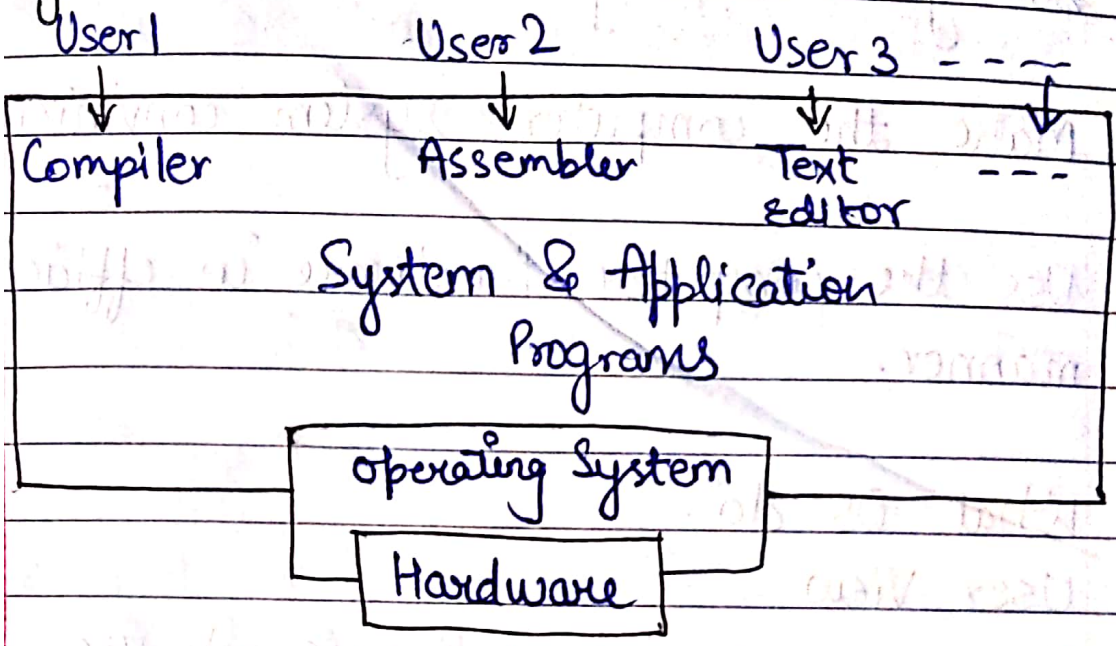
Page No.
Date: 7/1/20

Operating System

Abstract View of Components of a Computer System



I/O Device :- Secondary Storage - Harddisk

System Program :- The operating system programs running in the background without user's knowledge.

Operating System
OS provide an environment within which other programs can do useful task or execute.

Operating system is a program that acts as an intermediate between users and computer hardware.

OS goals

1. Execute user programmers and make solving user programs easier.
2. Make the computer system convinient to use
3. Use the computer hardware in efficient manner.

What OS do?

User View

1. User owns convinience & ease of use.
2. User doesnot care about resource utilization

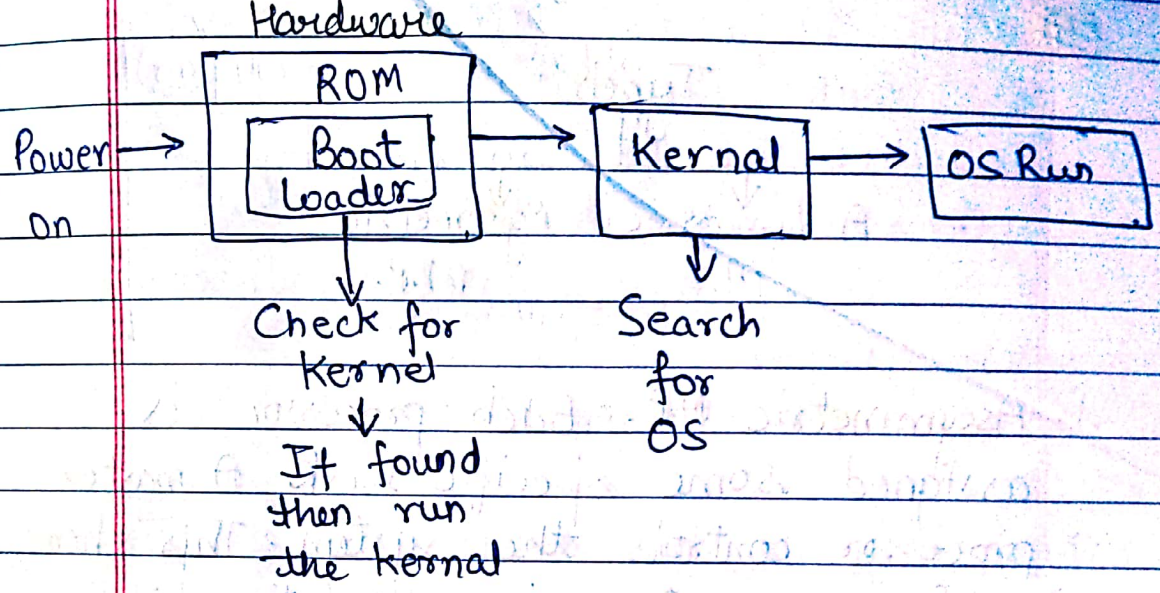
System View

1. How to deal with multiple users at a time.
2. How to manage more than one process at a time

* The one program running at all the times on the computer is known as kernel. Everything else is either system program or an application program.
* Bootstrap program locates & loads into memory, the OS kernal.
* Location of boot strap programme is ROM or EEPROM in the system hardware.

ROM - Read Only Memory
EEPROM - Electronically Erasable Programmable Read Only Memory

To start a Computer



2 8-1-20

Computer System Architecture

① Single Processor System :- On a single processor system there is one main CPU capable of executing processes. Almost all system have these specific or special purpose processor named as device specific processors such as disk, keyboard & graphic controller.

② Multiprocessor System :- Also called parallel system or tightly coupled systems.

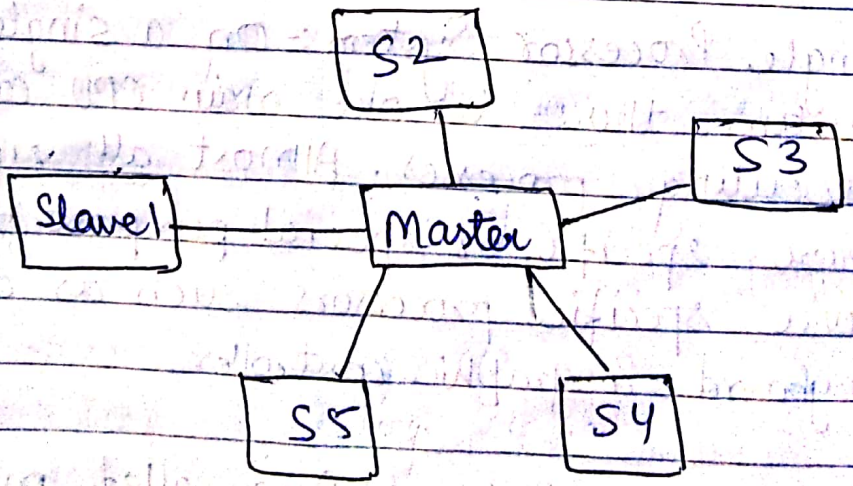
Advantages:-

(1) Increased Throughput.
(2) Economy of scale.
(3) Increased reliability (Fault Tolerant)

