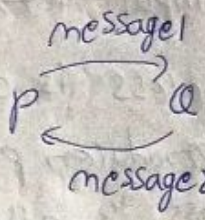
1st sec – x_2 – 2nd sec

0th sec – 1st – idk

1st sec – x_1 – 2nd

b) Ipc through message passing:-





when the computers in the network having Individual memory, sharing, communication can be done through message.

There are 3 different message passing Ipc available

a) Naming

b) Synchronization

c) Buffering

☆

a) Naming IPC:-

According to naming technique, while sending a message, a process P can directly mention the name of process, For example, Q. So that the message will be sent directly to that particular process.

They are:-

a) Direct communication

b) Indirect communication

a) Direct communication:- Two process can be made to communicate directly. There are 2 primitives (or) (or) Commands used for direct communication.

a) Send/SEND:- To send message

b) Receive/RECEIVE:- To Receive message

SEND(Receiver process name, message)

RECEIVE(Sender process name, message)

Ex:- producer - consumer problem.

```
producer process
{
  :
  SEND(consumer, message)
}

consumer process
{
  RECEIVE(producer, message)
  :
}
```

16/11/21

b) Indirect Communication:-

In Indirect Communication, a concept called "mail box" (or) "port", is used to store the messages (data). A group of processes will be sharing the mail box & through this the communication can be done. The same primitives (conditions) used in direct communication also will be used here. "SEND", "RECEIVE" communication also will be used here.

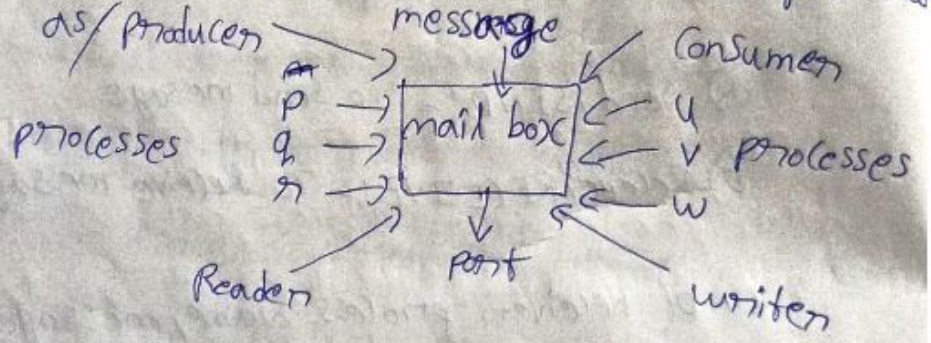
Syntax:-

Send and Receive

SEND(mailbox, msg)

RECEIVE(mailbox, msg)

The Indirect Communication can be represented



Ex:- Reader- writer problem

```
Reader process
{
  -
  -
  -
SEND(mailbox, msg)
}
```

```
writer process
{
RECEIVE(mailbox,m
  -
  -
  -
  -
}
```

=) CPU Scheduling Criterians:-

There are some Criterians, which will be useful to Identify the Best CPU Scheduling Algorithms, for process Execution. These Criterians are said to be "CPU Scheduling Criterians".

The following Criterian's Satisfy the Best CPU scheduling Algorithm's nature.

a) CPU utilization, b) Throughput
c) waiting Time, d) Response Time,
e) Turn around Time

a) CPU scheduling utilization:-

The CPU want to be utilized upto maximum time & it can't be idle to perform the process fastly.

b) Throughput:-

The number of processes which are Completed/Executed in a given time. It want to be maximum.

c) waiting Time:-

To Execute the other processes the waiting time must be less.

d) Response Time:-

The time which is taken to respond to a request for any service, it must be less.

e) Turn Around Time:-

It is defined as,

Turn Around Time = waiting Time + Execution Time

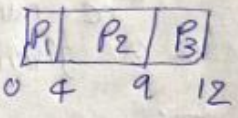
17/11/21 => Gantt chart:-

A Gantt chart is a pictorial representation which shows the different purposes, their Execution time, waiting time & turn around time.

Ex:-

Process	Execution Time
P_1	4s
P_2	5s
P_3	3s

Gantt chart



=> Preemptive Algorithm:-

Preemption means Suspension/ termination. when a process is an Execution by CPU, in middle of the Execution, the process is Suspended. An Algorithm which follows this logic is called preemptive Algorithm.

=> Non-P
when
the P
Proces
which
Algor
=> CPU
(F

=) Non-preemptive Algorithm:-

Non-preemptive means no suspension, when a process is on Execution by CPU, the process will not be suspended, till the process completely executed. Any Algorithm which follows this logic is called non-preemptive Algorithm.