Types

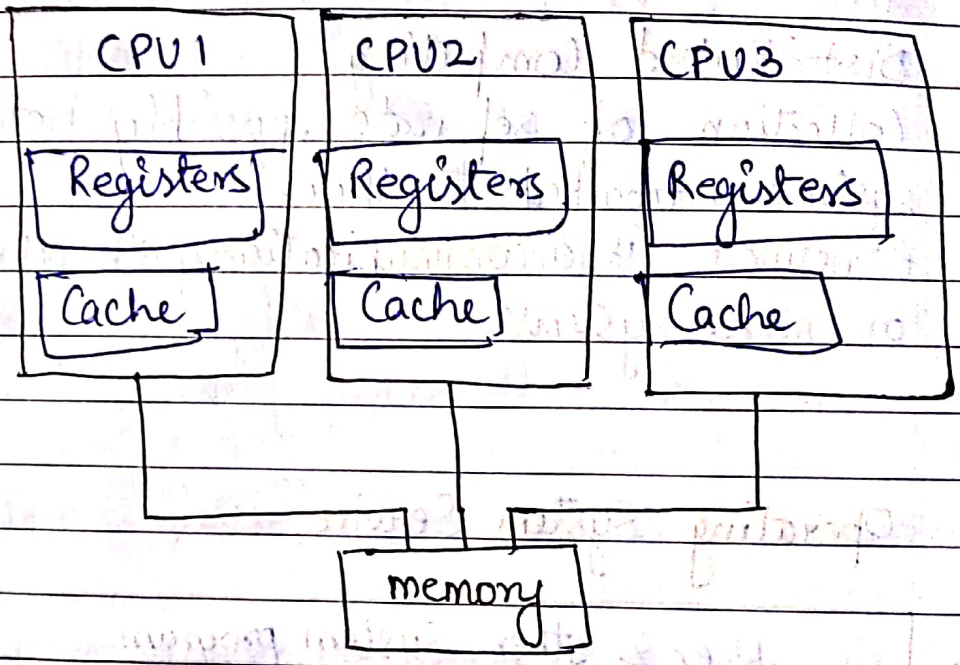
- Assymmetric M.P.
- Symmetric M.P.

1. Assymmetric M.P :- Each processor is assigned some specific task. A master processor controls the system. This scheme define a master slave relationship. The master processor schedule & allocate to the slave processor.



Page No. MIRAJ
Date: PREMIUM

Symmetric M.P:- Each processor perform all tasks within the O.S. All processors are P.s. peers i.e. no master slave relationship exists between processors.



Including multiple computing cores in a single chip are similar to multiprocessing system. It can be more efficient than multiple chips with single cores because on-chip communication is faster than between chip comm. One chip with multiple cores uses less power than multiple single core chips.

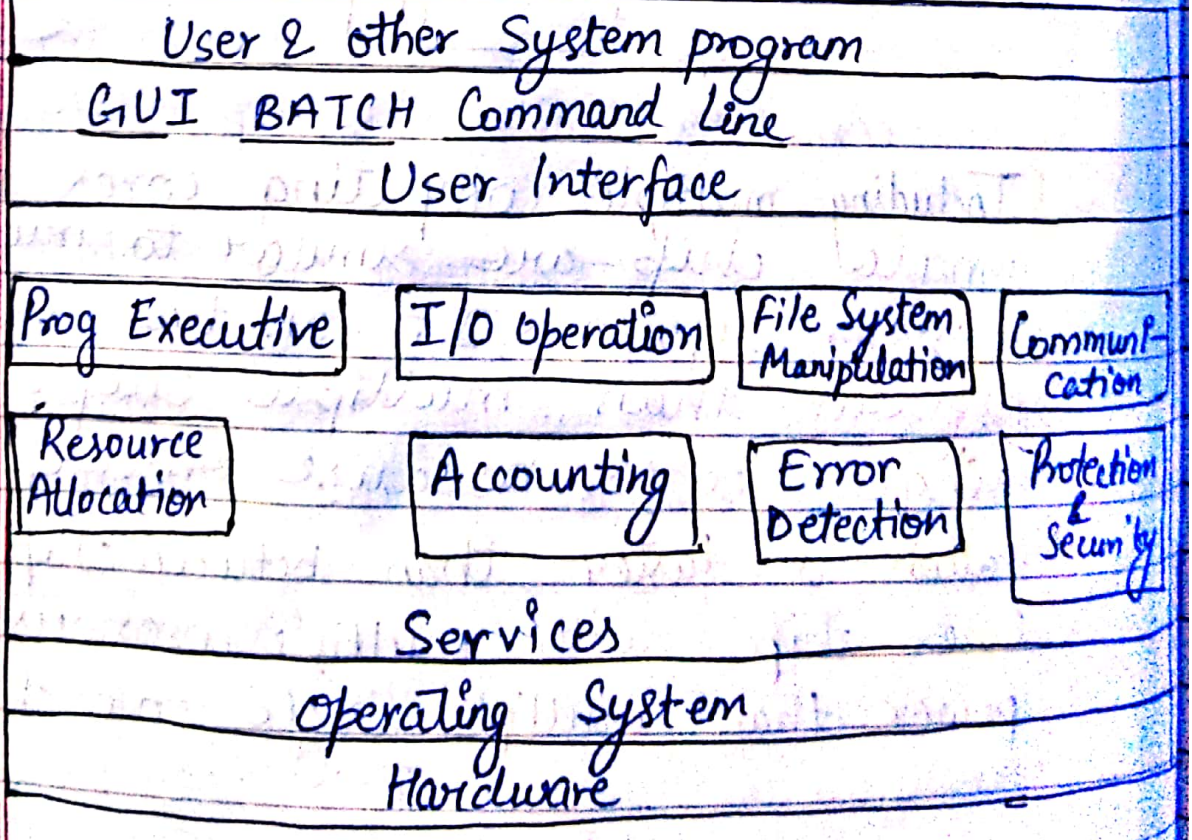
Special Purpose System

1. Real Time Embedded Systems
2. Multimedia Systems
3. Handheld Systems

Distributed Computing
Collection of separate, possibly heterogeneous systems networked together.
A network is a communication path between 2 or more systems.

3
13/1/20

Operating System Service



Program Execution :- The system must be able to load a program into main memory & to run that program & execution either normally or abnormally (indicating error).

I/O Operations :- A running program may require I/O which may involve a file or an I/O device.

File System manipulation :- Programs need to read & write files & directories create & delete them. Searching & list file information.

Communication :- Processes may exchange info. on the same computer or b/w computers over a network. Communication may be via shared memory or through message passing.

Resource Allocation :- When multiple users or multiple processes running concurrently resources must be allocated to each of them. Some main resources are CPU, main memory, file storage & I/O devices.

Accounting :- To keep track of which users use how much & what kind of computer resources.

Page No.
Date 14-1-20

Error Detection: OS needs to be constantly aware of possible errors, these errors may occur in the CPU, memory, I/O devices or in the user programs. For each type of error OS should take the appropriate action to ensure correct & consistence computing.

Protection & Security
Protection :- Protection involves ensuring that all access to system resources is controlled.
Security :- Security of the system from outsiders require user authentication.

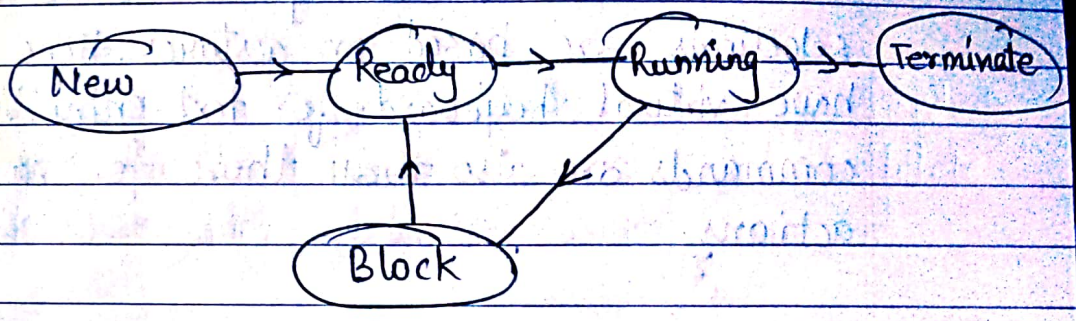
14/1/20
4

Process :- Process is an instance of program of a particular time.

State of a process :-
1. New
2. Ready
3. Running
4. Block
5. Termination
6. Suspended Ready
7. Suspended Block.

1. New :- Process at new state will be in the secondary memory.

2. Ready :- Process at ready state it will be bring in the primary memory from secondary memory.



15-1-20 5

Operating System Interface :-

① CL : (Command Line Interface) :- It allows users direct command entry. It is sometimes implemented in kernal & sometimes by system programs. It primarily fetches a command from user & executes it. Sometimes commands are built in sometimes just names of programs hadd to be entered.

② GUI (Graphical User Interface) :- It usually deals with mouse, keyboard & monitor different

icons represents files, programs & actions. Various mouse buttons over objects in the interface cause various actions like provide information, options, execute functions & open directory or open foulder.

(3) Touch Screen Interface!- In touch screen devices mouses is not required actions & selections are based on gestures. Here we have virtual keyboard for text entry. Voice commands are also ~~then~~ there for various actions.

System Calls!-

- System calls provide an interface to the services provided by operating system.
- System calls are written in high level language C or C++.
- We may use system calls with the help of high level application programming interface (API) rather than directly.

⇒ System call sequence to copy the contents of one file to another file:-

(1) Acquire Input file Name:-
Write prompt to screen.
Accept input.

(2) Acquire output file Name
Write prompt to screen
Accept Input.

(3) open the Input file
If file does not exist, abort.

(4) Create output file
If file exist, abort.

(5) Loop until read fails
Read the Input file.
Write to output file.

(6) Close files

(7) Write Completion message to screen.

(8) Terminate Normally.

Types of System Calls :-
There are 6 type of System Calls :-

(1) Process Control :-
End, abort
load, execute
wait, signal.

(2) File Manager :- create, delete
open, close
read, write
get file attributes, set file attribute

(3) Device Management : request, release
getdevice attributes, set device attributes
logically attach or detach device.

(4) Information Maintenance :
get time or date, set time or date,
get process attributes, set process
attributes.

(5) Communication :-
create, delete, communication connection,
send, receive message
attach or delete remove device.

(6) Protection & Security.
get / set permission.

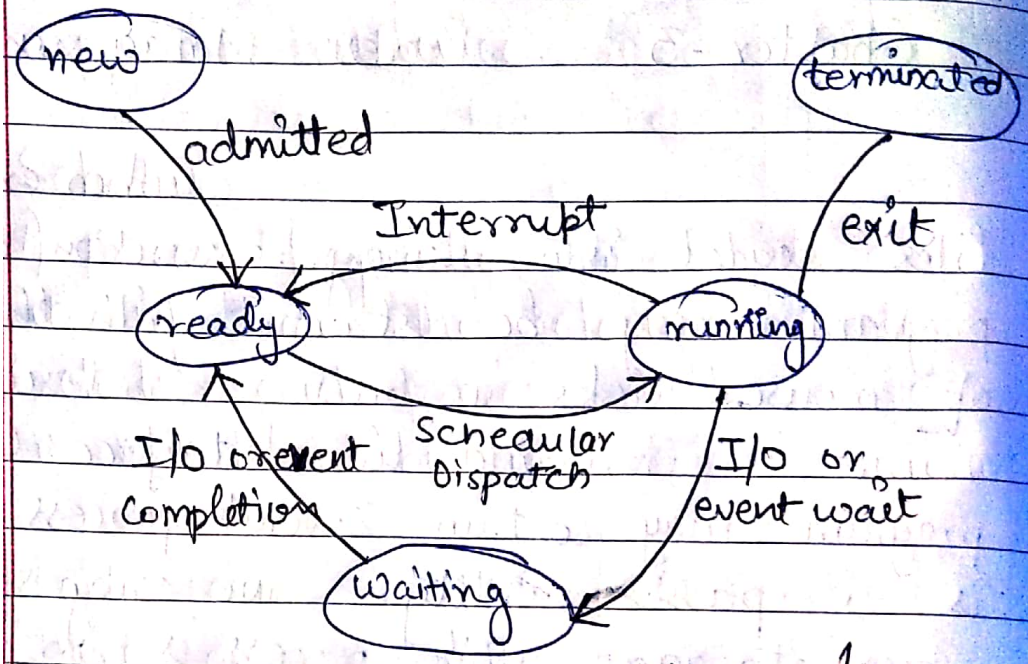
Unit - 2.

Chapter - 3 Process Management

when executable files loaded into memory. Execution of program may be started with the help of mouse clicks in GUI or entering its name in Command line Interface. One program may contain several process. Program is a passive entity save only secondary storage while process is an active entity that is running in the main memory.

Process States

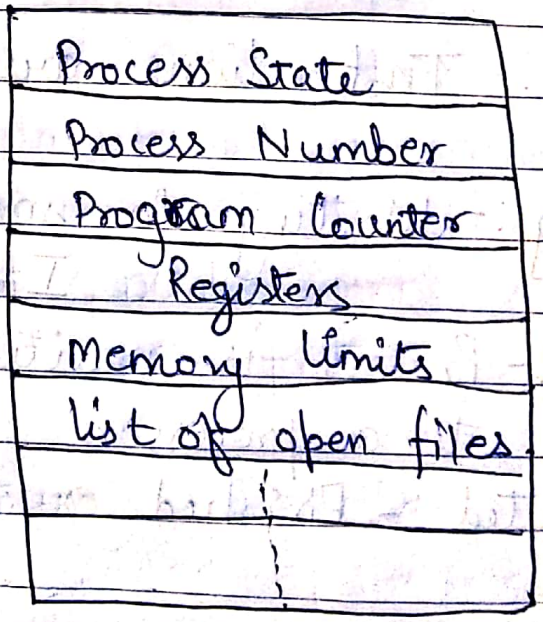
1. New:- The process is created / just arrived.
2. Running :- Instructions are being executed.
3. Waiting :- Waiting for some events such as I/O completion.
4. Ready :- Process is waiting to be assigned to a processor.
5. Terminated :- Finished execution.



Process State Diagram / Process life cycle.

Process Control Block (PCB)

(Representation of a process in Memory)



Page No. MIRAJ
Date: FRIDAY

Process state may be running waiting or ready. Program counter contains location of instruction to be executed next.
CPU scheduling information, priorities, Round Robin
Memory limits, m/m associated with Priority, SFC, SJF, ~~Round Robin these memory limits~~ process in RAM.

24/1/20 6.

Process Scheduling
Process scheduling is to switch the CPU among processes so frequently that users can interact with each program while running. for this process schedular selects an available process for the program execution on the CPU.
For a single processor system there is only one process at a time which can be in running state. If there are more processors the rest have to wait until the CPU is free & can be rescheduled.

Scheduling Queues
As the process enters in the system, they are kept into a job queue which consist of all the processes in the system.
All the processes that are in ready state are kept in the ready queue.

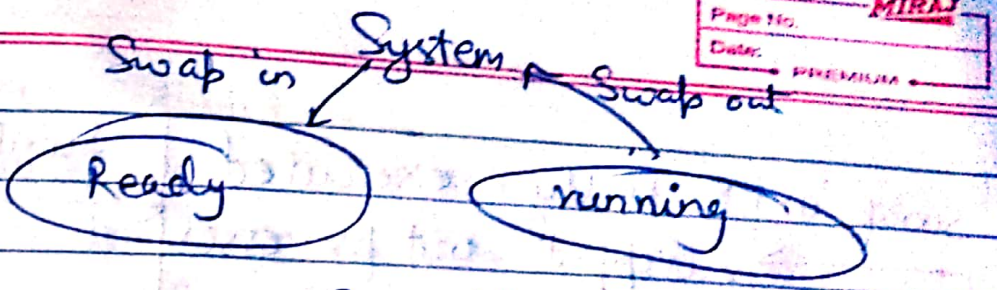
Device Queue:- Particular to a specific device.