=) CPU Scheduling Algorithms:-

There are many CPU scheduling Algorithms. Among them, the following four are better CPU Scheduling Algorithms,

1. FCFS (First come First served) Algorithm
2. priority Algorithm

} Non-preemptive Algorithms

3. SJF (Shortest Job First) Algorithm
4. RR (Round Robin) Algorithm

} ~~Non~~ preemptive Algorithms

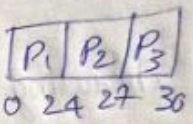
a) FCFS Algorithm:-

Logic:- The first process entered into the system, will be executed first.

Ex:-

Process	E.T. (Execution Time)
P_1	24s
P_2	3s
P_3	3s

Gantt chart:-



waiting times:-

P_1 – 0
P_2 – 24
P_3 – 27

Average time:-

= Total waiting time / Number of processes

$\Rightarrow \frac{51}{3} = 17s$

Turn around times:-

P_1 – 24s
P_2 – 27s
P_3 – 30s

3)51(17
51
0

21/11/21

b) F

The pr
Based
will b
having
first.

21/11/21

b) Priority Algorithm: (non-preemptive Algorithm)

Logic:-

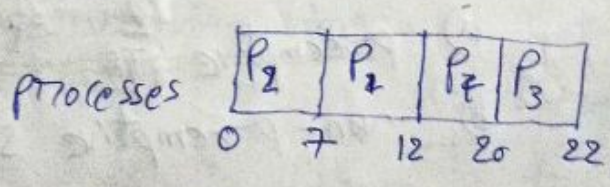
According to this Algorithm The processes will be having one priority value. Based upon the priority value only, The processes will be Executed, by the CPU. Which processes having the highest priority will be Executed first.

Problem:-

processes	E.T.	priority
P_1	5	2
P_2	7	1
P_3	2	4
P_4	8	3

Solution:-

Gantt chart:-



W.T. ⇒ Waiting Time

P_2 - 0s
P_1 - 7s
P_4 - 12s
P_3 - 20s

Average waiting Time:- $\frac{0+7+12+20}{4}$

$\Rightarrow \frac{39}{4} = 9.75$s

Turn Around Times:-

$P_2 - 7$

$P_1 - 12$

$P_4 - 20$

$P_3 - 22$

22/11/21

c) Shortest Job First:- (SJF) (preemptive)

According to SJF, the Smallest Execution time process is Executed first, & then the next Smallest Execution time process is Executed & so on.

There are 2 categories:-

a) Preemptive SJF

b) Non-Preemptive SJF

a) Preemptive SJF:-

According to preemptive SJF, when a process is being Executed by CPU, & at the same time, if a process is arrived with less Execution time, then the Currently

Executing

the Small

Ex:-

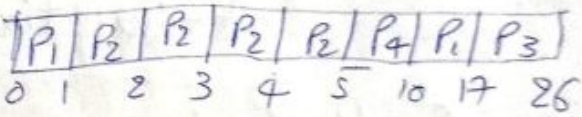
Gant

Executing process is preempted (suspended) & the Smallest Job is assigned to CPU.

Ex:-

Processes	P_1	P_2	P_3	P_4
Execution time	8	4	9	5
Arrival time	0	1	2	3

Gantt chart:-



waiting times:-

Right left Arrival time

$P_1 \Rightarrow 10 - 1 - 0 = 9s$

$P_2 \Rightarrow 4 - 3 - 1 = 0s$

$P_3 \Rightarrow 17 - 2 = 15s$

$P_4 \Rightarrow 5 - 3 = 2s$

Average waiting time:-

$$\frac{9 + 15 + 2 + 0}{4} = \frac{26}{4} = 6.5s$$

Turn around time:-

$P_1 = 17s$

$P_2 = 4s$

$P_3 = 24s$

$P_4 = 7s$

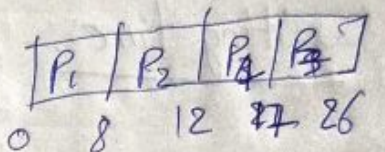
b) Non-preemptive SJF:-

According to Non-preemptive SJF, once a program is assigned to CPU, the process will not be suspended before it's completion.

Ex:-

processes	P_1	P_2	P_3	P_4
Execution time	8	4	9	5
Arrival time	0	1	2	3

Gantt chart:-



waiting times:-

$P_1 = 0s$

$P_2 = 8s - 1s = 7s$

$P_3 = (17s - 2s)$ $17 - 2 = 15s$

$P_4 = (12s)$ $12 - 3 = 9s$

Avg. waiting time $= \frac{31}{4} = 7.75$

Turn Around times:-

$P_1 \Rightarrow 0 + 8 = 8s$

$P_2 \Rightarrow 7 + 4 = 11s$

$P_3 \Rightarrow 15 + 9 = 24s$

$P_4 \Rightarrow 9 + 5 = 14s$

d) Round-Robin Algorithm:-

It is designed for time sharing systems. There is a concept called "time slice"/"time quantity", based upon this time, CPU is allocated to a process for Execution. once the time quantity expires, then the CPU suspends, the process & takes another process for Execution. The quantity is represented as 't'.