Schedulers
A process frequently changes its state during its life time. The selection process that selects which process will go for execution is carried out by appropriate schedulers.

Long Term Scheduler / Job scheduler
There are generally so many processes at a time in a computer system that must be kept for later execution. The job scheduler select processes from this job pole & loads them into m/m for execution.

Short term / CPU Scheduler
CPU scheduler selects a process from the ready queue and allocate the CPU to the process. It generally works once in 100ms. So it must be fast.

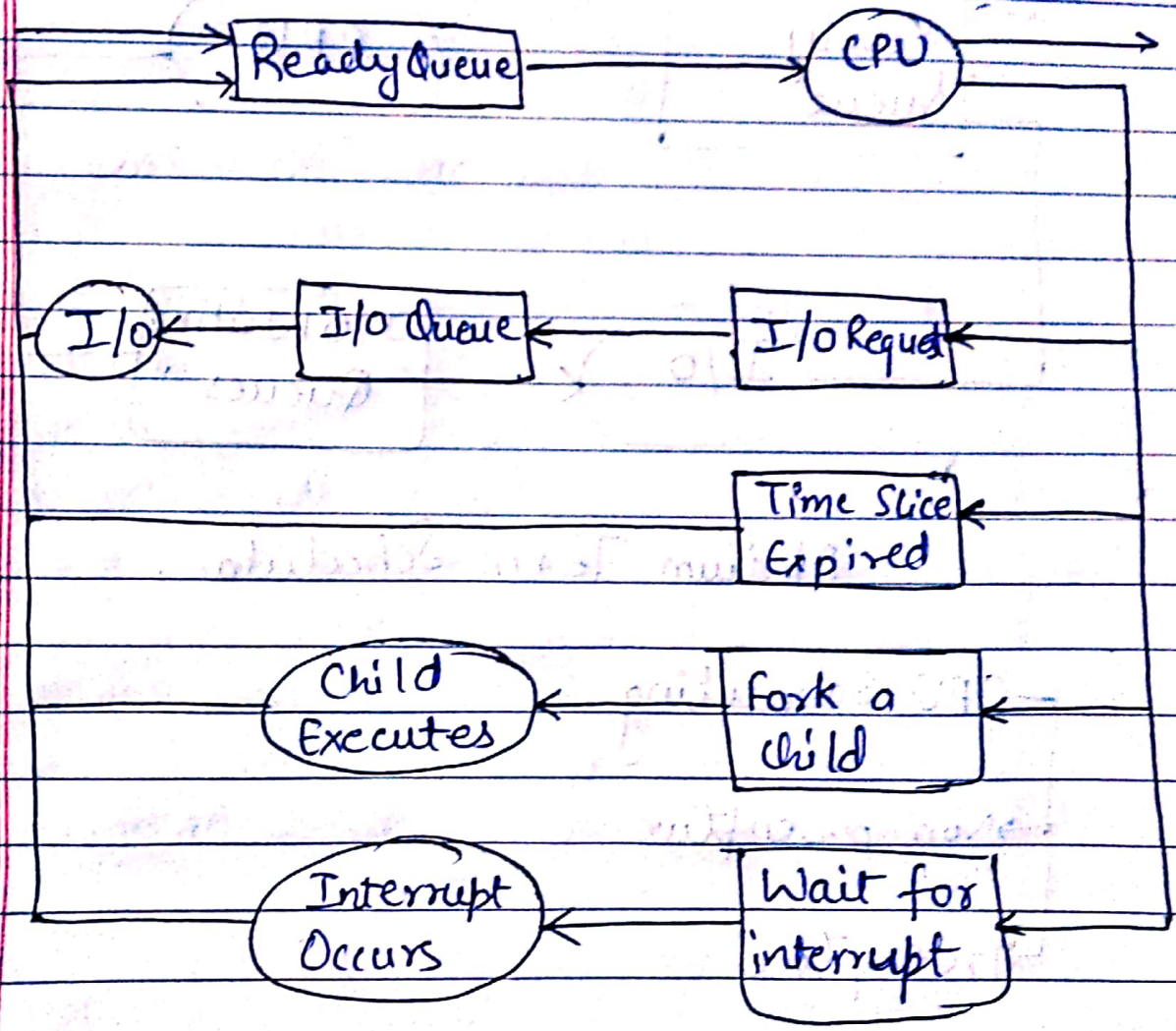
Medium Term Scheduler
It can be added if degree of multiprog needs to decrease.

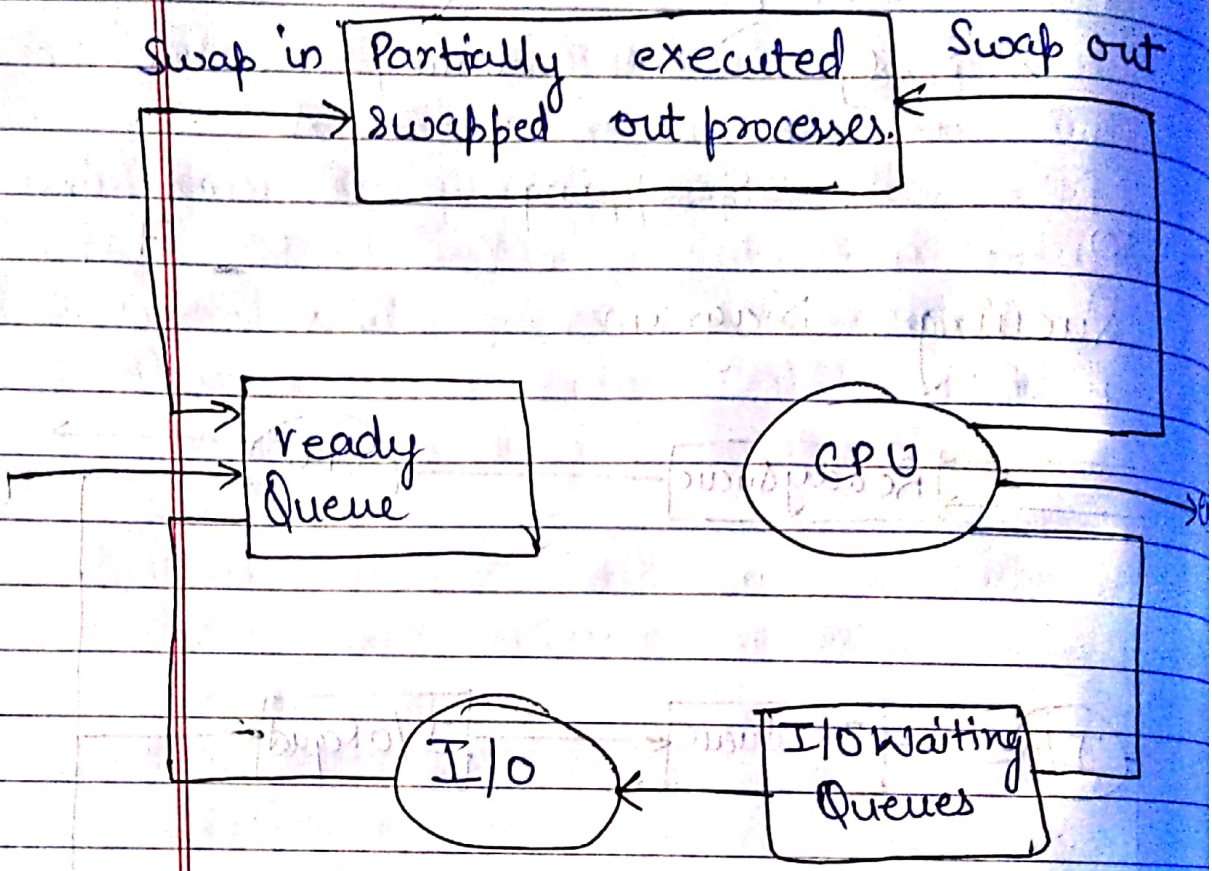


Swapping

22-1-20

Queuing Diagram





Medium Term Scheduler.