Conditions:-

if $t <$ burst time, upto 't' process is Executed.

if $t >$ burst time, upto burst time the process is Executed.

Advantages:-

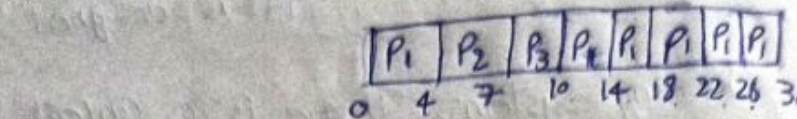
waiting time of processes is reduced.

Ex:-

Processes	P_1	P_2	P_3
Execution time	24	3	3
~~Arrival~~ time	0	1	2

$t = 4$

Gantt chart:-



waiting times:-

$P_1 = 6 - 0 = 6s$

$P_2 = 4 - 1 = 3s$

$P_3 = 7 - 2 = 5s$

14s

Avg. waiting times:- $\frac{14}{3} = 5.66 / 4.6$

Turn around times:-

$P_1 = 30s$ | 30s

$P_2 = 7s$ | 6s

$P_3 = 10s$ | 8s

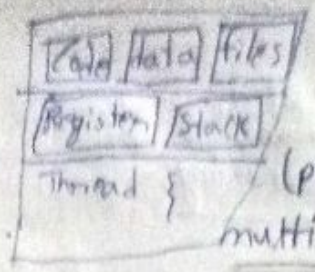
23/11/21

Thread:-

Def:- The thread is defined as a Light weight process. It is represented as 'T'.

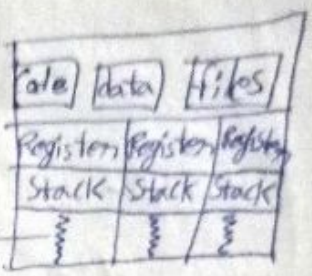
Generally, a process is nothing but a "traditional process", as a heavy weight process i.e, usually a process will be having

more work load.



Single Thread:-

multi thread:-



Benefits of multi-threading:-

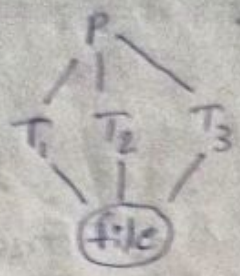
The following are:-

a) Responsiveness
b) Resource sharing
c) Economy
d) Utilization of multiprocessor architecture.

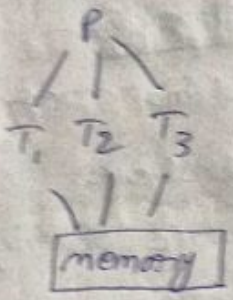
a) Responsiveness:-

when a process is split into threads then each thread must be assigned with some work. Example, A Thread T_1 is loading an image & T_2 may be interacts with the user, So, whenever user interacts the process the user has to feel that the process is responding.

b) Resource sharing:-



– process
– Threads
– s/w resource



– process
– Threads
– h/w resource

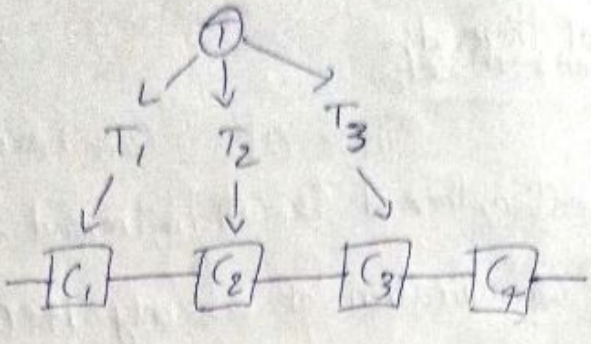
Resource allocated to a process can be shared by all the threads.

c) Economy:-

The costliest source is memory. once it is shared by all the threads. So, multithreading is definitely economic.

d) utilization of multiprocessor Architecture:-

process P — Reading → T_1
— Displaying → T_2
— writing → T_3



when multiple cpu's (processors) are available. one thread can be assigned to execute on 1 cpu, and other thread can be assigned on other cpu & soon. So that multiple threads will be executing on multiple cpus. So, task can be completed very fastly.

⇒ User level thread & Kernel level thread;-

Thread ← Thread creation, Thread execution } very fast
User level thread
OS
↑
Kernel level thread ← Thread creation, Thread execution } slow

User-level thread;-

A thread can be represented in user level & it can have it own thread creation, thread scheduling etc. To support this, there is a thread library. This is not connected to os as well as Kernel.