CPU Scheduling

→ Non preemptive

→ Preemptive

(1) Non-preemptive :- Once the CPU has been allocated to the process, the process keeps CPU until it releases the CPU either by terminating or by switching its state to waiting.

Dispatcher

Dispatcher gives control of the CPU to the process selected by short term or CPU schedular. With time taken by the dispatcher to stop one process and start another process running in known as dispatch latency.

Tutorial-1

1) What are the 3 main purpose of an OS?

A-1 i) Execute user program and make solving problem easier.

ii) OS make computer system easy to use.

iii) OS use computer hardware in efficient manner.

2) What is diffence between OS for main frame computer and personal computer?

A-2

Mainframe	Personal
1) Easy to built OS for mainframe.	1) Difficult to make OS for PC.
2) Not user friendly.	2) Easily understand by user
3) As, is not user friendly So, there is no response time.	3) Reserve response time, for an interactive user.
4) Not to handle time sharing for resources.	4) It must have to handle or switching rapidly b/w diff. jobs.

3) list 4 steps that are necessary to run a program on a completely dedicated machine.

A-3 a) Reserve machine (CPU) time of execusion.

b) Manually load program into memory.

c) load starting address and begin execution.

d) Monitor and control execution of program from console.

4) How does the distinction b/w kernal mode and usermode function as a rudimentary form of protection system?

A-4.

Kernal	Usermode
i) Some imp instruction only run in kernal mode.	While in user mode instruction will not work.
ii) Hardware control only access by OS while in kernal mode.	No interaction with hardware.
iii) CPU can perform various process in Kernel mode.	There is limited of CPU to perform process in user-instruction mode.
iv) When computer application is running it is in user mode. Ex :- browsering Internet running software.	If process request some hardware such as RAM, printer then system goes to user mode to kernel mode.

Tutorial 2

Q1. What is the purpose of system calls?

Ans: (1) Freeing users from studing low-level programming.
(2) It greatly increases system security.
(3) These interfaces make programs more portable.
(4) Networking.
(5) Process creation & management.
(6) Main m/m management
(7) Device handling (I/O).

Q2. What are the five major activities of an operating system in regard to process management?

=> • The creation and deletion of both user and system processes.
• The suspension and resumption of processes.
• The provision of mechanisms for process synchronization.
• The provision of mechanisms for process communication.
• The provision of mechanisms for deadlock handling.

Q3:- What is the purpose of the command interpreter? Why is it usually separate from the kernel?

A- It reads commands from the user or from a file of commands and executes them, usually by turning them into one or more system calls. It is usually not part of the kernel since the command interpreter is subject to changes.

Q:- What system calls have to be executed by a command interpreter or shell in Unix order to start a new process?

A:- In Unix systems, a fork system call followed by an exec system call need to be performed to start a new process. The fork call clones the currently executing process, while the exec call overlays a new process based on a different executable over the calling process.

Q:- What is the purpose of system program?

A:- System programs can be thought of as bundles of useful system calls. They provide basic functionality to users so that users do not need to write their own programs to solve common problems.

Date 27/1/20

Scheduling Criteria

① CPU utilization :- We want to keep the CPU as busy as possible.

② Throughput :- No. of processes that are completed per unit time

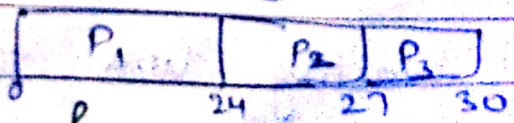
③ Turn around time :- The time interval from submission of a process to the time of completion

④ Waiting time :- It is a ~~std~~ sum of intervals spend waiting in the ready queue.

⑤ Scheduling algorithm

① FCFS :- (First come First Serve)
↳ (Non preemptive)

Process	Starting Time	Burst Time (CPU Burst)
P_1	0	24
P_2	0	3
P_3	0	3



Waiting Time of P_1 = 0

P_2 = 24

P_3 = 27

Average waiting Time = $\frac{24+27+0}{3} = \frac{51}{3} = 17$.

Page No.
Date:
MIRAJ
PREMIUM

Turn around Time $P_1 = 24$
$P_2 = 27$
$P_3 = 30$

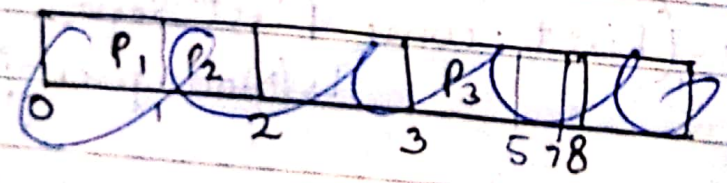
Average T.A. Time $= \frac{24+27+30}{3}$

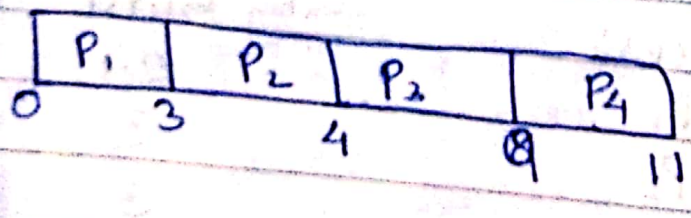
$= 2 \frac{8+}{3}$

$= 27$.

Q:-

Process	Arrival Time	Burst Time
P_1	0	3
P_2	1	1
P_3	3	5
P_4	5	2





Waiting Time for $P_1 = 0$
" $P_2 = 2$
$P_3 = 1$
$P_4 = 4$

A.W.T $= \frac{2+1+4}{4} = \frac{7}{4} = 1.75$

Turn around Time $P_1 = 3$

$P_2 = 3$

$P_3 = 6$

$P_4 = 6$

Average Turn Around Time $= \frac{3+3+6+6}{4} = \frac{18}{4} = \frac{9}{2}$

$= 4.5$

T.A.T = Finish Time - Arrival Time

② SJF :- (Shortest Job First)

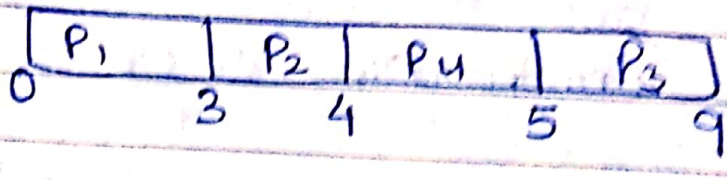
- ① Non Premptive
- ② Premptive

Non-Premitive

Process	Burst Time	W.T	T.A.T
P_1	24	6	30
P_2	3	0	3
P_3	3	3	6
		3	13

Process	Arrival Time	Burst Time
P_1	0	3
P_2	1	1
P_3	2	4
P_4	4	1

W.T. for $P_1 = 0$
" " $P_2 = 1$
" " $P_3 = 3$
" " $P_4 = 0$



T.A.T for $P_1 = 3$
" " $P_2 = 3$
" " $P_3 = 7$
" " $P_4 = 1$

Average W.T $= \frac{0+2+3+0}{4} = \frac{5}{4} = 1.25$

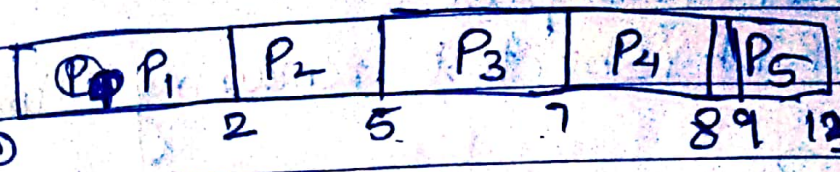
Average A.T $= \frac{3+3+7+1}{4} = \frac{14}{4} = 3.5$

Q :-

Process	Arrival Time	Burst Time
P_1	0	2
P_2	1	3
P_3	1	2
P_4	5	1
P_5	9	4

FCFS & SJF (Non Premptive)

FCFS



W.T for $P_1 = 0$

$P_2 = 1$

$P_3 = 4$

$P_4 = 2$

$P_5 = 0$

$$\frac{0+1+4+2+0}{5} = \frac{7}{5} = 1.4$$

T.A.T. for $P_1 = 2$

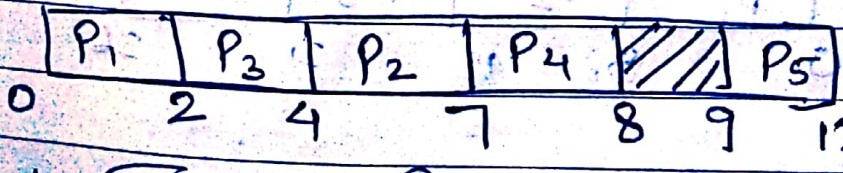
$P_2 = 4$

$P_3 = 6$

$P_4 = 3$

$P_5 = 4$

$$\frac{2+4+6+3+4}{5} = \frac{19}{5} = 3.8$$



W.T for $P_1 = 0$ T.A.T = $P_1 = 2$

$P_2 = 3$ $P_2 = 6$

$P_3 = 1$ $P_3 = 3$

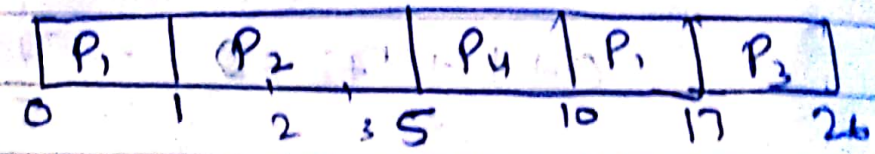
$P_4 = 2$ $P_4 = 3$

$P_5 = 0$ $P_5 = 4$

$= 1.2$ $\frac{18}{5} = 3.6$

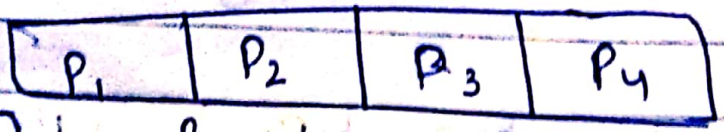
Q:- SJF Premptive
SRTF (Shortest remaining Time First

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5
P5		



W.T	T.A.T	
9	17	SJF
0	4	7.75
15	24	14.25
2	7	
6.5	13	

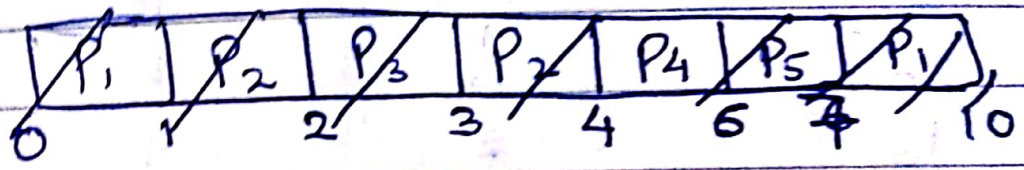
FCFS

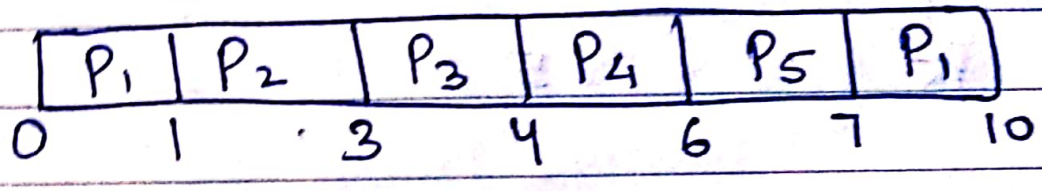


	W.T	T.A.T
P_1	0	8
P_2	7	11
P_3	10	19
P_4	18	23
AVG	$\frac{35}{4} = 8.75$	$\frac{61}{4} = 15.25$

Q

Process	Arrival Time	Burst Time
P_1	0	4
P_2	1	2
P_3	2	1
P_4	3	2
P_5	5	1





W.T	TAT
6	10
0	2
1	1
1	2
1	

$\frac{9}{5} = 1.8$

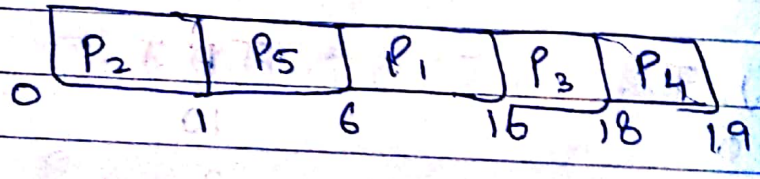
Page No.
Date:

Priority Scheduling

- Non Preemptive
- Preemptive

Non Preemptive

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	12

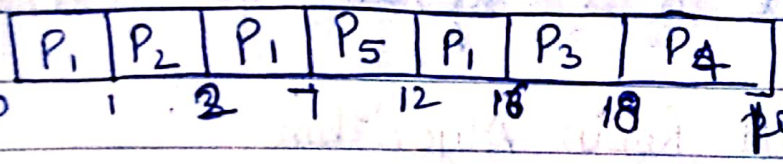


	WT	TAT
P_1	6	16
P_2	0	1
P_3	16	18
P_4	18	19
P_5	1	6

$\frac{6+16+18+1+0}{5}$ $\qquad$ $\frac{16+1+18+19+6}{5}$

$\frac{41}{5} = 8.2$ $\qquad$ $\frac{60}{5} = 12$

Process	Arrival Time	Burst Time	Priority
P_1	0	10 - 9, 5	3
P_2	1	1	1
P_3	2	2	4
P_4	5	1	5
P_5	7	5	2



	WT	TAT
P_1	6	16
P_2	0	1
P_3	14	16
P_4	13	14
P_5	0	5

$\frac{6+14+13}{5}$ $\frac{16+1+16+14+5}{5}$

$\frac{33}{5} = 6.6$. $\frac{52}{5} = 10.4$.

* Indefinate blocking & Starvation

Major problem with priority scheduling algo is indefinate blocking or starvation condition.

A process that is ready to run but waiting for the CPU for longer duration can be

considered blocked.
A priority scheduling algo can leave the low priority task & processor waiting indefinately.
A solution to the problem of indefinate blocking of low priority processes is aging. Ageing is a technique of gradually increasing the priority of processes that wait in a system for a long time.

Round Robin Algorithm

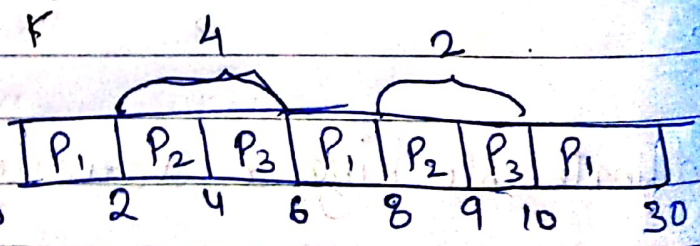
↳ Always Preemptive

'Time Quantom / Time Slice Concept'

Q:-

Process	Burst Time
P_1	24
P_2	3
P_3	3

Time slice = 2.