Note;- This user level thread is very fast.

Kernel-level threads:-

These also can have thread creation, thread execution, thread scheduling etc. But, with communication of OS only it can perform all it's actions.

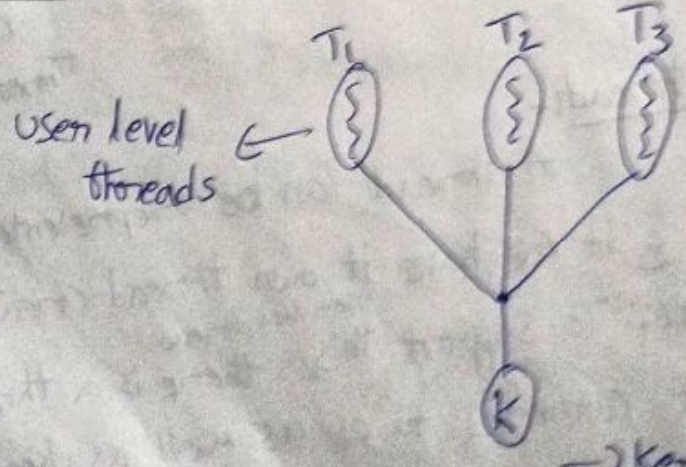
Note:- These are slow processing threads when compared with user level threads.

=> multi-threading models:-

3 different multi-threading models,

a) many to one model
b) one to one model
c) many to many model

a) many to one model:-



-> Kernel level thre

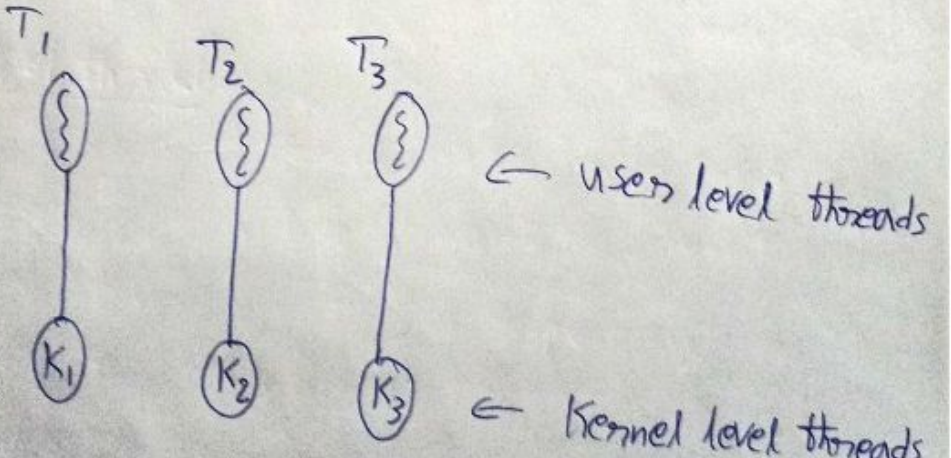
Represents
-> many -> user level threads

-> one -> Represents kernel level threads

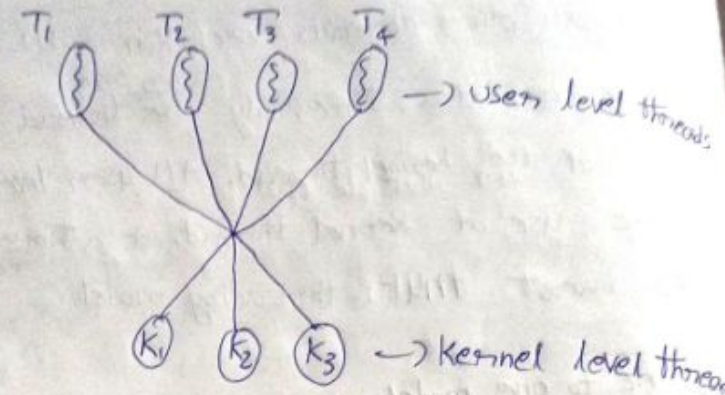
when the user executes a System Call "Blocking System()" then all other threads execution will be stopped. More over at a time, only one user signed thread can make use of the Kernel thread. All user level threads can't make use of Kernel thread, at a time. This is the worst ~~multi~~-threading model.

b) one to one model:-

This is the best multi-threading model, among all the models. Each user thread will have individual Kernel thread. If one particular user thread calls the System Call "Blocking System()", the process of that Kernel thread will be only blocked. The remaining threads will not be affected.



c) many to many model:-



In this model, any user level thread can make use of any kernel level threads.

If one user thread calls to System call "blocking system" only one kernel thread will be blocked. The remaining user threads will make use of other kernel threads. It is better multi-threading method.