W.T	T.A.T
6	30
6	9
7	10
$\frac{19}{3} = 6.3$	$\frac{49}{3} = 16.3$

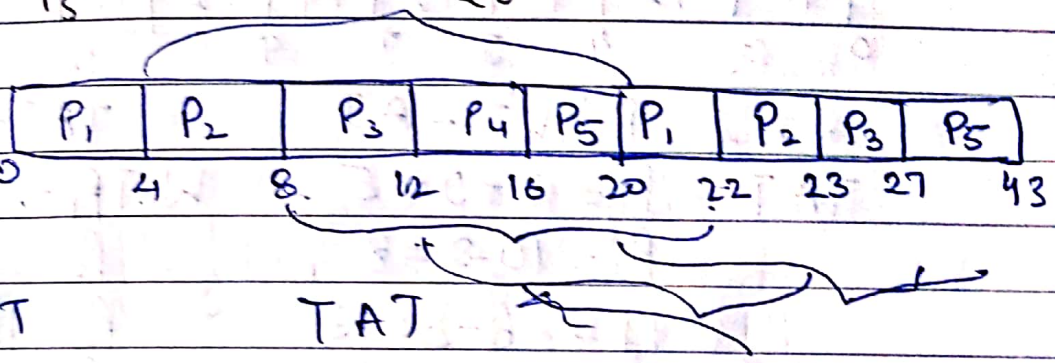
27
-16
11

MIRAJ
Page No.
Date:
PREMIUM

2:-

Process	Burst Time
P_1	6
P_2	5
P_3	8
P_4	4
P_5	20

Time slice = 4.



WT	TAT
16	22
14+4 = 18	
11+8 = 19	
12	
7	

Round Robin Algo

Process	Arrival Time	Burst Time
P_1	0	3
P_2	2	3
P_3	3	6
P_4	4	7

Time Quantum = 2.

Ready Queue

P_1	P_2	P_1	P_3	P_4	P_2	P_3	P_4	P_3	P_4

P_1	P_2	P_1	P_2	P_4	P_2	P_3	P_4	P_3	P_4

0 2 4 5 7 9 12 14 16 19

Finish T - AT

TAT $P_1 = 5 - 0 = 5$

$P_2 = 10 - 2 = 8$

$P_3 = 16 - 3 = 13$

$P_4 = 19 - 4 = 15$

TAT - BT

W.T $P_1 = 5 - 3 = 2$

$P_2 = 8 - 3 = 5$

$P_3 = 13 - 6 = 7$

$P_4 = 15 - 7 = 8$

Q:-

Process	AT	BT
P_1	3	3
P_2	5	4
P_3	4	5 - 4
P_4	4	6
P_5	2	8 - 4

Time Quantom = 4

Ready Queue

P_5	P_1	P_3	P_4	P_2	P_5	P_3	P_4

4 4

Gantt Chart

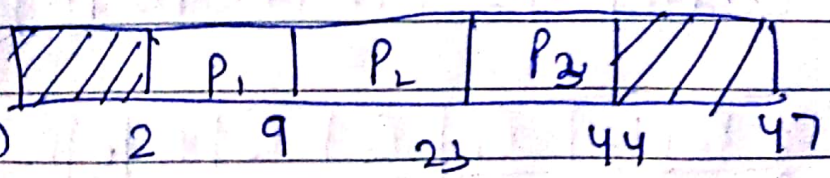
P_5	P_1	P_3	P_4	P_2	P_5	P_3	P_4

2 6 9 13 17 21 25 26 28.

	Fini	WT
TAT.	6	3
	16	12
	22	17
	24	18
	23	15

Q:- Consider 3 processes All arriving at time 0 with total execution time of 10, 20, 30 units resp. Each process spends the first 20% of execution time doing I/O operation. The next 70% of time doing computation & last 10% of time doing I/O again. The OS uses ~~shorte~~ shortest remaining time first algo assumes that all IO operations can be overlap as much as possible for what percentage of time does the CPU remain idle.

	TT	I/O(AT)	CPU Burst	I/O
P_1	10	2	7	1
P_2	20	4	14	2
P_3	30	6	21	3



$\frac{5}{47} \times 100 \Rightarrow 10.64\%$

Q:-

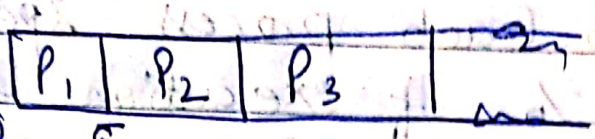
	AT	BT
P_1	0	5
P_2	1	7
P_3	3	4

FCFS

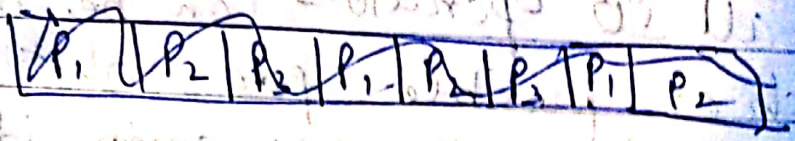
RR (2)

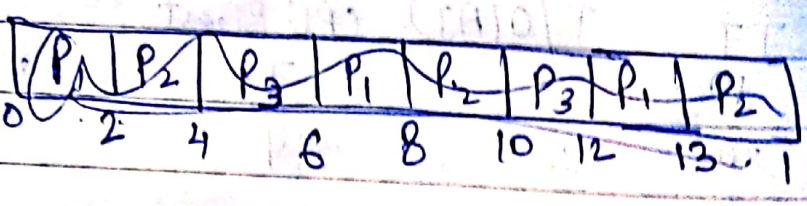
Process Completion Order:

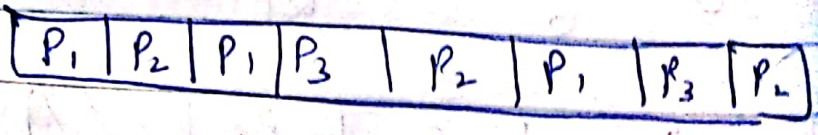
FCFS

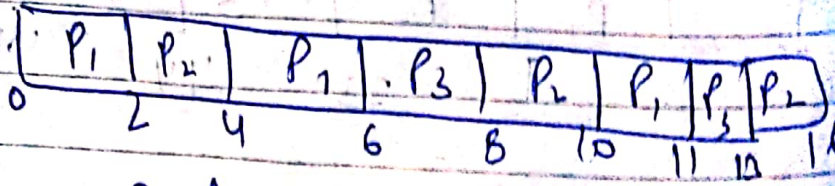


Order of completion : P_1, P_2, P_3









Order of completion = $P_1 P_3 P_2$

Tutorial - 3

Q1:- Discuss ~~three~~ major complications that concurrent processing adds to an operating System.

1. The OS has to keep track of the main m/m address space allocated to each process so that a process does not affect or harm other processes data. This is a necessary for ex:- a virus may try to destroy or modify other process data.
2. Switching from one process to another process leads to time overhead, this requires storing the current register values & loading the register values of the next process from its PCB (Program Control Block).
3. If a running process requires large space in m/m then other processes needs to be dumped back to hard disk which again leads to time ~~over~~ overhead.

Q2:- When a process creates a new process using the fork() operation, which of the following state is shared between the parent process & the child process?

a. Stack
b. Heap
c. Shared Memory Segments.

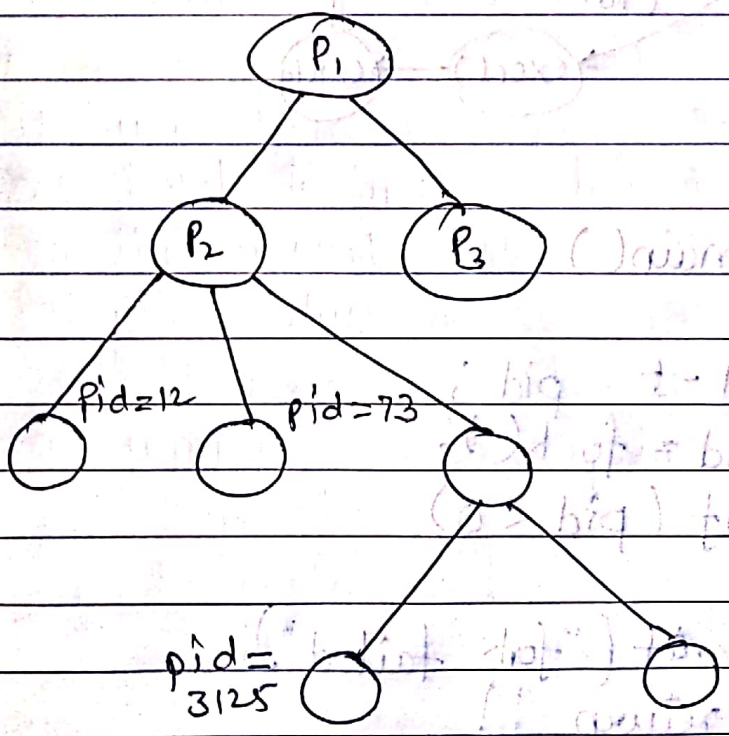
A:- Only shared memory segments are shared.

* Copies of the stack & the heap are made for newly created processes not for newly forked child process.

Operations on Processes

↳ Process Creation
↳ Process Termination

Parent process create child processes which create other subprocesses forming a tree of processes, generally processes are identified & managed by a process identifier i.e. pid



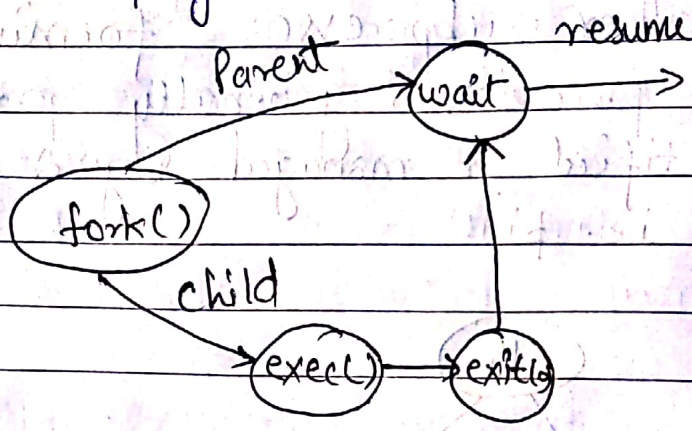
Resource Sharing Options :-

↳ Execution Option
  ↳ parent & child runs concurrently
  ↳ parent will wait for completion of child.

Page No.
Date:
PREMIUM

Fork()

Fork() system call create new processes.
exec() system call used after a fork() to replace the process m/m space with a new program.




```
int main()
{
   pid_t  pid;
    pid = fork();
     if (pid < 0)
{
    printf ("fork failed");
      return !!
   }
else if (pid == 0)
  {
   printf ("printed by child");
}
else
    printf ("printed by parent");
} return 0;
```

MIRAJ
Page No.
Date:
PREMIUM

Process Termination

↳ exit()
↳ abort()

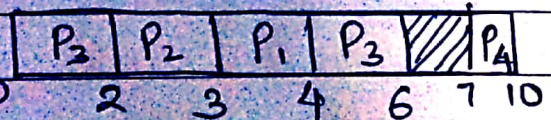
Abort()

Abort is used when child has exceeded allocated resources or task assigned to the child is no longer required. or the parent is exiting and the os does not allow a child to continue if its parent terminates

Process	AT	BT	Priority
P_1	3	1	2
P_2	2	1	2
P_3	0	4	3
P_4	7	3	1

RR1

SJF



FCFS, SJF, SRTF, Priority (Non Preemptive & Preemptive both),

Round Robin (Quantum = 1)

Avg waiting time,
Avg turn around time
No. of context switches
Process completion Order
CPU Idle time (Percentage

} In each case

Page No.
Date 5/12/20

Inter Process Communication (IPC)

① Independent :- It cannot affect or affected by other processes executing in the system or doesnot share data with any other process.

② Cooperating :- It can affect or be affected by other processes executing in the system or processors are sharing data with other processes.

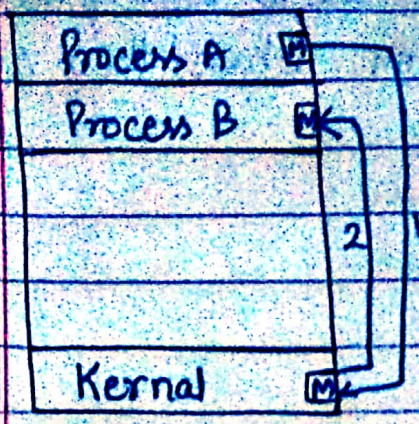
Why Cooperation is required ?

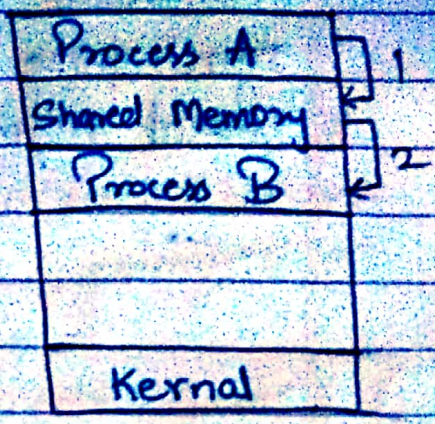
1. Information sharing
2. Computation speedup
3. Modularity
4. Convenience.

Cooperating process require an IPC mechanism so that will allow them to exchange data and information.

Two models for IPC are

1. Shared memory
2. Message passing





Message Passing. Shared Memory.

Message Passing

1. It is useful in exchanging smaller amount of data.
2. Easier to implement.
3. Uses system calls send(), receive() so it is more time consuming.
4. If 2 processes P and Q wants to communicate they need to establish a communication link & then they may exchange information via send

Shared Memory

1. Maximum speed & convinient.
2. Faster than message passing.
3. System calls are required only to establish shared m/m regions.
4. Major issue with shared m/m is to provide

mechanism that will allow user processes to synchronize their action when they acess shared m/m.

Page No.
Date: 10/09/20

Message Passing
It can be implemented with the help of these 3 methods:-
1. Direct or Indirect
2. Synchronous or Asynchronous
3. Buffering.

(1) 1. Direct Communication
Processes must name each other explicitly.
- Send (P, Message): Send a Message to Process P.
- Recieve (Q, Message): Recieve a Message from Q.

- B/w each pair of processes there must be exactly 1 communication link.

- The links may be unidirectional or bidirectional

2. Indirect Communication
- Messages are send & recieved with the help of mailboxes or ports. Each mailbox or port has a unique ID.
- Processes can communicate only if they share a mailbox.

(2) 1. Synchronisation
Message Passing can be of Two types:-
1. Blocking
2. Non-blocking.

Blocking is considered synchronous. We use 2 methods:-
1. Blocking send
2. Blocking recieve

in synchronous type :

Blocking send :- The sender is blocked until the message is recieved.

Blocking recieve :- The sender is blocked until the message is send.

2. Asynchronous.

Non blocking send :- The sender sends the message & continue.

Non blocking reciever :- The reciever recieve valid message or NULL Message.

(3) Buffering

Queue of messages is attached to the link.
This can be implemented in 3 ways :-

1. Zero capacity.
2. Bounded Capacity
3. Unbounded Capacity